Université d'Ottawa · University of Ottawa

Distributed Tree Schemes for DWDM Network Protection and Restoration

By

Yuna Zhang

A thesis submitted to the
School of Graduate Studies and Research
in Partial fulfillment of the requirement for the degree of

Master of Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
Faculty of Computer Science
University of Ottawa

September 2002

Abstract

This thesis proposes and studies distributed spanning tree algorithms for the protection/restoration of Dense Wavelength Division Multiplexing (DWDM) networks. A heuristic algorithm using tokens to construct a maximum spanning tree in a distributed manner as well as the corresponding protection/restoration schemes are proposed. One algorithm involves a triangular structure on which we have proposed a triangular tree protection/restoration scheme. Since we need a root to build a spanning tree, we give three root selection algorithms to select a proper root. Properties of these algorithms are then studied. Performance analysis in terms of mean path length, capacity ratio and restorability of the spanning tree to the mesh network and number of messages transmitted during the construction of the spanning tree are provided in this thesis. We also compare the spanning tree algorithms with another distributed spanning tree algorithm in the performance. It shows that the tree protection schemes can be an alternative to the DWDM network protection/restoration.

Acknowledgements

I am deeply grateful to my supervisor, Dr. Oliver W.W.Yang, for his knowledgeable academic guidance as well as his helpful comments and suggestions during the two years' study. I really appreciate the help from Mr. Peifang Zhou and Shahram Shah-Heydari. Thanks to everyone from the CCNR Lab for the nice cooperation in our CCNR group. The enjoyable working environment makes the study a great memory.

I will give my great thanks to my friends, Sheila Williams, Prof. Tet Yeap, Natalie Ward, Dale Ward, Jean Saint, Nigel Saint, Monica Williams, Peter Philpott as well as the Bible Study Group at the University of Ottawa. Thank you so much for the spiritual support! Thank all my old and new friends for all the help and friendship that I enjoy a lot here.

Lastly, I will contribute all my accomplishment to my family, especially my parents and my lovely daughter Jiawei! I also owe my sibling Yuhua, Yuhui, Kehui and their families a lot! I can never finish my study without the great love, precious understanding and endless support from my family! Thank you so much from my heart and all my life!

This work has been supported financially in part by Nortel Networks.

Table of Content

Table of Acronyms and Abbreviations

		Section of first Appearance
ADM	Add drop Multiplexer	1.3
AS	Active state	4.1
ATM	Asynchronous transfer mode	1.2
CRS	Central root selection	3.0
CST	Conventional spanning tree	1.10
CTT	Central triangle tree	5.3
DiRS	Distributed root selection	3.0
DRS	Downward root selection	3.0
DTT	Distributed triangle tree	5.3
DWDM	Dense wavelength division multiplexing	1.2
EDFA	Erbium-doped fiber amplifier	1.2
FDDI	Fiber distributed data interface	4.0
ID	Identification	3.3
IEEE	Institute of electrical and electronic engineering	4.0
IP	Internet protocol	1.2
KSP	K-shortest path	1.10
MST	maximum spanning tree	1.10
MTP	Mesh token polling	4.1
OXC	Optical cross-connect	1.2
OADM	Optical add-drop-multiplexer	1.2
P-Cycle	Pre-configured Cycle	1.6
PS	Passive state	4.1
SONET	Synchronous optical network	1.2
STP	Star token polling	4.1
TP	Token polling	4.1
TSP	Tree and shortest path	2.6
TT	Token tree	4.0
URS	Upward root selection	3.0
WDM	Wavelength division multiplexing	1.1

Table of Notations and Symbols

Appearance		Section of First
$C(e_i)$	The capacity of node N_I's ith incidental link	3.1
$C(N_i)$	The average capacity of node N_I	3.1
C(T)	The total capacity of the candidate tree	4.1
D	The nodal degree	3.1
E	The set of the links	3.1
E_t	The set of links in a candidate tree	4.1
G(V, E)	A bi-directional network	3.1
M_I	A message used to build a temporary tree	3.2
M_V	A message used to transmit a root candidate information	3.2
T (V_t, E_t)	The candidate tree	4.1
V	The set of the vertices	3.1
V_t	The set of vertices in a candidate of the tree	4.1
λ	The number of optical wavelength	2.2

List of Tables

List of Figures

Chapter One Introduction

1.1 Development of the Optical Network

High-speed networking has gone through three generations of technologies [MuBi92]. In the first generation networks, copper-based or microwave-radio technology are used in the physical layer. So both data transmission and switching are in the electronic domain. In the second-generation networks, optical fibers replaced the copper links and the microwave-radio links. Here switching is still performed in the electronic domain, even though data transmission is in the optical domain. Due to the vast unused optical fiber bandwidth, the second-generation networks still have low bandwidth efficiency. In the third-generation networks, all the transmission and switching are expected to perform in the optical domain. Lots of new technologies have been developed with the third generation network, of which Wavelength Division Multiplexing (WDM) technology is the most important technology to enable the network to use a much larger fraction of the tremendous optical fiber bandwidth. [MuBi92][ElMo02].

1.2 DWDM Network

WDM technology can multiplex different wavelengths into one optical fiber and transmit them in one direction. Each wavelength can be demultiplexed into the original one when it arrives at the destination.

Early WDM technology combines 1310 nm and 1550 nm wavelengths into a single mode optical fiber [LeLe98]. As the development of the optical technology, the number of wavelengths that can be multiplexed into one fiber increases all the time, and it numbers 64 to 128 nowadays. This higher number of wavelengths leads WDM to the name Dense Wavelength Division Multiplexing (DWDM).

DWDM offers an efficient bandwidth-utilization and therefore improves the data transmission speed greatly. Nowadays the transmission speed of one fiber can reach 100 Gbit/s. However, the electronic switching at the network layer cannot catch up with the fast-increasing link capacity, therefore traffic bottlenecks would occur in switching networks. So the point-to-point data transmission is much more successful for the long haul network at the first stage of DWDM technology.

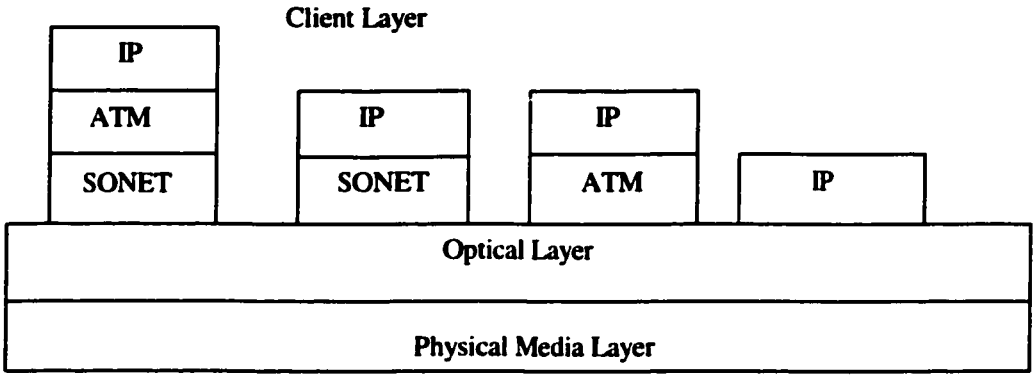


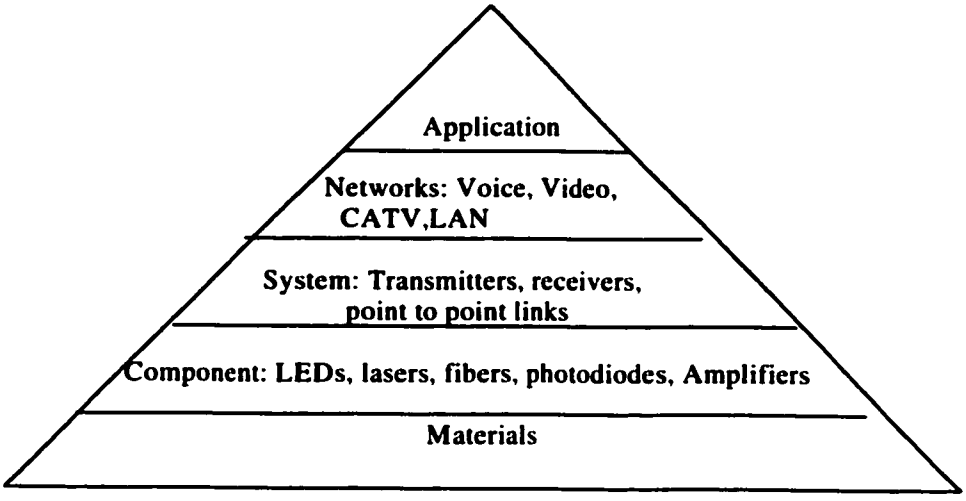
Fig. 1.1: Possible layers in a DWDM optical transport network [MuGu02]

As the new technologies such as optical cross-connect (OXC) [KaSt00] and optical add-drop-multiplexer (OADM)[KaSt00] become available, optical switching is more capable of solving the capacity problem and making the DWDM technology possible in all kinds of network topology. Different from point-to-point DWDM data transmission, the DWDM networking needs to have an optical DWDM protocol layer to take care of the setting-up and tearing-down of the light paths between two nodes; each light path has the same wavelength over each link.

From a protocol point of view, a DWDM network has three layers from low to high [MuGu02]: a physical media layer, an optical layer and a client layer. The optical layer consists of light paths and can provide protocol transparent circuit switching to the client

layer. A variety of clients can be supported in the same time, which means that different data formats, such as IP, SONET and ATM, can be transmitted together within one optical fiber. The optical DWDM layer can be the intermediate layer of the upper client layer and the lower physical media layer. When there is a failure, network can be recovered at the optical layer by rerouting the traffic on another light path. The possible layers in a DWDM optical network [MuGu02] are described in Fig.1.1.

According to the function, the optical layer can have its own three sub-layers: an optical channel layer, an optical multiplex section layer, and an optical transmission section layer. An optical channel layer can provide end-to-end networking of optical light paths for conveying the client data transparently. The optical multiplex section layer is responsible for aggregating optical signals of multi-wavelength. The transmission section layer takes care of the transmission of optical signals in different kinds of optical media such as the single mode and multimode fibers [ITUT872]. All the functions of these three layers can



guarantee that wavelengths can be set up and torn down at the optical layer.

Fig. 1.2: Optical communication pyramid

From a component point of view, an optical DWDM communication system has a pyramid structure [KaBe96] as depicted in Fig.1.2. Many new highly advanced new materials form the base of the optical telecommunication system. Opto-electronic devices are developed based on these new materials. The basic components for a DWDM system include LEDs, tunable lasers, fibers, photodiodes, flat gain optical amplifier such as Erbium-doped fiber amplifier (EDFA), wavelength converters, splitters, combiners, fast tuning receivers, OXCs and OADMs [GaMu99]. System designers use these components to design new subsystems and systems such as transmitters, transmission medias and receivers. Networking is another higher layer based on the system layer, and finally actual application layer are designed to utilize all the networks.

1.3 Physical Topology vs. Logical Topology

In a DWDM network, a physical topology is the layout of physical network facilities. The routers/switches/ADMs become the physical nodes of the network, and the optical fibers form the physical links. Because of the high traffic demand, a physical link can be made up of a bundle of fibers. Each fiber can be configured with a specific number of wavelengths. These multi-fiber configured networks are much more cost effective and bandwidth efficient than the single-fiber configured network.

Generally speaking, there are three major physical topologies: mesh, ring and tree. In a mesh topology, devices are connected with many arbitrary interconnections. Full meshes and partial meshes are two typical types of mesh networks. In a full-mesh topology, the nodes are connected with each other in the network. Although very expensive in implementation, they can redistribute the traffic to any other node in the network whenever there is a link or node failure. The average nodal degree of a partial mesh

topology is larger than two and smaller than the nodal degree of a full mesh. The network performance varies a lot with respect to the average nodal degree. Partial mesh networks yield less redundancy than full mesh topology and are less expensive to implement. One common network mesh topology nowadays is to use partial meshes as peripheral networks that are connected to a full meshed backbone.

A ring topology is a closed loop, and each node in the network is of degree two. The components for a ring topology are relatively simple and can span a large distance. Due to the sparse connections, ring topologies are relatively expensive. Recently ring topologies are widely used in the optical networks, because it can offer a higher bandwidth efficiency. SONET rings [WuTs92][KaSt00] are the typical ring structures nowadays for optical networks.

A physical tree topology is composed of connected links and nodes without cycles. A topology is a tree if and only if it is a connected graph with n nodes and n-1 links. Accordingly a tree topology has the minimum number of links to interconnect the whole network. Thus using tree topologies in an optical network will reduce the fiber expenditure. From an operational point of view, a tree topology offers many optimum data structures. Another advantage of tree topology is that a tree possesses a natural hierarchical structure, and each sub-unit of a tree topology is also a sub-tree. Therefore it is easy to expand an existing network and to construct a network that might span a wide area. This sub-tree property also makes it easy to isolate a failed node or a failed branch of the network. The disadvantage of a tree topology is that for a tree with a lower nodal degree, it is easier for a far end node to get disruptions in any other connections between the root and this node, because the traffic from this far end node to the other node on the other branch has to

traverse quite a few intermediate nodes. Also the demand for the capacity of the upper-level of a tree is much higher than that in the lower level.

A DWDM network can have a logical topology in addition to its physical topology. The light paths among node pairs in a DWDM network make up the optical layer, and the connectivity among light paths forms a virtual/logical topology of the network. Thus in a logical topology, a node is the routing node and a link is a light path in the network. Two directly connected nodes in a logical topology may not be connected directly in the physical network. To a DWDM network, a logical topology can be the same as the physical topology, but it is different in general. Fig. 2.1 in Chapter 2 shows a logical tree topology overlaid on a general mesh network.

1.4 DWDM Network Protection/Restoration

Each fiber in an optical DWDM network processes huge data because of the high bandwidth-efficiency. So a single fiber cut can bring a disaster to a high-speed optical network. Therefore very fast and efficient methods of restoration against cable cuts attract more and more attention [FaWE90][MaJC94].

The concept of network protection/restoration is to find an alternative route and redistribute the traffic whenever there is a link or node failure [YaSh02]. One of the most common failure modes for DWDM network is a single link failure that will affect all the traffic transmission through that link [LuMe00]. Path restoration and link restoration [DoWi94][FriT97][RaMu99a][RaMu99b][LuMe00] are the two main approaches to restore a single link failure. In a path restoration, all the traffic on the failed connection will be re-routed to a new established path from the source to the destination. In a link restoration, a new path is sought from one end node to the other end node of the failed link, and all the

unaffected links in an interrupted connection still remain fixed. Generally speaking, path restoration has a better restoration efficiency than link restoration, but it needs to know more network-connecting information. Link restoration has a better restoration time compared to path restoration and can be easy to be implemented in a distributed way.

