Label Assignment and Failure Recovery Approaches for IP Multicast Communication in MPLS Networks

Omar Qasem Banimelhem

A Thesis

In

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

December 2005

Your file Votre référence
ISBN: 978-0-494-16290-3
Our file Notre référence
ISBN: 978-0-494-16290-3

ABSTRACT

Label Assignment and Failure Recovery Approaches for IP Multicast Communication in MPLS Networks

Omar Qasem Banimelhem, Ph.D.

Concordia University, 2005

Multiprotocol Label Switching (MPLS) is an Internet Engineering Task Force (IETF) framework that provides for the efficient designation, routing, forwarding, and switching of traffic flows through the network. MPLS is widely used for multicast traffic engineering. However, integrating MPLS with IP multicast communication is difficult. This thesis proposes solutions for two problems: label assignment and failure recovery.

When a set of multicast flows (sessions) is deployed in an MPLS network, each multicast entry for the sessions consumes an MPLS label. However, MPLS labels are significant resources in MPLS networks, as the 20-bit field of the MPLS header limits the number of available labels. It is advantageous if different multicast sessions can share the same multicast tree in the network and re-use the MPLS label. In the first part of the thesis, we propose two algorithms to do this. The first one is called State Encoding (SE), in which a code is calculated for every tree built in an MPLS network. The second algorithm is called Tree Numbering (TN), where a number represents each tree. If the IP packets of different multicast sessions are delivered over the same tree, all those packets are then classified to the same Forwarding Equivalence Class (FEC) and only one label is used instead of using a number of labels equal to the number of those sessions. If the SE

algorithm is used, then the tree code will represent the FEC. If the TN algorithm is used, the tree number represents the FEC.

Network survivability when faults occur is important, especially when continuous media are carried. However, little research has been done that targets MPLS-based recovery for multicast communication. MPLS-based recovery can be implemented using local recovery, in which a backup path is set up for each element in the multicast tree. The drawback of local recovery is that a large amount of capacity is reserved over the backup paths. The amount of reserved capacity can be reduced if global (end-to-end) recovery is used. However, the disadvantage of global recovery is that it takes a long time to recover from a failure.

The second part of this thesis contributes to this area. To trade off between the large amount of bandwidth required for reserving backup paths in local recovery and the large recovery time taken in global recovery, a new tree division approach is proposed. In this approach, a multicast tree is divided into several domains, where each domain represents a local stand-alone sub-tree of the original one. Two MPLS-based methods are then proposed to set up the backup paths inside the domains.

A comparison is made among the local recovery approach, the global recovery approach, a method that sets up the backup paths between the branching points, and the proposed approach. The comparison is based on three metrics: the total backup capacity, the maximum and the average notification times, and the average number of the notification messages that are produced as a result of a failure. In terms of the reserved capacity, the results have shown that our architecture consumes backup capacity close to that consumed by the global recovery. However, in terms of the average notification time

and the average number of notification and activation messages our approach outperforms the global recovery approach and the one that sets up the backup paths between the branching points.

To My Late Mother

ACKNOWLEDGMENTS

The author would like to express his sincere appreciation and gratitude to his supervisors, Dr. Anjali Agarwal and Dr. J. William Atwood for their support and guidance over the last four years. They were extremely helpful during the course of this research work. I have benefited considerably by working under their supervision.

I gratefully acknowledge the financial support of Jordan University of Science and Technology.

I would like to thank the defense committee for offering their time to review my thesis.

I would like to thank my father, brothers and sisters for their deep love, encouragement, unbreakable support and the help they have given me through my life. Special thanks go to my twin Khalid and his wife for their help and encouragement especially in the last year when they joined me in Montreal. To Hashem Banimelhem, Khalid's son, may God bless you.

I am indebted to my friends in Montreal for their constant encouragement.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AS	Autonomous System
ATM	Asynchronous Transfer Mode
BDR	Border Router
BGMP	Border Gateway Multicast Protocol
BP	Backup path
BR	Branch Router
CBT	Core Based Tree
CL	Critical Link
CR	Critical Router
DVMRP	Distance-Vector Multicast Routing Protocol
ERM	Edge Router Multicasting
FEC	Forward Equivalence Class
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IR	Intermediate Router
TRPB	Truncated Reverse Path Broadcasting
ST	Steiner Trees
LAN	Local Area Network
LBP	LSP between Branching Points
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switching Path
LSR	Label Switching Router
MIGP	Multicast Interior Gateway protocol
MM	Multicast Management
MOSFP	Multicast Open Shortest Path First
MPLS	Multiprotocol Label Switching Protocol