Based on where the restoration algorithm is calculated and implemented, the restoration can be categorized into central restoration and distributed restoration [WuTs92]. A central controller is needed to be responsible for the path calculations in a centralized restoration algorithm. Thus the controller has to know the up-to-date information about the whole network. In the distributed restoration, each node involved in the failure will try to implement the restoration algorithm by itself.

Methods commonly employed for link restoration can be classified into preplanned and dynamic [LuMe00]. In the preplanned restoration, one route and wavelength has been chosen for the protection path and they cannot be changed while the dynamic restoration involves a search for a free path using spare capacity by broadcasting of help message [ChBi93], thus both the route and wavelength can be changed to satisfy a new demand. Preplanned methods often can be faster than the dynamic restoration, but cannot use the real time spare capacity as the dynamic restoration.

Network failures can be recovered either at the optical layer or at the client layers [MuGu02]. At the client layer, SONET, IP and ATM systems may employ their own failure recovery techniques. For example, SONET can perform a protection switch; An ATM system can reroute its virtual circuits; An IP system will run its routing protocol to seek out alternative routes. All these restoration techniques may result in deadlock and complications because of their interaction. So to recover the failure in the optical layer

becomes a popular issue currently. The advantages of realizing restoration in the optical layer lie in: 1) The number of recovered light paths in the optical layer will be much less than the number of components to be recovered in the client layer. 2) Performing restoration in the optical layer can be faster and more economical than that in the physical layer. 3) The optical layer can provide restoration in networks that currently do not have a protection scheme. 4) To implement the restoration in the optical layer can not only help restore service quickly but will also result in less traffic and control overhead.

1.5 Current Network Protection Schemes

To implement network protection/restoration in an optical DWDM network is not only to seek out a backup route through the logical topology, but also to guarantee that enough protection capacity is already available on that backup route for the affected connection. Seeking a protection path is in fact to overlay a logical topology on a mesh network. The most widely used logical topologies for network protection/restoration nowadays are ring, mesh and tree. Based on these topologies, several techniques have been developed to achieve a fast protection/restoration.

By itself, a single unidirectional or bi-directional ring does not protect signals against a single span or node failure. Self-healing rings are obtained by adding a second protection ring to the main working ring to get a duplicate diverse path for each single pair [MoGr98]. The 2-fiber Unidirectional Path Switched Ring (UPSR), Bi-directional Line Switched Ring (BLSR/2), and the 4-fiber Bi-directional Line Switched Ring (BLSR/4) are the three major self-healing ring structures that have been adopted as SONET standard [UPRS95][BLSR95]. The UPSRs are designed on the concept of dual-feeding or 1+1 protection while the BLSRs use looping-back mechanism to implement the working

wavelength protection. The ring protection schemes can offer the fastest restoration scheme [StGr99a], but because of the 100% redundancy, they are expensive in implementation.

Mesh-restorable networks have been widely studied as an alternative to ring-based self-healing networks [MuBi00]. Mesh restorable networks are designed on the concept that a much more generalized and efficient path-finding mechanism will restore failed signal units by multi-path rerouting through relatively much smaller allocations of spare capacity throughout the network. Topology, routing and capacity design are the major concern for mesh-restorable networks. Many algorithms have been proposed for solving the related problem. The KSP (K-successively shortest link-disjoint paths) [AnGr94][Grov89] in a multi-graph is connected to be an efficient one. This consists of a path set that is equivalent to finding the shortest alternate path, then the second shortest alternate path that is link disjoint with the first, and so on. In this thesis, we refer the KSP as the first shortest path from one end node to the other end node of a link. The main advantage of mesh-based restoration is the high capacity efficiency due to the spare capacity being shared by many different possible failure scenarios. So a mesh restorable network can be two to three times more capacity-efficient than ring-based networks in terms of the total spare capacity and the unused working capacity to assure 100% restorability when a failure happens [GeRa00]. However in order to obtain a 100% restoration, how to reroute working demands and correspondingly place spare capacity with a minimum total capacity is a comparatively practical but complicate problem to the mesh restorable network. Because of the route calculating and capacity distribution, mesh restorable network cannot guarantee a faster restoration compared to the self-healing rings.

Pre-configured cycles, called p-cycles [ElGe00][GrSt98], are introduced to use in network protection/restoration to achieve a self-healing ring restoration speed but a mesh restorable capacity efficiency. A p-cycle cannot only work as an unit-capacity bi-directional line-switched ring to protect the links on the cycle, but it also can protect the straddling links that have two end nodes on the cycle by providing two backup routes in clockwise and counter-clockwise directions [StGr00]. The disadvantage of p-cycle restoration includes: 1) It is hard for a cycle to protect a node failure and multiple link failures. 2) Usually the path length in a cycle increases with the network size and it is expensive to reroute the traffic with a long protection path. 3) It is not easy to implement distributed algorithm in the cycle-based protection schemes.

Tree topology has been studied as an alternative for network protection/restoration [MeFi99][FiMu97]. A tree uses the minimum number of links to interconnect a set of nodes and it is easy to find a route in the tree to cover each non-tree link. These properties make the tree topology suitable for network protection. The properties that a tree is a hierarchical structure and each sub-unit is also a sub-tree make it easy to implement distributed algorithms with tree topology. Conventional Spanning Tree (CST) is used for network protection in [StGr99]. This is implemented by the following procedures: Prim's algorithm [Brua92] is executed to find a maximum tree. A single copy of this tree is then configured as a pattern (of unit link capacity) in the network spares. The unallocated spare capacity pool and uncovered working links are updated and the revised configuration is the starting point for the next tree-forming iteration. Iteration continues until it is no longer possible to build a tree of more than one link in the unallocated spare capacity or

until no uncovered working links remain. In this thesis, we referred to the CST as using one spanning tree for network protection restoration.

1.6 Tree Classification

According to the structure, tree topology has different variations. Spanning Tree [NISTa] refers to a connected, acyclic subgraph containing all the vertices of a graph. A binary tree is a tree with at most two children for each node. Both the spanning tree and binary tree are used widely in switching network.

In a ring tree topology [WaLi99], each node is within a ring and these rings are interconnected via a tree-like topology. The ring tree can give an optimum number of wavelength converters and is recommended to use in an optical network.

A TreeNet [GeFr88] is a two-level architecture: The top level is a tree and the bottom level is a linear optical bus. The TreeNet topology is suitable for metropolitan area network.

Two Center Tree [Yang84] is constructed by connecting the root nodes of two binary trees. Two center tree has different variations: Binary Tree (BT), Quaternary tree(QT), X-tree (XT) and Ring tree (RT). Two center tree is thought to be efficient in solving the traffic bottleneck problem.

Hierarchical Tree [ShYa01] is a special type of spanning tree that each node has its parent and children. Tree ID is used to identify the position for each node in the tree. A hierarchical tree is suitable for network protection/restoration.

1.7 Spanning Tree Algorithms

There are two major approaches to implement a spanning tree algorithm. The central spanning tree algorithm needs a central controller to get all the information for the whole

network while the distributed algorithm does not need a central controller, but each node will implement the algorithm.

There are many central algorithms proposed to build a logical spanning tree topology overlain on a mesh network, of which the breadth-first search, the depth-first search, Prim's algorithm and Kruskal's algorithm are the most classic ones.

Gallager, Humlet and Spira gave the classic distributed MST algorithm with an O (2E+5NlogN) message complexity and an O (NlogN) time complexity [GaHu83]. The algorithm is asynchronous. At first, each node exchanges messages with its neighbors and selects an outgoing link with the least cost to connect its neighbors. These nodes form a fragment of a lower level. Fragments are combined together with a least-cost link and form a fragment of a higher level till a tree. There are about six kinds of messages used in this algorithm to build a spanning tree. There are quite a few papers after that giving improvement of this algorithm. Faloutsos and Molle give an optimal distributed MST algorithm [FaMa95] with message complexity O(E+NlogN) and time complexity O(N) under some constraints. These distributed spanning tree algorithms are widely thought efficient in broadcasting messages through a network.

In recent years, some tree algorithms have also been developed for preplanned recovery in network restoration [MeFi99][SaYa01][ShYa01]. A distributed hierarchical tree (HT) algorithm has been developed to construct a spanning tree with the link capacity arranged from a higher layer of the tree to the lower layer of the tree in a descending order to gain the maximum capacity usability [ShYa01]. At network start-up or after a topology change, the nodes in the network transmit tree ID labels to their neighbors. The label contains the information of the hierarchical tree structure and the capacity of the path from

this node to the root of the tree. In this algorithm, each node may be visited more than one time.

1.8 Motivations

Based on the literatures we have researched in, we feel that there are many issues that need to be investigated further. The few that are of interest here are discussed in the following:

Many existing protection algorithms are not scalable w.r.t. the size of a network. Since a tree network is scalable, we would like to investigate protection based on logical trees which presently do not appear to have much work done. Because a tree network creates a hierarchy, changes on one branch have limited impact on the other branches as well as the higher level of the hierarchy. When a node or a link is added to the network, the protection capacity assignment can be updated through local signaling and does not affect the nodes in the other parts of the network. So it is easy to maintain a tree network. As a hierarchical structure, it is easy to use "tree ID" [ShYa01] to identify the position for each node in the tree. This tree ID will help each node to reroute the traffic easily when there is a link or node failure. In addition, it is easy to implement a distributed tree algorithm that reduces the required computation power and enhances the real time response in the network. The nodes of the network do not need to be aware of the complete network topology and only some local information about their neighbors is sufficient for them to build the protection tree. In the same time, each link of an optical network may have its spare wavelengths (spare capacity), using the most spare wavelengths of the optical network may offer the largest restorability when there is a link failure. A maximum spanning tree can use the maximum spare wavelength of the network to connect all the

nodes without link redundancy. So building a maximum spanning tree in a distributed manner and using the spanning tree to protect a single link failure is a practical approach to study.

In a tree topology, the shape of the network is that of an inverted tree with the central root branching and sub-branching to the extremities of the network. In such a topology, the root needs to process all the messages between a source and a destination node which might reside in different branches. So the root of a tree needs to process more information than any other nodes in the network. To select a proper root is very important for the traffic to be distributed properly in the network. But so far distributed root selection algorithms are seldom studied.

Token has been used for medium access for a long time. The typical example is the Token Ring in IEEE 802.5 and FDDI [TaAn96], in which only the station who receives the token can transmit the packets. Using tokens to control the executing sequence and reduce the visiting times for each node is a new idea in the distributed spanning tree algorithm.

Traditional tree protection scheme uses tree links to protect non-tree links, and there is no way to protect tree links [StGr99]. To increase the restorability in a DWDM network, we need to find methods to protect the tree links. Since tree topologies come in many variations such as ring tree and two-center tree. We have an interest in variations of tree topologies, and this has motivated us to seek out some sub-structures that would facilitate the process of constructing a logical tree. A triangular tree topology that is defined in this thesis is thought to be suitable for network protection/restoration.

1.9 Objectives

The purpose of this thesis is to investigate the methods of implementing DWDM network protection/restoration with distributed MST algorithms. Specially we would like to work on the following:

1) Propose algorithms for selecting a root candidate in a distributed way for a MST. Evaluate and compare the performance of the heuristics.
2) Propose heuristics to build a MST in a distributed way. Evaluate and compare the performance of these heuristics.
3) Define a triangular tree topology. Propose algorithms to build such a triangular tree topology on a mesh network. Evaluate and compare the performance of the heuristics.

1.10 Approaches and Methodology

In order to achieve the goal of this thesis, we build a logical tree topology in a distributed manner on a physical topology. As discussed before, finding a path does not mean that the corresponding link can be protected, but all the spare capacity of the links in that path have to be equal or larger than the protected working wavelength. So we would like to use the spare capacity as the criteria to build a maximum spanning tree (MST). This MST is used to protect the non-tree links in the network. Meanwhile, we would like to seek a shortest path to protect the tree links.

Borrowed the idea from the roll-call-polling [RollPoll] and hub-call-polling [HubPoll], we use two token polling approaches to poll tokens to reduce the visiting time to each node in building a spanning tree. One is distributed from one single node to all the other nodes in the network and the other is to distribute from the current node to the other node in the network. Further comparison among ways of token distributions in the

distributed spanning tree algorithm will give us a better understanding of a token tree algorithm.

In building a distributed spanning tree, several kinds of messages are transmitted to exchange information between nodes. Different messages have different formats and different functions. The number of messages transmitted during the executing of the algorithm is one criterion to evaluate the performance of these algorithms. When analyzing the performance of the built MST algorithm, mean path length between each pair of the tree nodes and the ratio of the built tree capacity to the original network capacity can be used as the comparison standard.

The message complexity and time complexity as well as the property analysis provide us a better understanding of the algorithms. Restorability is an important factor to evaluate whether the algorithm is suitable for the network protection and restoration. Restorability among different protection scheme can offer a better understanding of the protection schemes. In calculating the restorability, we must address the capacity distribution problem. So in the thesis, we distribute the working capacity randomly and fix the ratio of the spare capacity to working capacity at different values in our investigation.

Our protection scheme in the token tree algorithm uses the tree links to protect the non-tree links, and seeks out the shortest path between the two nodes of a tree link in order to protect this tree link. For comparison, we also study the performance of two other protection schemes:1) KSP (k-Shortest Path) protection schemes [AnGr94], which finds the shortest path to protect each link in the network. 2) CST (Conventional Spanning Tree), which builds a spanning tree to protect non-tree links [DeSt99].

All the simulations are run on some topologies based on the existing networks. We shall conduct our investigations by simulation of our algorithm in Java.

1.11 Thesis Contributions and Organization

The contributions of the thesis are:

- Systematic study of the tree topology for various purposes. Giving the tree classification
- Propose different root selection algorithms to implement distributed root selections
- Propose a token tree algorithm using tokens to control the visiting time for each node when implementing a distributed spanning tree algorithm
- Propose a triangular tree topology and corresponding algorithms to build such a triangular tree
- Estimate the time complexity and message complexity for each algorithm
- Give the performance evaluation of each algorithm
- Combine mesh restorable network protection scheme and tree protection scheme, propose a new protection scheme

The rest of the thesis is organized as following:

- Chapter 2 gives network modeling, operations, definitions and assumption. All the modeling and assumption for composing the algorithm is given in this chapter.
- Chapter 3 gives three algorithms for root selection. Performance evaluation for these three algorithms is given in this chapter.
- Chapter 4 provides a token tree algorithm that proposes using two ways of token distribution to build a distributed spanning tree. Performance evaluation of the token tree algorithm, the performance comparison with a hierarchical tree algorithm

and the comparison between the two ways of token distribution are given in this chapter.

- Chapter 5 gives a triangular tree definition. Algorithms for building such a triangular tree are given in this chapter. The performance such as restorability, mean path length, the capacity ratio of the built triangular tree and some performance comparison are given in this chapter.
- Chapter 6 is the conclusion section.

1.12 Publication

Part of this thesis has approaches the following publications:

- Yuna Zhang and Oliver Yang, " Complexity Analysis of Hierarchical Tree Algorithm", 21st Biennial Symposium on Communications, Kingston, Canada, May, 2002, pp 46-50
- Yuna Zhang and Oliver W.W. Yang, "A Distributed Tree Algorithm for WDM Network Protection/Restoration" IEEE 5th International Conference on High Speed Networks and Multimedia Communications (HSMNC2002), July,2002, Jeju Island Korea, pp 289-294
- Yuna Zhang and Oliver W.W. Yang, "Different Implementations of Token Tree Algorithm fro DWDM Network Protection/Restoration", to be presented in IC3N2002, USA, Miami, Florida, Oct.2002

Chapter Two Network Modeling, Operations, Definitions and Assumptions

In this chapter, the modeling and assumptions of the DWDM network under study are provided along with the necessary operations and protocols.

2.1 Network Model

Our protection schemes are based on a multi-fiber configured mesh network that employs DWDM technology, which can be described in a graph G(V, E) in the view of the graph theory. Here V represents the physical nodes and E represents the physical links in the network. Also let I be the unique node ID of the ith node N_I in the network. This node ID can be used as the node address for the corresponding node in the network. Let $C(N_i)$ be the average capacity of node N_I that has the following equation $C(N_I) = \frac{\sum_{i=1}^{D} C(e_i)}{D}$, where $C(e_i)$ is N_I's ith incidental link capacity and D be the average nodal degree.

In the DWDM network studied in this thesis, a link is actually a physical span in which there are multiple fibers running parallel between two end nodes. Each fiber in such a link can be configured with any number of wavelengths. In general, a light path is unidirectional. So a traffic demand will be accomplished by a pair of bi-directional light paths on the same route. When transmitting messages, some wavelengths are used to carry the traffic and some are idle all the time. The ones carrying the traffic are the working wavelengths while the ones without traffic are the spare wavelengths. The number of working wavelengths and the number of spare wavelengths on each span are called working capacity and spare capacity respectively.

A node is a physical router. It can also be a switch or an OADM (Optical Add Drop Multiplexer). Each node has a unique node ID which can be used as the physical node address. The node will also have a unique tree ID after it is connected to a hierarchical spanning tree to identify its position in this spanning tree. The node ID will be different from the tree ID of the same node. Each node keeps a vector called "CurrentRoot" to keep the information of the selected root which will be introduced in Chapter 3.

Optical lambda switching makes the DWDM networks possible, and the optical switching fabric is the central part of an optical lambda switch. However, it is the existence of the routing control module that makes a plain optical switch into a wavelength router.

According to the different switches, DWDM networks can be classified as static DWDM networks and optical routing networks [Zhan02]. The static optical networks use plain switching that does not have intelligence in distributing wavelength while the wavelength routing optical networks can collect the wavelength information from all the nodes in the network and can re-distribute the wavelength when there is a network topology change. Our protection schemes are intended for the optical routing networks.

2.1 The Logical Tree Topology

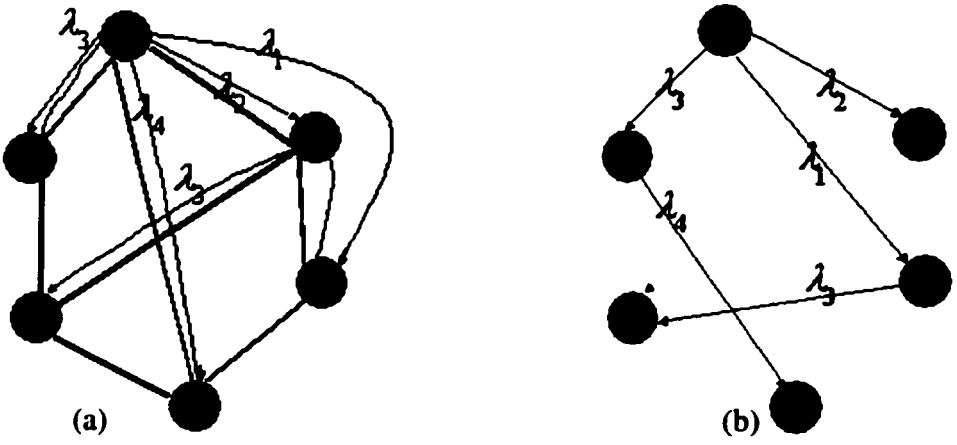


Fig. 2.1: A physical mesh network and a logical tree topology

We will focus our study on the physical mesh network as exampled in Fig.2.1(a). By distributing different wavelengths between node pairs, these light paths form a logical tree topology of the network as in Fig.2.1(b). The lines in a dark color are the physical links of the network while the directional lines are the light paths. Because the wavelength between nodes 1 and 5 is different from the wavelength between nodes 5 and 4, a wavelength converter needs to be placed in node 5 if information will be transmitted from node 1 to node 5. Node 2 also needs a wavelength converter if information will be transmitted from node 1 to node 6.

In order to setup and tear down the light paths, wavelength routing and distribution protocol has to be used to solve the wavelength assignment and routing problems. A wavelength distribution protocol is used as signaling protocol to set up (tear down) end-to-end light paths while a wavelength routing protocol is used to collect network-wide wavelength information and provide an efficient end-to-end wavelength path in seconds. These two protocols are the most important factors for the implementation of the DWDM network.

2.3 The Hierarchical Tree Protection Scheme

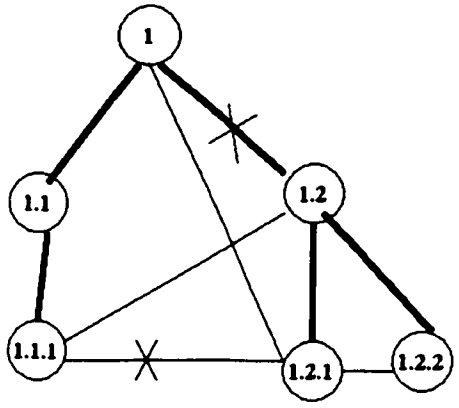


Fig. 2.2: A hierarchical tree structure

Fig.2.2 shows a hierarchical tree structure. A "tree ID" is used to identify the position of each node in the tree. The designation in each node represents the tree ID for each node. Tree ID "1" identifies this node is a root node at level 0 of the tree, and tree ID "1.2.2" means this node is at level 2 of the tree and is the second child of node 1.2. So after a hierarchical tree structure has been built, each node will have two different IDs: One is the node ID to identify the physical node address in the network. The other is the tree ID to identify the node position in the constructed tree.

A hierarchical tree is constructed from a root node. The root node has a tree ID "1". A node will transmit a tree ID to its children. A tree ID is formed by appending the sequence of this node among all its siblings to the end of the node's own tree ID. The sequence of the children is decided by the corresponding link capacity in a descending order. For example, the second child of the node with "tree ID" 1.3 will have a tree ID 1.3.2. Each node will have the same mechanism to build the hierarchical structure for its children. If a node selects one node as its parent, it just keeps the tree ID from this parent. It will delete all the tree IDs from the other non-parent nodes. Each node will keep the tree IDs it transmits to its children. If a node receives a message from its children to cancel the tree membership, it will delete the corresponding tree ID for this child. The hierarchical structure of the tree built in Chapter 3 and 4 Three will be formed according to this mechanism.

In the thesis, we propose three approaches of link restoration. The first one is using tree links to protect non-tree links which means seeking a shortest path in the tree. This can be done by consulting the tree ID table and finding the shortest path through the tree. For example, Fig. 2.2 considers the non-tree-link between the node with a tree ID 1.2.1 and the

adjacent node with a tree ID 1.1.1. If this link fails, the restoration path can be determined as (1.2.1)->(1.2)->(1)->(1.1)->(1.1.1) which consist of all tree links.

The second one is seeking a shortest path through the network. The algorithm used to calculate the shortest path can be the classic one such as Dijkstra's Algorithm [Tane96].

The third one is using two links of a triangle to protect the other link. Since each node will know whether it is included in a triangle. (Whether a node issues a higher priority to its children or whether a node receives a higher priority from its parent) (Chapter 5), so it is easy for the node to calculate the restoration path by their Tree IDs. Such as nodes with tree IDs 1.2, 1.2.1 and 1.2.2 will be easy to calculate their restoration paths between each other.

2.4 Assumptions

In this part, we will give all the assumptions we need in this thesis.

1) Since it is difficult to have a real network traffic demand distribution, we assume the traffic demand on each link is randomly and independently distributed, which is also the working capacity between each node pair. In the simulations of this thesis, we randomly distribute the working capacity between 3 and 8. The distribution of the spare capacity among all links will decide the shape of the tree, which does not change with the ratio of the working capacity to the spare capacity. So we set this ratio as 1 in building the spanning tree. But the restorability changes with the ratio of the working capacity to the spare capacity. We fix this ratio as 0.5, 1, 1.5 and 2 of each link at different simulations when we calculate the restorability.

2) To guarantee that the network can operate properly, we assume that both wavelength distribution protocol and wavelength routing protocol can gather all the required network

topology information. Each node in the network studied in this thesis can know the topology information of its connected links.

3) We assume that the wavelength routing switches that make the DWDM network possible in the optical network are available wherever needed in the network. Also wavelength conversion and ADMs are enough to guarantee that the optical network operate properly.

4) In the following discussion, the tree routing and shortest path routing can always be found and the wavelength can be always guaranteed when we need to transmit a message in building the spanning tree.

2.5 Message Transmission

Since the algorithms discussed in this thesis are distributed, they can be accomplished by message-exchanges between nodes. There are several messages are needed in the following algorithms. All these messages are identified by their message ID and we assume that there are no messages lost during the executing of the algorithms. The details of the messages will be introduced in the following chapters.

2.6 Definitions

The followings are the definitions of some common terminologies used through out the thesis:

1) A spanning tree is defined as a collection of network nodes and links that provides connectivity between any pair of nodes in the network without forming any loop.
2) Tree members refer to all the nodes and the links that are connected to the spanning tree. Note that if a node has a neighbor node that is a non-tree member, then the link connecting them is also a non-tree member.

3) A tree link refers to the link that is selected by the algorithm to be part of the constructed spanning tree.

4) A non-tree link refers to the link that is not part of the constructed spanning tree.

The followings are the definitions of some common performance measures used through out this thesis:

5) Restorability in this thesis is computed as the protected working wavelengths divided by the total number of the working wavelengths in the whole network.

$$restorability = \sum_{i=1}^{E} \sum_{j=1}^{W} X_{ij} / (EW)$$

where E is the number of links in the network. W is the number of working wavelengths on each link. $X_{ij} = 1$ if the working wavelength can be protected. $X_{ij} = 0$ if the working wavelength cannot be protected.

6) Average Hop Count of Shortest Path refers to the mean path length in number of hops among all pairs of nodes in the constructed tree.

7) Capacity Ratio refers to the ratio of the total capacity of the constructed tree to the total capacity of the network. The capacity of a link refers to the spare wavelengths of the link.

8) Number of Visiting Time refers to the total number of occurrences the algorithm has been recalled in all the nodes of the network to construct the spanning tree.

2.7 Simulated Networks

We have run our algorithms in some models based on the existing networks. These networks include the USA Long Haul Network with 28 nodes, the France Telecom network with 44 nodes, the Japan Network with 56 nodes and the MCI network with 41 nodes. The work capacities are randomly chosen between 3 and 8, and the spare capacities

are set according to the ratio between the spare capacity and the working capacity, which ranges from 0.5, 1.0, 1.5 to 2.0. All the simulations are run in Java on a Pentium Cyrix 120MHz computer using the Windows NT System. Each simulation run consists of about hundreds of packets, and takes about 5 to 20 minutes to run. Our Simulation results have been tested for convergence as demonstrated by their 95% confidence intervals. Some examples are tabulated in appropriate place in this thesis.

Chapter Three Root Selection Algorithms

Any node in the network can be the root of a logical tree. Therefore there must be a criterion for its choice. Since a root in a hierarchical tree structure must be able to process more potential traffic re-routing (e.g. in case of a link failure) than other nodes in the network, so we would like to select a root with the largest average link capacity in order to restore the traffic properly. Such root selection algorithms can be run centrally at a node or can be done in a distributed manner.

In this chapter, we propose three distributed root selection algorithms. The Upward Root Selection (URS) algorithm first builds a temporary breadth-first-search tree, and then selects a root candidate with the largest average capacity. The Downward Root Selection (DRS) algorithm implements the selection while building the temporary tree. The Distributed Root Selection (DiRS) algorithm transmits messages between neighbors without building a temporary tree.

Performance analysis is presented in this chapter to compare these three algorithms.

3.1 Problem Statement

The goal of our algorithms is to select a node with the largest average capacity of its incidental links for a logical tree topology. So let V be the set of vertices and E the set of links in the graph G(V,E). Let i be the unique node ID of the i^{th} node N_i in the network. Let $C(N_i)$ be the average capacity of node N_I that has the following equation $C(N_I) = \frac{\sum_{i=1}^{D} C(e_i)}{D}$, where $C(e_i)$ is N_I's ith incidental link and D be the nodal degree. Then our problem can be posted as following:

Input: A physical bi-directionally connected mesh network G = (V, E).

Output: A node N_r with the maximum average capacity (In the case of a tie, a node N_r with a smaller node ID).

3.2 Implementation

To implement the root selection, each node will keep a vector called "CurrentRoot" to keep the information of the selected root. The "CurrentRoot" information includes two parameters, node capacity and node ID, which are set with its own capacity and node ID at the start-up state of the algorithm. To know the information of its connected neighbors, each node has to exchange messages with its neighbors.

The following are the common features of the URS (Section 3.2.1) and the DRS (Section 3.2.2) algorithms. Both are centralized algorithms as opposed to the distributed version (Section 3.2.3).

The URS algorithm and the DRS algorithm begin and end with one node, called Node BEGIN; and in between, each node only needs to exchange information with its neighbors. Node BEGIN can be chosen randomly from among all nodes. (e.g. It can also be selected with the smallest average distance in hops to each other node to cut down the time to run this algorithm). Both root selection algorithms consist of two steps: a propagation of messages from node BEGIN towards the leaf nodes, and a reverse propagation of the collected information from the leaf nodes back to node BEGIN.

In the first step, the DRS and URS algorithms build a breadth-first spanning tree rooted at node BEGIN. This spanning tree is called a "temporary tree" because it is only used to provide a path from node BEGIN to every other node of the network and is not required after the root is selected.

Message ID(0)	Source Address	Destination Address	Node BEGIN Address

Fig. 3.1: An M_1 message

Two message types are used in the URS algorithm. Message M_I (Fig.3.1) is used to build a temporary tree. The whole selection process will be implemented through this tree later. Here "Message ID" is used to identify the message (Here we designate 0 as the message ID for the M_I message). "Source Address" is the current node address and "Destination Address" is the corresponding neighbor address. "Node BEGIN Address" is the physical address of node BEGIN.

Message ID(1)	Source Address	Destination Address	CurrentRoot	Connecting Information

Fig. 3.2: An M_V message for the URS algorithm

The second message type in the URS is M_V which is used for propagating the information of the selected root candidate from the current node to the other nodes. The field "CurrentRoot" is used to carry the information of the root candidate. The field "Connecting Information" is used to transmit the connecting information (e.g. its incidental links and the corresponding neighbors) of the current node to its temporary parent. This will become apparent in section 3.2.1. We assume that the network can have its own mechanism to expand the connecting information to message M_V.

Message ID(2)	Source Address	Destination Address	CurrentRoot

Fig. 3.3: An M_V message for the DRS

There is only M_V message (Fig.3.3) to be used in the DRS algorithm. As obviously distinguished in Fig.3.2 and 3.3, the M_V message for the DRS algorithm does not have the "Connecting Information". But this message can accomplish the function of temporary-tree building and root selection.

3.2.1 The Upward Root Selection (URS) Algorithm

The URS algorithm is accomplished in two steps. The first step is to transmit a simple message M_I to build a temporary tree. This step is accomplished as follows: Node BEGIN transmits a M_I message to all its neighbors. Upon receiving M_I, if a neighbor node X is not a member of the temporary tree yet, it will join the tree by marking itself as a temporary tree member. Besides identifying the node where M_I comes from as a temporary parent, this node X also transmits a copy of M_I to its unmarked neighbors. This process will be repeated until a terminal node, called the leaf node is reached when there are no more unmarked neighbors to send (obviously, all leaf nodes have a node degree of one in the logical tree). Message M_I will be discarded at a leaf node.

The second step is to transmit a message M_V (Fig.3.2) from each leaf node back to node BEGIN in order for node BEGIN to select a root candidate with the largest capacity. Message M_V has been updated with the largest capacity of the nodes it passes through till it reaches node BEGIN. This step is implemented as follows.

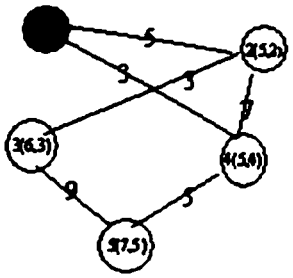
After receiving a message M_I, a leaf node will discard the M_I message and transmit a M_V message (see Fig.3.2) to its temporary parent. Each non-leaf node, after receiving all the M_V messages from its temporary children, will select the "CurrentRoot" containing the largest capacity. In case of a tie, a node with a smaller node ID can be chosen. Then the node will use this information to update its own vector "CurrentRoot". It then prepares its message M_V with the updated "CurrentRoot" information, and in the same time it will expand its own connecting information and all the "Connecting Information" from its temporary children to the "Connecting Information" field of Message M_V. Message M_V is transmitted to its temporary parent.

After receiving all M_V messages from its temporary children, node BEGIN can now have the connecting information of the whole network and can select a root node with the largest capacity. After node BEGIN finishes selecting, it will broadcast the root information all over the entire network. Fig.3.3 shows the pseudo-code of the URS algorithm described above.

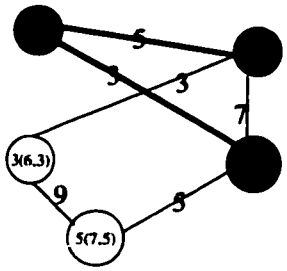
```
Step 0:Initialization of each node:
  Set the capacity of the CurrentRoot with its own capacity;
  Set the node ID of the CurrentRoot' with its own node ID;
Step 1: Node BEGIN starts to transmit a message M_I:
  Record all the unmarked neighbors as its children;
  Send a message M_I to its children;
  Mark itself as a temporary tree member;
Step 2: Process the messages:
  For each node N_i:
    While (Not receiving message M_V from all its children){
      Receive Messages;
      Switch (received message){
        Case (M_I ):{
            Record the node where the message came from as its parent;
            If (Node N_i is a leaf node){
              Delete M_I;
              Initialize a message M_v with its "CurrentRoot" and its own connecting Information;
              Send M_v to its parent;
            }Else{
              Record all the unmarked neighbors as children;
              Send message M_I to all its unmarked neighbors;
              }
        }
        Case (M_V ):{
        Select a root candidate from the current node and all the received M_V with the
          largest capacity of the "CurrenRoot" or in case of a tie, select the one with a
          smaller node ID;
        Update "CurrentRoot" of node N_i ;
        If (Node N_i is node BEGIN )
          GOTO Step3
        Else {
        Update M_v with CurrentRoot information of Node N_i ;
        Expand "Connecting Information" of node N_i and all "Connecting Information"
          from the received M_V;
        }
        } Send a Message M_V to its parent;
      }
    }
Step3: Broadcast the message to the whole network
```

Fig. 3.4: Pseudo code of the URS algorithm

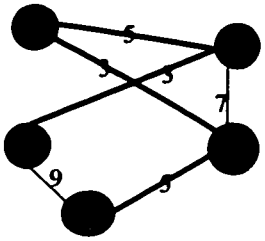
3.2.1.1 An Example of the URS Algorithm



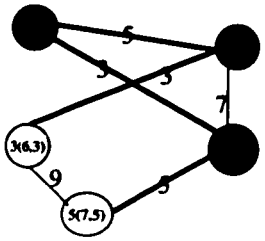
(a) Node 1 is Node BEGIN marked as a temporary tree member. Each node keeps a "CurrentRoot" vector initialized with its own capacity and its own ID.



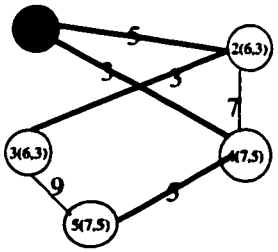
(b) Node BEGIN transmits a M_t to node 2 and node 4



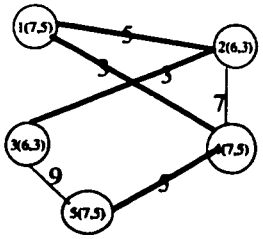
(c)Node 2 transmits a M_t to node 3. Node 4 transmits a M_t to node 5



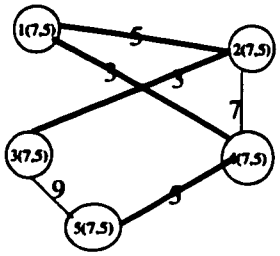
(d) Node 3 and node 5 become leaf nodes. Node 3 transmits M_v back to node 2 and node 5 transmits M_v back to node 4



(e) Node 2 and node 4 update its "CurrentRoot" and transmits M_v back to node 1



(f) Node 1 updates its "CurrentRoot" and selects the root



(g) The root information is broadcasted all over the network

Fig. 3.5: An example of the URS algorithm

Fig. 3.5 illustrates how the root is selected in the URS algorithm. The designation x(y,z) in a node means that node x keeps a vector "CurrentRoot" with capacity y and node ID z. The number on each link is the spare capacity of the corresponding link. In Figures 3.5a to 3.5c, the nodes in dark color are the ones already connected to the temporary tree. The nodes without color are the ones that have not received a message M_I and are not a temporary tree member yet. In Fig. 3.5d to 3.5g, the nodes in dark color are the ones connected to the temporary tree but have not received message M_V yet while the nodes in a light color are the ones that have received message M_V and have selected a root candidate.

Following diagram sequence in Fig.3.5, Diagram (a) shows that node 1 is node BEGIN and each node in the network keeps a vector "CurrentRoot" initialized to its own capacity and its own ID. In Diagram (b), node BEGIN (Node 1) transmits a message M_I to nodes 2 and 4. In Diagram(c), node 2 transmits a message M_I to node 3, node 4 transmits a message M_I to node 5. In Diagram (d), nodes 3 and 5 become leaf nodes, node 3 transmits a message M_V back to node 2, node 5 transmits a message M_V back to node 4. In Diagram (e), nodes 2 and 4 update its "CurrentRoot" and transmit a message M_V back to node 1. In Diagram (f), node 1 updates "CurrentRoot" and selects the root. In Diagram (g), the root information is broadcast all over the entire network.

3.2.2 The Downward Root Selection (DRS) Algorithm

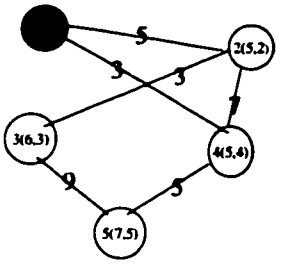
In the DRS algorithm, only message M_V (Fig.3.3) is used to select the root. Upon receiving a message M_V (Step 2), if node X is a leaf node, node X will mark itself as a temporary tree member. It will select the "CurrentRoot" with the larger capacity from the incoming message M_V and node X itself, and will use the selected "CurrentRoot" information to update the unselected one. The message M_V will be transmitted directly back to node

BEGIN. If node X is a non-leaf node, then node X marks itself as a temporary tree member and selects the "CurrentRoot" with the larger capacity from the incoming message M_V and node X itself. Node X uses the selected "CurrentRoot" information to update the unselected one. Node X then transmits a message M_V to all its unmarked neighbors. This procedure is repeated until all the leaf nodes transmit a message M_V back to node BEGIN. Finally (Step 3), node BEGIN will be responsible for the selection of the root candidate and propagate the information to all its neighbors. Figure 3.6 shows the pseudo code of the DRS algorithm

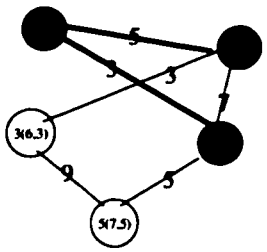
```
Step 0: Initialization of each node:
        Set the capacity of the "CurrentRoot" with its own capacity;
        Set the ID of the "CurrentRoot" with its own Node ID;
Step 1: Node BEGIN starts to transmit a message M_V:
        Record all the unmarked neighbors as its children;
        Send a Message M_V to its all children;
        Mark itself as a temporary tree member;
Step 2:For each node N_I:
        If (unmarked node)
        {
                Receive M_V from marked neighbors;
                Mark current node as a temporary tree member;
                Select the "CurrentRoot" with the larger capacity from the incoming
                  message M_V and node N_I itself, or in case of a tie, select the
                   "CurrentRoot" with a smaller node ID;
                Update the unselected one with the selected "CurrentRoot";
                If all the neighbors are marked nodes,
                        Send a M_V back to Node BEGIN;
                Else transmits a new M_V to all its unmarked neighbors
        }
Step3:  Node BEGIN selects a root from all the leaf nodes, then broadcasting all over the network
```

Fig. 3.6: Pseudo code of the DRS algorithm

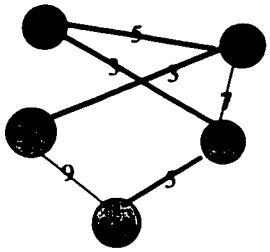
3.2.2.1 An Example of the DRS Algorithm



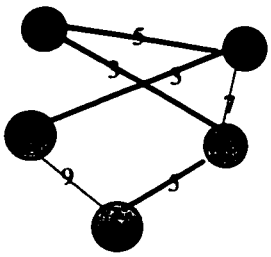
(a) Node 1 is Node BEGIN and mark its self as a temporary tree member. Each node keeps a "CurrentRoot" vector initialized with its own capacity and its own ID.



(b) Node BEGIN transmits a M_V to node 2 and node 4. Node 2 and node 4 update their "CurrentRoot" and mark themselves as temporary tree members



(c) Node 2 transmits a M_V to node 3, node 4 transmits a M_V to node 5. Node 3 and node 5 update their "CurrentRoot" and mark themselves as temporary tree members;



(d) Node 3 and node 5 transmit a M_V to node 1 respectively. Node 1 selects the root, update its "CurrentRoot" and broadcasting it all over the network

Fig. 3.7: An example of the DRS algorithm

Figure 3.7 is an example of the DRS algorithm. We will keep the same notations and color designations as in Fig 3.5 of Section 3.2.1.1. In Diagram (a), node 1 is node BEGIN. Each node keeps a "CurrentRoot" vector initialized with its own capacity and ID. Node 1 marks itself as a temporary tree member. Diagram (b) shows that node BEGIN transmits a message M_V respectively to nodes 2 and 4. Nodes 2 and 4 update their "CurrentRoot" and mark themselves as temporary tree members. In Diagram (c), node 2 transmits M_V to node 3, and node 4 transmits M_V to node 5 . Nodes 3 and 5 update their "CurrentRoot" and mark themselves as temporary tree members. Diagram (d) shows that nodes 3 and 5 transmit

M_{VS} back to node 1. Node 1 updates its "CurrentRoot" and broadcasts the root candidate all over the network.

3.2.3 The Distributed Root Selection (DiRS) Algorithm

The DiRS algorithm is quite simple in comparison with the other two centralized algorithms. Like the DRS algorithm, only messages M_V are used to complete the root selection. At first, each node creates a message M_V with its own "CurrentRoot" information and exchanges this message with its neighbors. As long as the capacity in the "CurrenRoot" of the incoming message M_V is larger than that of its own "CurrentRoot", (or in the case of equal capacity, we choose the one with a smaller node ID), it will update its own "CurrentRoot" with that of the incoming message M_V. Then M_V is transmitted to all its neighbors except the one where the message comes from. The algorithm finishes when there is no more message transmission in the network. The node whose node ID is the same as the "CurrentRoot" field, is the selected root. Fig.3.8 shows the pseudo code of the DiRS algorithm.

```
For each node N_i;
Step0:Initialization of each node:
        Set CurrentRoot's Capacity as its own capacity;
        Set CurrentRoot's ID as its own Node ID;
        Initializing a M_V with its CurrentRoot Information;
Step 1:Send a M_V message to all its neighbors;.
Step2:
        For each node N_i, do while there is still message transmission {
        Receive M_V from all its neighbors;
        If the capacity of the CurrentRoot in the incoming message M_V is larger than that of node
        N_i , or in a case of a die, the node ID is smaller;{
                Update the CurrentRoot with that of the coming M_V;
                Send a M_V message to all its neighbors;
                }
        Else delete M_V ;
Step3: The root is selected when there is no more message transmission.The one
        whose CurrentRoot.ID is the same as its own ID is the root.
```

Fig. 3.8: Pseudo Code of the DiRS Algorithm

3.2.3.1 An Example of the DiRS Algorithm

Node Id	Step 1 Current Root		Step 2 Current Root		Step 2 Current Root	
	C	ID	C	ID	C	ID
1	4	1	9	2	9	2
2	9	2	9	2	9	2
3	9	3	9	2	9	2
4	1	4	9	2	9	2
5	6	5	9	3	9	2

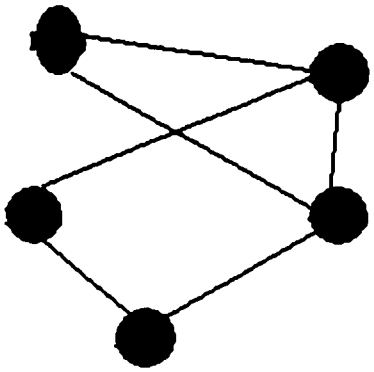


Fig. 3.9: An example of the DiRS algorithm

Instead of using similar diagrams as those in Fig.3.5, we illustrate the selection steps in one table in Fig.3.9. In the table, the essential contents of the "CurrentRoot" field are shown. Here "C" stands for the capacity while "ID" represents the node ID. The table shows that in the initial state, each node keeps "CurrentRoot" which is set with its own capacity and node ID. After the first round of message exchanging, each node selects a root candidate from among all its neighbors. In the next round of message exchanging, the root is selected which is node 2.

3.2.4 The Central Root Selection Algorithm

For comparison purpose later on, we have also included the Central Root Selection (CRS) algorithm. The CRS is a quite simple algorithm as following: Average capacity of each node is obtained by dividing the total capacity of its incidental links by the number of the incidental links. The node with the largest average capacity is selected as the root.

3.3 Performance Analysis and Evaluation

It would be of interest to always evaluate and compare the performance of these root selection algorithms as following.

3.3.1 Time Complexity

The upward and downward root selection algorithms are in fact breadth-first search algorithms except that each node may be visited twice, because both of the algorithms build a temporary tree and transmit messages M_V back from the leaf nodes to node BEGIN through the temporary tree. Since we know the complexity of the breadth-first search is O(N+E) [GoTa00], so the complexity for these two algorithms is also O(N+E). The worst case of E is N^2 in a full mesh network and the overall worst case time complexity is therefore O(N^2).

In the DiRS algorithm, each node has to keep on transmitting message M_V to its neighbors if the capacity of the "CurrenRoot" of the incoming message M_V is larger than that of its own. The worst-case complexity happens when the topology is a line and all the other nodes have a larger capacity than that of the current node. In this case the node will receives N-1 messages and transmit a copy of them to its neighbors. So the worse case complexity is O(N^2).

3.3.2 Message Complexity

When selecting the root, in the upward and downward root selection algorithms, there is one message transmitted on each link down-stream from node BEGIN to each leaf node in building the temporary tree. The message complexity for this part is E (the number of links in the network). When M_V is transmitted back from a leaf node to node BEGIN, there can be at most one M_V transmitted on each link of the temporary tree. So the message

complexity for this part is N-1. The total message complexity for the root selection algorithm is therefore E+N-1, which is O(E+N).

In the distributed root selection algorithm, the largest message number a node can receive is N-1. There are N nodes in the network, so the worst case message complexity is $O(N^2)$.

3.3.3 Performance Evaluation

In order to provide a better understanding of the algorithms, we have run these algorithms in some existing networks as mentioned in Chapter 2, and compared their performance against the CRS algorithm (Section 3.2.4).

Table 3.1: Simulation result of the CRS algorithm

Item	US28	FranceTelcom	Japan	MCI
Selected Root ID	6	36	25	14

Table 3.2: Simulation result of the URS algorithm

Item	US28	FranceTelcom	Japan	MCI
Selected Root ID	6	36	25	14
Message M_I Transmitted	63	97	111	80
Message M_V Transmitted	27	43	55	40

Table 3.3: Simulation Result of the DRS algorithm

Item	US28	FranceTelcom	Japan	MCI
Selected Root ID	6	36	25	14
Message M_V Transmitted (downward)	63	97	111	80
Message M_V Transmitted (Upward)	12	29	46	24

Table 3.4: Simulation Result of the DiRS algorithm

Item	US28	FranceTelcom	Japan	MCI
Selected Root ID	6	36	25	14
Message Transmitted	72	218	223	51

We can see from the simulation results that each of the four algorithms demonstrates the same result (root node) as the Central Root Selection Algorithm in these four networks. This confirms that the three algorithms get the correct result. But messages transmitted by the three algorithms are different. DiRS algorithm uses the fewest messages in the US Long Haul Network and the MCI Network, but uses the most messages in the other two networks. URS algorithm and DRS algorithm use the same number of M_l messages, but URS algorithm uses more Mv messages than the DRS algorithm.

3.4 Concluding Remark

In this chapter, we proposed three root selection algorithms that are used to select a proper root candidate for a logical tree topology. The simulation result shows that different ways to implement the root selection obtained the same root candidate. All these three algorithms are implemented by message transmission and the simulation results show that these three algorithms use different message numbers during the execution of the root selection.

The three algorithms also have their own characteristics. Since the node BEGIN in the URS algorithm can get the topology information of the whole network after finishing the execution of the algorithm, this algorithm is suitable for the future work that needs a node to maintain all the information of the whole network. The DRS algorithm may need fewer message transmissions, but in a real network it may be difficult for the node BEGIN to know how many M_V messages it should receive. The DiRS algorithm is easy to

implement and there is no need for a node to be responsible for the whole root selection process, so it is suitable for a completely distributed algorithm.

With a root selected by one of the algorithm, one can go ahead to build a logical tree for network protection. This will be detailed in next chapters.

Chapter Four Token Tree Algorithms

This chapter concerns the construction of a logical tree for network protection after a root has been selected in the last chapter. We present a token tree algorithm here. These algorithms have the features of reducing the visiting times for each node in the spanning tree (which is a critical issue of restoring a network in the shortest time after a failure). They also make use of the tree links to protect the non-tree links and use the shortest path to protect the tree links. Towards the end of this chapter, we evaluate and compare their performance against some known protection schemes.

4.1 Problem Statement

Consider the general DWDM network depicted in Ch.2, that can be represented by a general graph G(V.E), where V is the set of vertices and E is the set of links. Let V_t be the set of vertices, and E_t be the set of links in a candidate tree T (V_t, E_t). We are interested in overlaying this tree logically over the physical network represented by G. In other words, we want $T \subseteq G$, $V_t \subseteq V$ and $E_t \subseteq E$. Let the total capacity of the candidate tree be $C(T) = \sum_{i=1}^{N} C(e_i)$ where $C(e_i)$ is the capacity for the i-th link, and N be the number of the nodes in the tree. Then our problem of constructing a spanning tree can be posted as follows:

Input: A physical bi-directional connected mesh network G = (V, E).

Output: A root node N_R with the largest average capacity and a spanning tree $T=(V_t,E_t)$ with the largest capacity C(T).

The problem of deciding the root node N_R has been discussed in the last chapter. We shall focus below on building the tree itself.

4.2 Implementation of the Algorithms

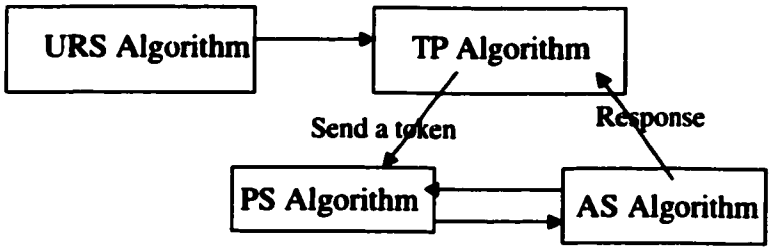


Fig. 4.1: Relationship among the sub-algorithms

Our Token Tree (TT) Algorithm consists of three sub-algorithms: Token Tree (TT) Algorithm, Passive State (PS) Algorithm and Active State (AS) Algorithm. The relationship among the three sub-algorithms is illustrated in Fig. 4.1. The TP algorithm sends a token to each node according to a breadth-first search sequence. Most of the time, a node is in its passive state and executes the PS algorithm. After a node receives a token, it will enter its active state and execute the AS Algorithm to form a logical spanning tree. After finishing the AS algorithm, a node goes into its passive state and executes the PS algorithm again. Note that each time only one token is distributed in the network, and only one node is in the active state.

For completion, we shall include a root selection algorithm. For the TT algorithm, we have to use the URS algorithm to select the root, because this algorithm uses the M_V message that contains a field called "Connecting information" in the message M_V. After all the M_V messages are transmitted back from the leaf nodes, the node BEGIN can collect the network connecting information and use it to send the Star-Token that will be introduced in Section 4.2.3.1. The details of the URS algorithm can be found in Section 3.2.1

4.2.1 The TP Algorithms

The spanning-tree construction begins after the root is selected. For comparison, we come up with two algorithms to implement the token polling. In the Star Token Polling (STP) Algorithm, one node (Node BEGIN) sends the token to all the other nodes. In the Mesh Token Polling (MTP) algorithm, the current active node sends the token to the next active node. The ideas of the STP and the MTP are similar to the roll-call and hub-call polling methods [Schw88] except for their implementations which we shall detail below.

4.2.1.1 The STP Algorithm

Message ID (3)	Source Address	Destination Address

Fig. 4.2: Format of Star-Token message

In this algorithm, a message called the Star-Token (Fig.4.2), is distributed from one node to all the other nodes in the network. Already having known the network information after the root selection process, node BEGIN can send the token in a breadth-first search sequence. The first token is always sent to the root node. Upon receiving a token, a node will begin to execute the AP algorithm (in Section 4.2.1.3). After completing the AP algorithm, the node will transmit a response message called Star-Response back to node BEGIN. Then node BEGIN will transmit another Star-Token to the next node. This procedure is repeated until all nodes receive a token and reply. Here "Message ID (3)", identifies the message type as a "Star-Token". "Source Address" is the address of Node BEGIN, and "Destination Address" is the node address of the corresponding node where the token will be sent. A Star-Response message can be as simple as a Star-Token message, but with the source and

destination addresses exchanged. It has a message ID of 4. The STP algorithm is executed in node BEGIN and can be described as the pseudo code in Fig 4.3:

For each node N_i in the network

```
While there are still nodes in the network that have not received a token
{
        Prepare a Star-Token and send it to node N_i according to the breadth-first search
          sequence;
        Wait until the Star-Response comes back;
}
```

Fig. 4.3: Pseudo code of the STP algorithm

4.2.1.2 The MTP Algorithm

Message ID (5)	Source Address	Destination Address	Node ID List

Fig. 4.4: Format of Mesh-Token Message

In this algorithm, a message called the Mesh-Token (Fig.4.4) is passed from the current active node to the next active node. Here "Message ID (5)" identifies the message type as "Mesh-Token". "Source Address" is the address of the current active node (in active state), and "Destination Address" is the node address of the next active node. The "Node ID List" is used to hold the sequence for which the token should be passed and therefore the sequence for which the nodes will be connected to the tree. With the help of the pseudo code in Fig. 4.5, we can now describe the MTP algorithm as follows.

In the start-up state of the algorithm, the root node prepares a Mesh-Token message in the format of Fig.4.4. It adds its own ID and all its neighbor's IDs to the Node ID List. The sequence to add the neighbors' IDs is in a descending order of link capacity. A pointer is used to indicate which node this message should be passed to next. This pointer is set with the root ID. Upon receiving this mesh-token message, a node becomes an active node and it will add the IDs of its neighboring nodes (nodes that are not yet included in the node

ID list) to the tail of the list, then move the pointer to the next node ID in the node ID list. After an active node finishes the AP algorithm, it will transmit the token to the node currently indicated by the pointer. When the pointer comes to NULL (i.e. having exhausted the list), the token message will be discarded, and the TT algorithm ends.

```
If the current active node is a root node
{
        Put the root node ID into the Node ID List of the Mesh-Token;
        Add the neighbors' IDs to the tail of the Node ID List according to the descending
          order of the link capacity;
        Move the pointer to the next node in the Node ID List;
        Wait until AP Algorithm finishes;
        Send the token to the node pointed by the pointer of the Node ID List;
}
Else if the pointer of the Node ID List is  not NULL
{
        Add the neighbors' IDs that are not included in the Node ID List to the tail of the
          Node ID List according to the descending order of the link capacity;
        Move the pointer to the next node in the Node ID List;
        Wait until AP Algorithm finishes;
        Send the token to the node pointed by the pointer of the Node ID List;

}Else Exit.
```

Fig. 4.5: Pseudo Code of the Mesh-Token Polling

4.2.2 The PS algorithm and the AS algorithm

Message ID (6)	Source Address	Destination Address	Tree ID

Fig. 4.6: Format of mark-tree-link message

A mark-tree-link message (Fig.4.6) is transmitted by the current active node N_i to its non-tree-member neighbor N_j to inform N_j that the corresponding link $L(N_i, N_j)$ has been marked as a tree member (Chapter 2). This message is used in both the PS and the AS algorithms. Here "Message ID (6)" identifies the message type as "mark-tree-link". "Source Address" is the address of the current active node, and "Destination Address" is the node address of the corresponding neighbor. "Tree ID" (Chapter 2) is used to identify

the node position in the constructed tree. The life span of the mark-tree-link message is one hop.

A Release-Tree-Link message is transmitted by the current active node N_i to its tree-member neighbor N_j to inform N_j that the corresponding tree link $L(N_i, N_j)$ is no longer a tree member in the PS Algorithm. This message has the same format as the mark-tree-link message but with a different message ID 7. It also has a one-hop life-span value.

Each node alternates between two states: "passive" and "active". In the "passive" state, the node is idle except that it may still receive mark-tree-link messages or release-tree-link messages from its neighbors, and mark the corresponding links as tree members or cancel their memberships. When a node receives a token, it will enter the "active" state. In this state, the node will mark itself as a tree member, select a parent from its marked tree-member neighbors with the largest capacity, and transmit a release-tree-link message to all the tree-member neighbors except its parent node. Then it will transmit a mark-tree-link message to all its non-tree-member neighbors. After finishing these procedures, this node become inactive and enters its "passive" state.

Fig.4.7 and Fig.4.8 are the pseudo code of the PS algorithm and the AS algorithm respectively. We use M_a and M_b to represent the mark-tree-link message and release-tree-link message respectively.

```
Do while there is no token and the TT algorithm does not end
{
    Receive message;
    {       Switch on the received message from link L(Nj, Ni )
            {
                    Case (Ma):Mark the link as a tree member;
                    Case (Mb):Mark the link as a non tree member
            }
    }
}
```

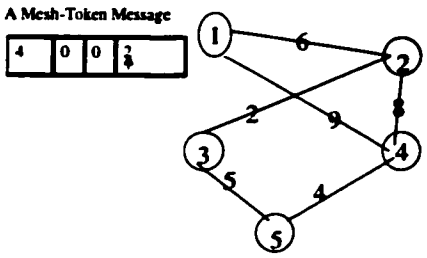
Fig. 4.7: Pseudo code of the PS algorithm

```
If node Ni is a root node
  {
      For each neighbor node Nj in the descending order of link capacity
      {
            Mark the corresponding link as a tree member;
            Send a Ma to Nj;
      }
      Set this node as a tree member;
  }
Else
{
    For all tree neighbors Nj
    {
        If NK is the node connected by the link of the largest capacity;
        Set NK as its parent;
        Else
        {
              Mark the link as a non-tree member
              Send a message Mb to Nj;
        }
    }
    For all the non-tree neighbor Nj in the descending order of link capacity
    {
              Mark the link as a tree member;
              Send a message Ma to Nj ;
    }
}
    Mark Ni as a tree member;
    Find a shortest path from the Ni to NK;
```

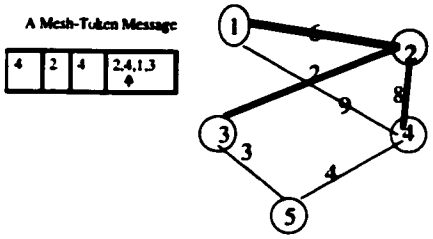
Fig. 4.8: Pseudo code of AS algorithm

4.2.3 An Example Using the Mesh Token Polling

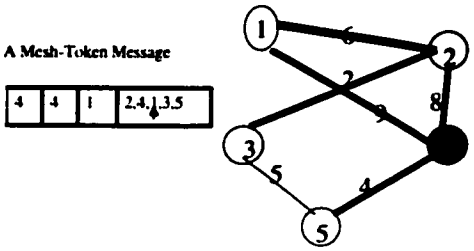
Fig. 4.9 is an example to show how the spanning tree is constructed in the TT algorithm by showing the detail procedures of the AS algorithms for all the nodes. In all these diagrams, the nodes in a dark color are tree members. The nodes in a light color are the ones that have not received a token message yet and therefore are not connected to the tree. The dark-colored solid links are tree members. A dotted line indicates that a previous tree-membership has been cancelled. The thin lines are the non-tree members. The block in the diagram is a Mesh-Token message (Fig.4.4) and the arrow in the block is the pointer.



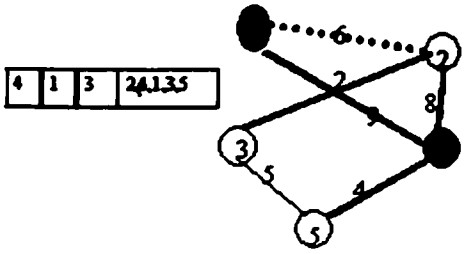
(a) Node 2 is the root node and begins its AS algorithm



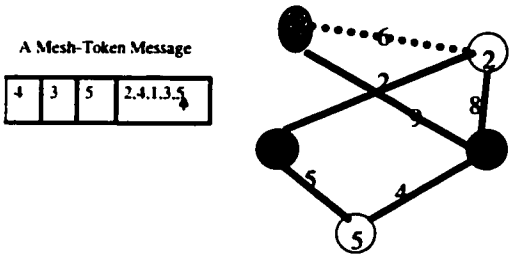
(b) Node 2 transmits mark-tree-link message to node 4.1.3 and prepares a Mesh-Token message to node 4. Node 2 finishes the AS algorithm and enters its PS algorithm



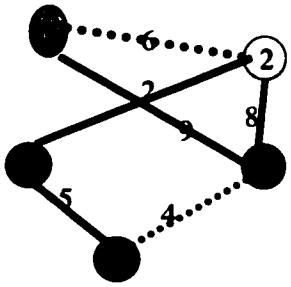
(c). Node 4 begins its As algorithm, transmits mark-tree-link messages to Node 1.5 and prepares a Mesh-Token message to node 1. Node 4 finishes the AS algorithm and enters its PS algorithm.



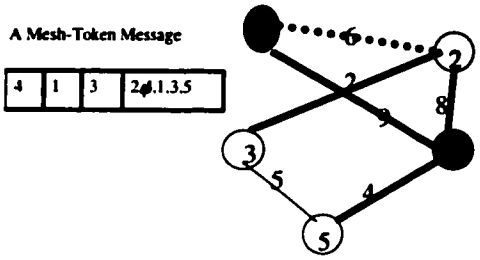
(d) Node 1 begins the AS algorithm, transmits Release-tree-link message to node 2 and prepares a a Mesh-Token message to node 3. Then nodes enters its PS state



(e) Node 3 begins its AS algorithm, transmits a mark-tree-link message to node 5 and prepares a Mesh-Token message to node 5. Node 3 enters its PS state



(f) Node 5 begins its AS algorithm, transmits a Release-tree-link message to node 4. The algorithm finishes



(d) Node 1 begins the AS algorithm, transmits Release-tree-link message to node 2 and prepares a a Mesh-Token message to node 3. Then nodes enters its PS state

Fig. 4.9: An example of the TT Algorithm with the MTP

Diagram (a) of Fig. 4.9 is the initial state of the algorithm with the root at node 2. This root enters its active state and execute the AS algorithm. It adds itself to the node ID list. The pointer in the node ID list therefore points to 2 in the node ID list. In Diagram (b), node 2 transmits a mark-tree-link message to nodes 4, 1 and 3. After node 2 appends 4, 1 and 3 to the node ID list; it will move the pointer to 4 in the node ID list, and transmit the token to node 4. After that, node 2 enters its passive state and executes the PS algorithm. Diagram (c) shows after receiving the token, node 4 begins the AS algorithm. It marks itself as a tree member. The node 4 will transmit a mark-tree-link message to nodes 1 and 5. Because nodes 1 and 2 already in the node ID list, node 4 only appends its neighbor node 5 to the node ID list, moves the pointer to 1 in the node ID list and transmits the token to node 1. Node 4 then enters its passive state and executes the PS algorithm. In Diagram (d), node 1 begins the AS algorithm. It will select node 4 as its parent and transmits a non-tree-link message to node 2. It then prepares a Mesh-Token message to node 3. Because all its neighbors are already in the node ID list, so there is no node ID added into the Mesh-Token message. Node 2 will receive a release-tree-link message in its passive state and it will only need to conceal the tree-membership of the corresponding link. Then node 1 enters its PS state. Similar steps are repeated for nodes 3 as illustrated in Diagram (e) respectively. Diagram (f) shows that node 5 receives the token and marks itself as a tree member. After node 5 selects node 3 as its parent and transmits a release-tree-link message to node 4, it will move the pointer to the next which is NULL. The algorithm ends.

4.3 Performance Analysis and Evaluation

4.3.1 Properties of the Built Tree

Since a token is used to activate a node (executing the AS algorithm) and therefore form a spanning tree, the token sequence decides the node sequence to be connected to the spanning tree. In both the PS and the AS algorithms, the tokens are distributed in a breadth-first search sequence without repetition. So there is no loop in the tree.

Like any other heuristics we cannot guarantee that maximum capacity be reached by our heuristics. However, we have incorporated two steps in our heuristics that we believe would help to build trees to achieve a maximum capacity. In the first step, when a node selects its parent, a tree member neighbor connected by the link with the largest capacity is selected. In the second step, mark-tree-link messages are transmitted to the neighbors that are non-tree-member nodes. When these neighbors select their parents, they will select a node connected by a link with the largest capacity again.

4.3.2 Complexity Analysis

When a node executes the AS and PS algorithms, it will transmit either a mark-tree-link message or a release-tree-link message to all its neighbors, and it may receive a release-tree-link message from some of the neighbors. The worst-case message complexity for this part is 2D where D is the average nodal degree. At the same time the TP algorithm will transmit a token to activate the AS algorithm in a node. Because each node may become an intermediate node of the other node in transmitting the token, the worst-case message complexity for this part is N. So for each node, the overall message complexity in the worst case is 2D+N. The total worst-case message complexity for building the tree is therefore N(2D+N) which is of order $O(N^2)$.

From the above analysis, we can also get the time complexity for this distributed tree algorithm: The worst case time complexity for selecting the parent is O(D), which is the time to process the mark-tree-link message and the release-tree-link message. The worst case time complexity for connecting the children is O(D). The time complexity to poll a token and to transmit the token is O(N). So the overall time complexity in the worst case for each node is O(ND). The total worst-case time complexity for building the tree is therefore of order O(N^3). The time complexity of the Token Tree Algorithm is larger than the time complexity of the classic distributed algorithm discussed in chapter one that has a O($N^2 \log N$) time complexity to build a spanning tree, but the message complexity of the Token Tree Algorithm is smaller and there are only four types of message to be used in the Token Tree Algorithm.

4.3.3 Performance Evaluation

We have run our algorithms in topologies based on some of the existing networks to obtain the performance data. These networks are the ones introduced in Chapter 2. The performance measures we use here are the restorability, capacity ratio, average hop count, number of messages and visiting times. We have given the definitions of these measures in Chapter Two. In addition, we define " the number of hop used in Star-Token (Mesh-Token)" to be the total number of hops traversed by all the tokens divided by (N-1), where N is the total number of nodes in the network. The ratio of the spare capacity to working capacity is first fixed at 0.5 for all links in the whole network. Then simulations are repeated for ratios of 1.0, 1.5 and 2.0 respectively

4.3.3.1 Performance of The Token Tree Algorithm

Table 4.1: Simulation result of the TT Algorithm

Item	US28	FranceTelecom	Japan	MCI
Number of Nodes	28	44	56	41
Average Nodal Degree	3.2	3.18	2.96	2.93
Average Hop Count of Shortest Path	5.423	6.215	8.034	6.702
Capacity Ratio	0.671	0.681	0.724	0.733
Number of Message A	45	70	83	60
Number of Message B	18	27	28	20
Number of hops used in Star-Token	2.51	2.67	3.65	3.68
Number of hops used in Mesh-Token	2.59	2.44	3.35	2.78
Number of times Visited	28	44	56	41

Table 4.2: Simulation Result of the Hierarchical Tree Algorithm

Item	US28	FranceTelecom	Japan	MCI
Number of Nodes	28	44	56	41
Average Nodal Degree	3.2	3.18	2.96	2.93
Average Hop Count of Shortest Path	5.438	6.128	7.608	6.778
Capacity Ratio	0.661	0.639	0.696	0.691
Number of Messages	1218	2454	3908	2042
Number of times Visited	46	58	72	52

Tables 4.1 and 4.2 are respectively the simulation results of the TT algorithm and the HT algorithm. From the simulation result, we can see that except in the Japan Network, the Mesh-Tokens traverse smaller hop-numbers than the Star-Tokens. Since the Mesh-Tokens do not need node BEGIN to know the whole network topology, so Mesh-Token Polling algorithm is suitable for the distributed MST algorithm.

Except in the FranceTelecom Network and the Japan Network, the tree built by the TT algorithm has a smaller average hop number. This average hop-number can be seen as the mean path length for the restoration path. In all of the four networks, the trees completed by the TT algorithm have a larger capacity ratio than the trees built by the HT

Algorithm. The visiting times and the messages transmitted during the execution of the algorithm in the TT algorithm are much less than those of the HT algorithm. But the tokens have to traverse some hops when constructing the spanning tree. There is no such cost in the HT algorithm. So this is the trade-off of using the tokens to reduce the message number.

So the comparison between the performance of the TT algorithm and Hierarchical Tree Algorithm shows that the performance of the algorithms depends a little bit on the network topology. Except the visiting times, the difference of the other performance measures is very small between these two algorithms. So using tokens to reduce the visiting times for each node can have a desirable result.

4.3.3.2 Restorability

Table 4.3: Restorability Comparison

Item	US28	FranceTelecom	Japan	MCI
KSP Restorability	0.448	0.449	0.444	0.400
CST Restorability	0.097	0.095	0.107	0.070
TT Restorability	0.275	0.290	0.326	0.260

Table 4.4: Confidence Interval of the TT Algorithm

Item	US28	FranceTelecom	Japan	MCI
Average Hop Count of Shortest Path	5.127	5.715	7.652	6.362
	5.720	6.716	8.416	7.042
Capacity Ratio	0.665	0.670	0.718	0.726
	0.677	0.692	0.732	0.739
TT Restorability	0.253	0.256	0.303	0.225
	0.298	0.324	0.349	0.294

The evaluation is done by simulation. Each simulation collects 10 samples and takes bout 10 minutes to run in Java on a Pentium Cyrix 120 MHz Computer using NT System. We have

obtained 95% confidential interval of some important parameters shown in Table 4.4. The small interval values indicate the convergence of our simulation results.

After we build a spanning tree to protect the non-tree links, we would seek out the shortest path to protect each tree link in the TT algorithm. To see the difference among different protection schemes, we compute and compare their restorability respectively.

The simulation results are presented in the Table 4.3 by picking a ratio (the spare wavelengths to the working wavelengths) of 1. Here the KSP restorability refers to the restorability using the K-Shortest Path Algorithm [DuGr94]. The CST restorability refers to the restorability using the Conventional Spanning Tree algorithm. The TT algorithm restorability refers to the restorability of our algorithm. Our algorithm has a better restorability than the conventional tree protection scheme, but a lower performance than the K-successively Shortest link disjoint Path restorability.

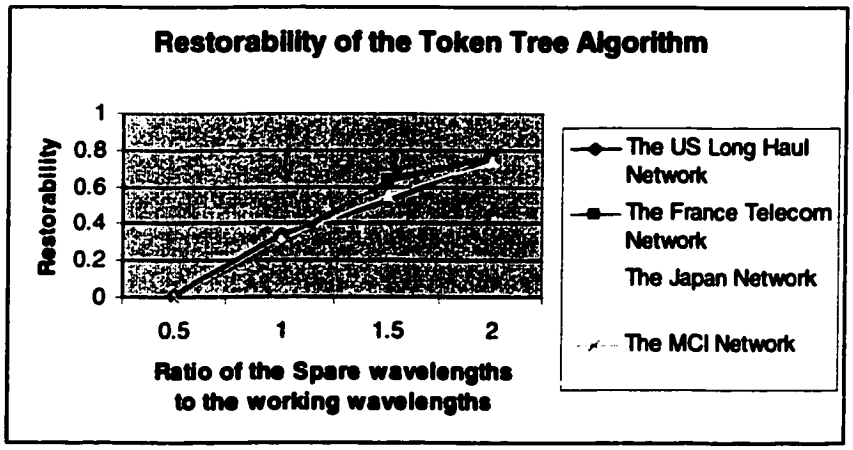


Fig. 4.10: Restorability of the Token Tree Algorithm

The ratio of the spare wavelength to the working wavelength will make a big difference to the restorability. In the simulation, we set the ratio of the spare capacity to working capacity as 0.5, 1.0, 1.5 and 2.0 respectively. Fig. 4.10 shows the restorability of

the TT algorithm. The simulation result shows that the higher the ratio is, the larger the restorability.

4.4 Concluding Remark

In this paper, we proposed a TT algorithm to build a distributed maximum spanning tree (MST) for network protection/restoration in a mesh network. The essence of the algorithm is that each node picks up a parent from the connected tree-member neighbor. The simulation shows promising results for our heuristic algorithm. Although the performances of the average hop number and the capacity ratio of the trees built by the TT algorithm and the HT algorithm have only a small difference, our TT algorithm has smaller visiting times, and the number of messages transmitted are 10 times less. Even though we did not run these algorithms in bigger network, since we use tokens to decide the executing sequence and control the visiting times for each node, the simulation result should be scalable to bigger network.

The STP algorithm needs a node to know the whole network topology while the MTP algorithm does not need such a node. While in most of the networks, Mesh-Tokens traverse smaller hop-numbers than the Star-Tokens, the Mesh-Token distribution is more suitable for a distributed MST algorithm.

In the study on the impact of the ratio between the spare and working wavelengths, we can see that using tree links to protect non-tree links while seeking out the shortest path to protect the tree links can give us a better restorability than using only tree links. The trade-off is the increased hop count for the tokens.

Chapter Five Triangular Tree Algorithm

From our experience with the implementation of the tree algorithm and their performance evaluation, we have discovered a basic triangular structure for network protection. This forms the basics of a new algorithm, called the Triangular Tree Algorithm which we would like to explore its implementation and study its performance. We also discuss its variants and properties.

5.1 Definitions and Problem Statement

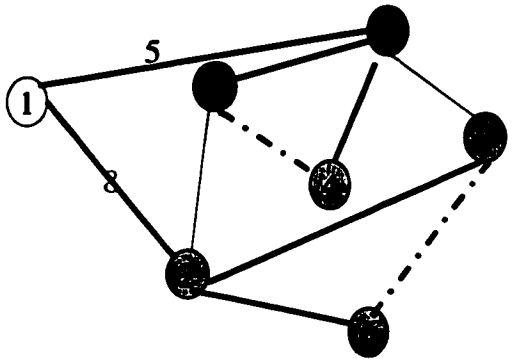


Fig. 5.1: A Triangular Tree overlain on a Mesh Network

The goal of the triangular tree algorithm is to construct a logical triangular tree topology from a selected root on a physical mesh network. Fig.5.1 is a logical triangular tree topology overlaid on a physical mesh network. The lines in a dark color are the tree-member links. The dark-dashed lines are the base-links of the triangles. The light dashed lines are neither tree-member links nor triangular base links, and we call them non-tree-member links. Each node in a triangular tree has two kinds of children: triangular children and non-triangular children. A triangular child has links connected to its siblings and therefore the parent and its two triangular children form a triangle. In the triangle, the tree-links connecting the parent and its two children are called triangular links, and the non-tree link connecting the two children is called a triangular base.

The basic structure is a triangle formed by a node, its parent node and the sibling node of the same parent. In the example of Fig.5.1, Nodes 3,4 and 5 form a triangle with node 5 as the parent and nodes 3, 4 as the triangular children. The links L(5,3) and L(5,4) are the triangular links while link L(3,4) is the triangular base. The non-triangular child does not have any connection to its siblings (Nodes 2 and 5).

The protection scheme lies in using two links in the triangle to protect the other link and seeking out a shortest path from one end node to the other end node to protect the corresponding link that is not included in the triangle.

5.2 Implementations

The nature of the Triangular Tree Algorithm is to select children for each node in the network. The nodes with a smaller hop number to the root have a higher priority to select their children. In the case of a tie, the one with a larger capacity has a higher priority. The criterion to select children is as following: when selecting children, a node will select the triangular children with a higher priority and these children cannot be selected by other nodes. Those non-triangular children will be selected without a higher priority and can be selected by other nodes as triangular children.

The Triangular Tree Algorithm can also be implemented in two different ways: Central Triangular Tree Algorithm (CTT) and Distributed Triangular Tree Algorithm (DTT). Both of the algorithms begin from the root and need a two-step comparison when a node selects the non-triangular children.

5.2.1 The CTT Algorithm

We assume here all the connecting information of the network has been obtained. With reference to the problem of the root selection has been discussed in Chapter 3, this can be done by using the CRS, the URS or the DRS algorithms.

```
Step1: Put the root into the queue;
Step2: While the queue is not empty
{
        Pop a node N_V from the queue;
        Sort the neighbors of N_V by their capacity in a descending order;
        For each non-tree-member neighbor N_I according to the sorted order
        {
                If N_I is connected to other non-tree-member neighbor of N_V or connected to other
                tree-member neighbor with the same parent N_V
                {
                        Put N_I into the queue;
                        Mark N_I as a tree-member with a higher priority;
                        Record N_V as N_I 's parent;
                }
                Else If not the case that N_I has one tree-member neighbor N_K connected to a non-
                tree-member neighbor N_J
                {
                        Put N_I into the queue;
                        Mark N_I as a tree-member;
                        Record N_V as N_I 's parent;
                }
        }
        If N_V does not have a higher priority
                Find a shortest path to its parent;
        }
```

Fig. 5.2: Pseudo Code of the Central Algorithm

Fig. 5.4 is the pseudo code of the central algorithm. The CTT algorithm establishes a queue to hold the nodes in the sequence to be connected to the tree. The algorithm starts by giving the root node a higher priority and puts it into a FIFO (First-In-First-Out) queue. All the nodes with a higher priority will have a decided parent-child assignment and cannot be changed later. Before a node is put into the queue, it will be marked as a tree member. Each time, the CTT algorithm pops a node from the queue, and this node will select its children from the non-tree member nodes. If the selected children are triangular children, this node will put them into the queue directly with a high priority. If the children are non-triangular children, the algorithm will check whether these children can be selected by other tree-member nodes as their triangular-children. If yes, these children will not be put into the queue. Otherwise, these children will be put into the queue without a higher priority. If the popped node is a node without a higher priority, The CTT

algorithm will find the shortest path [Tane96] from this node to its parent. All the node will be given a tree ID by its parent before it is put into the queue. The above procedure repeats until the queue is empty.

5.2.2 The DTT Algorithm

The DiRS algorithm (discussed in Chapter 3) can be used to select a proper root for the DTT algorithm. After the root selection, the following algorithm can be executed.

The DTT Algorithm accomplishes its functions by exchanging two messages among nodes. In these messages, "Message ID", "Source Address" and "Destination Address" have the same definition as those in Chapter Three. "Node Information" includes current node status. "Hop Number" refers to the number of hops this message has passed. "Priority" is used to identify the message's priority. 'Tree ID" (Chapter 2) is used to identify the node position in the built tree.

Message ID (9)	Source Address	Destination Address	Node Information	Hop Number

Fig. 5.3: A Node-Info message

A Node-Info message (Fig.5.5) is used to exchange connection information between nodes. The life span for this message is set to two hops because the message is not needed after it pass the information of the two nodes. Upon receiving a Node-Info message, a node will check the Hop-Number field. If the value is two, this message will be discarded. Otherwise, it will increment the hop number by one and transmit a copy of this message to all its neighbors except the one where the message comes from.

Message ID (8)	Source Address	Destination Address	Priority	Tree ID

Fig. 5.4: A Select-Child message

```
Step 1: Nodes exchange Node-Info messages between neighbors;
Step 2: The root send messages to its neighbors:
        Send a Select-Child message with a higher priority to its triangular child
          neighbors;
        Send a Select-Child message without a higher priority to its non-triangular child
          neighbors;
        Mark itself as a tree member;
    Step3: For each node N_i
        {Receives messages:
                If the message is a Node-Info message
                {
                        If the hop-number is greater than 2
                                Delete the message;
                        Else transmit a copy to its neighbors except the one where the
                          message came from;
                }
                Else if the message is a Select-Child message with a higher priority
                {
                    Send a Select-Child message with a higher priority to its triangular
                      Child neighbors;
                    Send a Select-Child message without a higher priority to its non-
                     triangular child neighbors;
                    Mark itself as a tree member;
                    Record the node where the message comes from as its parent;
                }
                Else if the message is a Select-Child message without a higher priority
                {
                    if N_i has one tree-member neighbor N_J and has other non-tree-member
                      neighbors connected to N_J
                    {
                        Wait until a Select-Child message with a higher priority comes;
                        Send a Select-Child message with a higher priority to its triangular
                          Child neighbors;
                        Send a Select-Child message without a higher priority to its non-
                         triangular child neighbors;
                        Mark itself as a tree member;
                        Record the node where the message comes from as its parent;
                    }
                    else
                    {
                        Send a Select-Child message with a higher priority to its triangular Child
                        neighbors;
                        Send a Select-Child message without a higher priority to its non-
                         triangular child neighbors;
                        Mark itself as a tree member;
                        Record the node where the message comes from as its parent;
                    }
                }
    }
```

Fig. 5.5: Pseudo code of the distributed algorithm

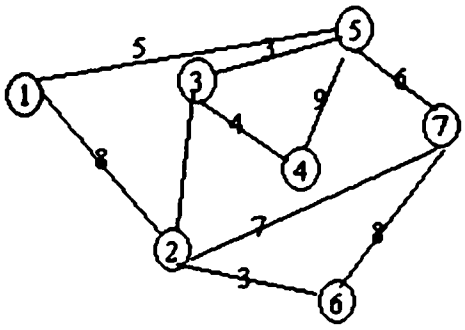
Select-Child message (Fig.5.6) is used to inform its corresponding non-tree-member neighbor that this neighbor is selected as its child. If the value of the priority field is set as 1, this child is a triangular-child and cannot be selected by other nodes. If the value is 0, this child is a non-triangular child and can be selected by other nodes as a triangular child.

Figure 5.7 is the pseudo code of the DTT algorithm. The DTT algorithm begins from the selected root. Each node exchanges Node-Info messages with its neighbors at the start-up time and when there is a status change of the node. Each node will execute the algorithm once. When executing the algorithm, the node will mark itself as a tree member first and then transmit Select-Child messages to their non-tree-member neighbors to mark them as their children. Because the node already knows the information of its children, it can choose to transmit a Select-Child message with a higher priority or without. When the child node receives a Select-Child message with a higher priority, it will begin to execute the same procedure as described above. Otherwise it will check its neighbors' status. If it can be selected by other tree-member nodes as a triangular child, it will wait for that Select-Child message. If it cannot be selected, it will repeat the same procedure as its parent does.

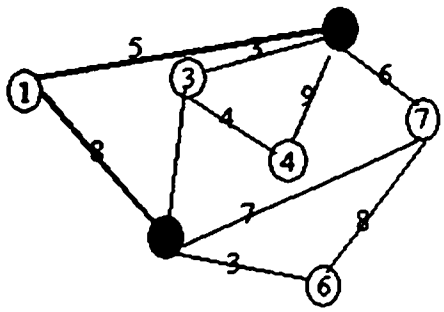
5.2.2.1.1 An Example of the Triangle Tree Algorithm

Fig. 5.8 is an example for building a triangular tree in a distributed way on a network with 7 nodes. The explanation of the links and nodes are the same as described in section 3.2.1.1. Diagram (a) shows node 1 is selected as the root. In Diagram (b), node 1 selects nodes 5 and 2 as its non–triangular children without a higher priority. Diagram (c) shows that node 2 selects nodes 6 and 7 as its triangular children with a higher priority, and it selects node 3 as its non-triangular child without a higher priority. Diagram (d) shows that

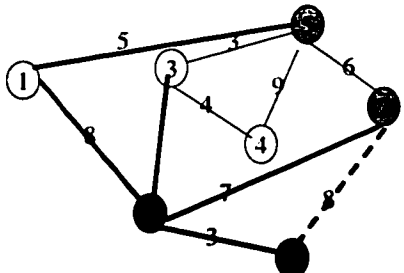
node 5 selects nodes 3 and 4 as its triangular child with a higher priority, and link (3,2) becomes a non-tree link again.



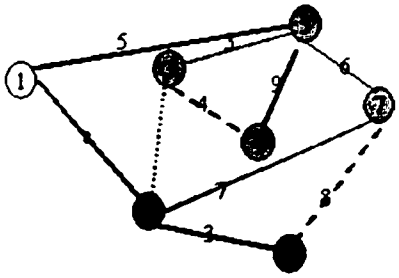
(a) A network of 7 nodes with the root node 1



(b)Node 1 selects nodes 5 and 2 as its children without priority



(c) Node 2 select nodes 6 and 7 as its children with higher priorities, select node 3 as its children without priority.



(d) Node 5 selects nodes 3 and 4 as its children with higher priorities, link (3,2) becomes a non-tree link

Fig. 5.6: An example of the Triangle Tree Algorithm

Note that in Diagram (b), when selected as non-triangular children by node 1 without a higher priority, both node 2 and node 5 become tree members (indicated by a dark color). Both of them have no tree-member connected to other neighbors. But in Diagram (c), when selected as a triangular child by node 2 without a higher priority, node 3 does not become a tree member and does have a change of color. This is because node 3 has a tree-member neighbor (node 5) connected to its neighbor node 4. In this case, node 3 can be chosen by node 5 as a triangular child later.

From the example we can see that when a node receives a message with a higher priority, it will become a tree-member directly (indicated by changing its color). When it receives a message without a higher priority, it may become a tree member and may not change its color directly.

5.3 Performance Analysis and Evaluation

We have evaluated our algorithms by simulation (Section 2.7). A typical run of a 44-node network takes about 10 minutes. Our Simulation results have been tested for convergence as demonstrated by some 95% confidence intervals of some important parameters as shown in Table 5.5.

5.3.1 The Triangular Tree is Loop Free

In the CTT, we use a queue to hold the node sequence to be connected to a tree. Since these nodes are put into the queue one by one without repetition. It can be assumed that there is no loop in the built tree.

In the DTT, each tree member node will select non-tree member neighbors as its children and these children will become tree members. All the triangular children cannot be selected by the other nodes. For the non-triangular children, if they can be selected by others to form a triangular, they will identify themselves as a tree member until a select-child message with a higher priority comes. So the nodes are connected to the tree by only one parent-child assignment, and there is no loop in the tree.

5.3.2 Time Complexity Analysis

When a node (called node A) has no connected sibling in the CTT algorithm (or when a node receives a message without priority in the DTT algorithm), a two-step comparison has to be done to decide whether node A will be marked as a tree member. The first step is to

check whether node A has a neighbor node(called node B) that is a tree member node. If it has, a further step will be taken to check whether node B has a non-tree-member node C connected to node A. The worst case to determine a node has no sibling would require N (the number of nodes in the network). The same is true for the other two steps, so the worst case complexity is O(N^3), and there is a complexity of O(N^4) to check every node in the whole network. On the average, it has a complexity of O(ND^3) where D is the average nodal degree.

5.3.3 Performance Evaluation

We simulate the triangular tree algorithms in topologies (Ref: Chapter 2) to obtain the performance data.

5.3.3.1 Simulation Result of Triangle Tree Algorithm

Table 5.1: Simulation Result of CTT Algorithm

Item	US28	FranceTelecom	Japan	MCI
Number of Nodes	28	44	56	41
Average Nodal Degree	3.21	3.18	2.96	2.93
Average Hop Count of Shortest Path	4.920	5.819	7.242	6.674
Capacity Ratio	0.621	0.622	0.666	0.676

Table 5.2: Simulation Result of the DTT Algorithm

Item	US28	FranceTelecom	Japan	MCI
Number of Nodes	28	44	56	41
Average Nodal Degree	3.21	3.18	2.96	2.93
Average Hop Count of Shortest Path	4.920	5.819	7.242	6.674
Capacity Ratio	0.621	0.622	0.666	0.676
Number of Message (5)	123.1	201.6	234.9	163.2
Number of Message (6)	39.9	49.4	61.2	41.7

Table 5.3: Simulation Result of the Hierarchical Tree Algorithm

Item	US28	FranceTelecom	Japan	MCI
Number of Nodes	28	44	56	41
Average Nodal Degree	3.2	3.18	2.96	2.93
Average Hop Count of Shortest Path	5.438	6.128	7.608	6.778
Capacity Ratio	0.661	0.639	0.696	0.691
Number of Messages	1218	2454	3908	2042
Number of times Visited	46	58	72	52

Tables 5.1, 5.2 and 5.3 are respectively the simulation results of the CTT algorithm, the DTT algorithm and the hierarchical tree algorithm [ShYa01]. Except that the DTT algorithm needs message exchanging, one sees that the CTT algorithm and the DTT algorithm have the same simulation result. So we just compare the DTT algorithm with the Hierarchical Tree algorithm.

The simulation result in both tables, "Number of Message 5" and "Number of Message 6" in this section refers to the number of Node-Info and Select-Child messages transmitted to build the spanning tree respectively.

From Tables 5.2 and 5.3, In all the four networks, we can see that the trees built by the DTT algorithm have smaller average hop numbers but larger capacity ratio than the tree built by the HT Algorithm. We can also observe that the messages transmitted (Both Node-Info and Select-Child messages) in the DTT algorithm are fewer by far than those in the hierarchical tree algorithm. Take the Japan network as an example, the number of messages can be 10 times less.

5.3.4 Restorability

Table 5.4: Restorability of Triangular Tree Algorithm (Capacity Ratio = 1)

Item	US28	FranceTelecom	Japan	MCI
Number of Message (5)	123.1	201.6	234.9	163.2
Number of Message (6)	39.9	49.4	61.2	41.7
KSP Restorability	0.442	0.449	0.415	0.435
CSP Restorability	0.075	0.080	0.080	0.063
TSP Restorability	0.296	0.316	0.325	0.336
DTT(CTT) Restorability	0.444	0.449	0.415	0.435

Table 5.5: Confidence Interval of the (CTT)DTT Algorithm

Item	US28	FranceTelecom	Japan	MCI
Average Hop Count of Shortest Path	4.764	5.377	6.576	6.258
	5.077	6.261	7.908	7.090
Capacity Ratio	0.612	0.608	0.659	0.664
	0.631	0.636	0.673	0.689
DTT(CTT) Restorability	0.411	0.424	0.380	0.405
	0.477	0.473	0.450	0.465

In a triangular tree, two links of the triangle are used to protect the other link, and seek out the shortest path to protect the link that is not included in a triangle. To see the difference this protection scheme makes, we compute and compare the restorability performance against the other protection schemes.

Simulation results are presented in Table 5.4 by setting the ratio of the spare wavelength to the working wavelength to 1. With reference to Section 1.5, "KSP restorability" refers to the restorability using the K-successively Shortest Link Disjoint Path Algorithm. "CST restorability" refers to the restorability using the Conventional Spanning Tree Algorithm. "TSP restorability" refers to the protection scheme of using the tree links to protect non-tree links and seeking the shortest path to protect the tree links.

"DTT(CTT) restorability" refers to the restorability of our algorithm. Our algorithm has a better restorability than the conventional tree protection scheme as well as the tree and the shortest path protection scheme, and the same performance as the k-Shortest Path restorability.

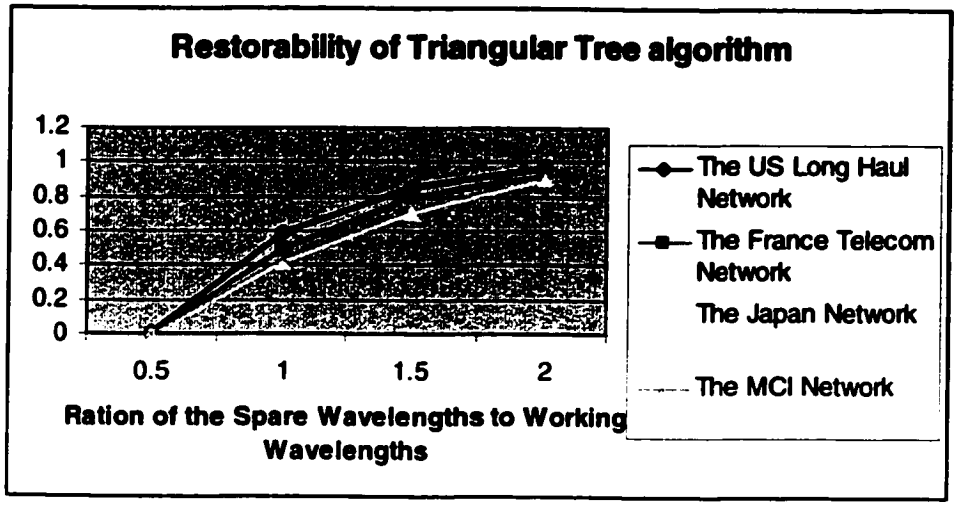


Fig. 5.7: Restorability Trend

The ratio of the spare capacity to working capacity will make a huge difference to the restorability. In the simulation, we set the ratio of the spare capacity to the working capacity as 0.5, 1.0, 1.5 and 2.0 respectively. We can see from Fig.5.10 that the larger the ratio is, the larger the restorability. As the ratio of the spare wavelength to working wavelength reaches 2, the restorability of DTT algorithm on the US Long network can even reach 100%.

5.4 Concluding Remark

In this chapter, we propose a Triangular Tree Structure for network protection/restoration, and implement algorithms to construct a spanning tree. These algorithms can be executed centrally or in a distributed manner. Simulation results indicate that our triangular tree protection scheme is a promising approach. The messages transmitted during the

constructing of the algorithm are much less than that of the hierarchical tree algorithm. The restorability of the triangular tree structure can reach the same performance as the k-shortest path restoration.

Chapter Six Conclusion

From the knowledge we gain on reviewing the development of the optical network and the tree algorithms, we have proposed three root selection algorithms for selecting a proper root for a spanning tree. In order to reduce the visiting times for each node when building the distributed spanning tree for optical network protection and restoration, we have introduced and studied a token tree algorithm, a triangle tree structure, and their variants.

The token tree protection scheme provides a mechanism which uses tree links to protect non-tree links and uses a shortest path to protect the tree links. This mechanism improves the restorability compared to the traditional tree protection scheme that only uses tree-links to protect non-tree links. On the other hand, the triangle tree protection schemes use two links in a triangular substructure (in a tree) to protect the third link, and uses a shortest path to protect the corresponding link that is not included in a triangle. These algorithms can have the same restorability as a k-shortest path mesh restoration protection scheme.

6.1 Future Work

There are some interesting aspects we feel are worthy of further study.

First we need to find methods to split the failed link capacity among different protection paths, and therefore to enhance the restorability. Since the triangle tree algorithm has a larger time complexity, we would like to reduce the complexity of this algorithm. We also believe that constructing signaling system for the protection detection and failure restoration is a practical procedure to these algorithms to be implemented in the real network. We also need to build the routing mechanism in the tree system to make the tree

system feasible for network protection/restoration. Finally, we would like to optimize the wavelength distribution among the tree system to improve the capacity efficiency.

Reference

[AnGr94] D. Anthony, Wayne D. Grover, "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 823-834.

[BLSR95] GR-1230-Core, SONET Bidirectional Line-Switched Ring Equipment Generic Criteria, Bellcore, Issue 2, November 1995.

[Brua92] R. Brualdi, Introductory Combinatorics, Englewood Cliffs, Prentice Hall, 1992.

[ChBi93] C.E. Chow, J. Bicknell, S. McCaughey and el al, "A fast distributed network restoration algorithm", Proc. of 12th Int. Phoenix Conf. Computers and Communications, Phoenix, AZ, 1993, pp 261-267

[DoWi94] R, Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration algorithm", J. Network and Systems Management, Vol. 2, No.2, 1994, pp 95-123

[DuGr94] D.A. Dunn, W.D. Grover, M.H. MacGregor, "A comparison of k-shortest paths and maximum flow methods for network facility restoration," IEEE Journal on Selected Areas in Communications, Jan. 1994, vol. 12, no. 1, pp. 88-99.

[ElGe00] George 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.

[ElMo02] Jaafar m.H. Elmirghani, Hussein T. Mouftah, "Technologies and Architectures for Scalable Dynamic Dense WDM Networks", IEEE Communications Magazine, Vol.38, No.2, Feb.2000, pp.58-66.

[FaMo95] Michalis Faloutsos and Mart Molle, Creating Optimal Distributed Algorithms for Minimum Spanning Trees, Thesis, University of Toronto May 9, 1995

[FaWE90] W.E. Falconer, "Service Assurance in Modern Telecommunication Networks," IEEE Communications Magazine, Special Issue "Surviving Disaster", June 1990, pp 32-39

[FiTa97] Steven G. Finn, Muriel M. Medard, Richard A. Barry, "A Novel Approach to Automatic Protection Switching Using Trees", Proc. of 1997 IEEE International Conference on Communication (ICC'97), Montreal, CA, June 1997, pp272-276

[FriT97] T. Frisanco, "Optical spare capacity design for various protection switching methods in ATM networks", Proc. of 1997 IEEE International Conference on Communication (ICC'97), Montreal, CA, June 1997, pp 293-298

[GaHu83] R. Gallager, P. Humblet, and P. Spira. "A distributed algorithm for minimum-weight spanning trees," ACM Transactions on Programming Languages and Systems, vol. 5, no. 1, Jan. 1983, pp. 66-77

[GaMu99] Muralikrishna Gandluru, " Optical Networking and Dense Wavelength Division Multiplexing(DWDM)", Course materials, 1999, http://www.cis.ohio-state.edu/~jain/cis788-99/dwdm/index.html

[GeFr88] Mario Gerla and Luigi Fratta, "Tree Structured Fiber Optics MAN's" 1988 IEEE Journal on Selected Areas in Communications, VOL. 6, No. 6, July pp934-pp943

[GeRa00] O.Gerstel, R. Ramaswami, "Optical layer survivability - An implementation perspective," IEEE JSAC Special Issue on Next Generation Optical WDM Networks, vol.18, no.10, Oct. 2000, pp.1885-1899.

[GoTa00] Michael T. Goodrich and Roberto Tamassia, Data Structures and Algorithms, John Wiley and Sons (WIE), Aug.2000

[Grov89] W.D.Grover, "Self healing network: A distributed algorithm for k-shortest link-disjoint paths in a multigraph with applications in real time network restoration." Ph.D. Dissertation, university of Alberta, Fall, 1989.

[GrSt98] Wayne D. Grover, Demetrios Stamatelakis, "Cycle-Oriented Distributed Preconfiguration: Ring-like speed with mesh-like capacity for self-planning Network Restoration", Proc. of 1998 IEEE International Conference on Communication (ICC'98), Atlanta, USA, Vol.1, 1998, pp 537-543.

[ITUT872] International Telecommunications Union-Telecommunication Standardization Sector (ITU-T) Recommendation G.872

[KaBe96] Leonid Kazovsky, Sergio Benedetto and Alan Willner, Optical Fiber Communication Systems, Artech House, Boston, London, 1996

[KaSt00] Stamations V. Kartalopoulos, Introduction to DWDM Technology: data in rainbow. Wiley, John & Sons, Incorporated, 2000

[Lele98] S.R.Lee and G.H.Lee, Web Materials, http://optcom.korea.ac.kr/optinet/opnet_home.htm#What%20is%20Optical%20Network?, 1998

[LuMe00] Lumetta SS, Medard M, Tseng YC , "Capacity versus robustness: A tradeoff for link restoration in mesh networks", J. of Lightwave Technology, Vol.18, December 2000, pp 1765-1775

[MeFi99] Muriel Medard, Steven G. Fiin and 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.641-652.

[MoGr98] D. Morley and W.D. Grover, "A Comparative Survey of Methods for Automated Design of Ring-based Transport Networks", TRLabs Technical Report, TR-97-04, 1998

[MaJC94] J. C. MacDonald, "Public Network Integrity - Avoiding a Crisis of Trust", IEEE JSAC Integrity of Public Telecommunication Networks, vol.12, no.1., Jan. 1994, pp. 5-12.

[MuBi00] B. Mukherjee, "WDM optical communication networks: progress and challenges," IEEE Journal Selected Areas in Communications, vol.18, no.10, Oct. 2000, pp. 1810-1824.

[MuBi92] Biswanath Mukherjee, "WDM-Based Local Lightwave Networks Patr I: Single-Hop System" IEEE Network, May 1992, pp12-27.

[MuGu02] C. Siva Ram Murthy and Mohan Gurusamy, "WDM Optical Networks: Concepts, Design, and Algorithms", Prentice Hall, 2002

[NIST1] http://www.nist.gov/dads/HTML/spanningtree.html

[NIST2] http://www.nist.gov/dads/HTML/binarytree.html

[RaMu99a] S. Ramamurthy and B. Mukherjee, " Survivable WDM mesh networks-Part I: Protection", Proc. Infocom, 1999, pp 744-751

[RaMu99b]S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks-Part II: Restoration", Proc. of 1999 IEEE International Conference on Communication (ICC99), Vancouver, Canada,Vol.3 1999, pp 2023-2030

[Schw88] M. Schwatz, Telecommunication Networks: Protocol, Modeling and Analysis, Addison-Wesley, 1988

[ShYa01] Shaharam Shah-Heydari and Oliver W.W.Yang, "A Tree-based Algorithm for Protection/Restoration in Optical Mesh Networks", CCECE, May 2001, Toronto, Canada, pp.1169-1174.

[StGr00] D. Stamatelakis, W.D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Pre-configured Cycles ("p-cycles")," IEEE Transactions on Communications, vol.48, no.8, August 2000, pp.1262-1265.

[StGr99] Demetrios Stamatelakis and Wayne D. Grover, "Network Restorability Design Using Pre-configured Trees, Cycles and Mixtures of Pattern Types", TRLabs Technical Report, TR-1999-05

[SzYa01] Lech Szymanski and Oliver W. W. Yang, "Spanning tree algorithm for spare network capacity", Proc. of CCECE, Toronto, Canada, May 2001, pp.447-543.

[Tane96] Andrew S. Tanenbaum, "Computer Networks", Prentice Hall, 1996

[TRLAB99]C Codes of the Trlabs ring-finding Algorithm, from TRLabs, University of Edmonton, Alberta, Canada

[UPSR95] GR-1400-Core, SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria, Bellcore, Issue 1, Revision 1, October 1995.

[WaLi99] Peng-jun Wan, Liwu Liu and Ophir Frieder,"Optimal Placement of Wavelength Converters in Trees and Trees of Rings", Proc. of IEEE IC3N' Boston, Massachusetts, October 11-13,1999, pp.392-397

[WuTs92] Tsong-Ho Wu, "Fiber Network Service Survivability", Artech House, Boston, London, 1992

[Yang94] Oliver W.W.Yang, "Two-center Tree Topologies for Metropolitan area networks", IEEE Proc.-Commun., Vol.141, August 1994, pp.280-288

[YaSh02] Oliver W.W.Yang, Shahram Shah-Heydari, Yuna Zhang, "Hierarchical Network Protection Scheme for Failure Recovery in Optical Mesh Networks", CCNR Technical Report, February 2002

[Zhan02] Hanxi Zhang, "DWDM Network Protection/Restoration with Ring/Cycle Topologies", Master Thesis, University of Ottawa, March 2002.