MRT	Multicast Routing Table
NIMS	Network Information Manager System
OPSF	Open Shortest Path First
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PML	Path Merge LSR
PSL	Path Switch LSR
QoS	Quality of Service
RP	Rendezvous Point
RPB	Reverse Path Broadcasting
RPM	Reverse Path Multicasting
RSVP	Resource Reservation Protocol
RSVP-TE	RSVP Traffic Engineering
SBR	Segment between two BRs
SE	State Encoding
SP	Service Provider
TLV	Type-Length-Value
TN	Tree Numbering
TNT	Tree Node Table
TTL	Time To Live
WP	Working Path

List of Symbols

N^{i}_{egress}	The number of egress LERs downstream from router i
WDS_i	The partial weight value received from the downstream router connected directly to interface i
W_i	The weight value of interface i
u	The number of multicast trees that can be generated and shared by the same ingress LER
L_{ave}	The average number of labels
Z_{TNT}	The size of the TNT table
Z_{SE}	The size of the table needed to maintain all the codes of all the trees in the SE approach
Z_{TN}	The size of the table needed to maintain all the tree numbers in the TN approach
f	The number of interfaces at each router
$MPLS_{comp}$	The compatibility of any proposed approach with MPLS protocol
$\mathrm{d}^{\mathrm{root}}$	The root of the domain
$od(R)$	Outgoing degree of R
$\mathrm{T}^{\mathrm{root}}$	Root of tree T
N_i	Number of nodes which have a degree equal to i in tree T.
N_{LER}	Number of egress LERs in tree T.
ND	Number of domains.
N_{SBR}	Number of partitioning SBRs
$T^{notify}_{BDRi}(e)$	The time it takes until BDRi receives the notification message after the

	detection of the failure of element e.
T_{BDRi}^{active}	The time it takes until the backup path between BDRi and its PSL router is active.
S_f	The set of BDRs that are disconnected when element e fails.
C_t	The total backup capacity required to reserve the backup paths for tree T.
$N_{mess}(e)$	Number of routers that receive the notification or activation messages after the detection of the failure of element e.
N_{mess}^{ave}	The average number of routers that can receive the notification or activation messages
T_{detect}	The time needed to detect the failure.
T_{notify}	The time needed to notify the PSL router about the failure.
T_{switch_ov}	The time needed to switchover the data to the backup path by the PSL.
ζ_i	The ratio of the number of BRs to the number of BDRs in domain i.
N_{BR}^{i}	Number of BRs in domain i.
N_{BDR}^{i}	Number of BDRs in domain i.

Chapter 1

Introduction

1.1 Thesis Motivation and Objective

The increasing growth of the Internet has been facilitated by advances in networking and switching technologies, and has been accompanied by the appearance of real-time and multimedia applications. Some of these applications require group communication in which the same message sent by a source must be received by all group members. This type of communication is called multicast communication or simply multicasting. In recent years, and since it was introduced by Steve Deering [1, 2], IP multicasting has been given considerable attention, mainly because it enables many applications to scale. In other words, it services the users of these applications with less load on network and server resources.

At the same time, the Multiprotocol Label Switching (MPLS) protocol [3] has emerged as an Internet Engineering Task Force (IETF) framework that provides for the efficient designation, routing, forwarding, and switching of traffic flows through the

network [4]. The use of MPLS has resolved issues related to scalability and routing and it can exist over existing asynchronous transfer mode (ATM) and frame-relay networks. It is expected that MPLS will play a significant role in next generation networks. MPLS is considered a traffic engineering tool that has come up with useful solutions for unicast communication. For example, with MPLS it will be easier to overcome the congestion problem in current IP networks by efficiently utilizing the network resources.

However, despite these advantages offered to unicast communication, integrating MPLS into multicast networks is still accompanied by many problems and limitations. The solutions to these problems have to be proposed in order to integrate MPLS into backbone networks that support multicasting. This thesis proposes different solutions for two of these problems:

1. The first part of the thesis targets the scalability issue in multicast communication and how to assign the same MPLS label for different multicast sessions that deliver their IP packets over the same multicast tree.
2. In the second part of the thesis, we handle one of the most important issues in networking, that is, network survivability. We propose MPLS-based recovery architecture for multicast communication.

Although the thesis is divided into two parts, the integration of these parts is easily achieved and the solutions proposed in the two parts can be employed in the same network. In other words, the solution proposed in the first part can be employed alone if another MPLS-based recovery approached is used or they can be used with the MPLS-recovery architecture proposed in this thesis.

1.2 Contribution of the Thesis

This thesis introduces solutions for two areas of multicast communication in MPLS networks. Firstly, two schemes for efficient label assignment in MPLS networks are proposed. The main contributions in this aspect are the saving in the memory resources needed to maintain the FECs of multicast sessions at the edges of the MPLS network, and the reduction in the number of labels needed. Reducing the size of memory that store the tree information will speed up the processing time of the IP packets at the edges of the MPLS network. In addition, in our approaches, we have handled the dynamic behavior of a multicast session, which is one of the basic characteristics of multicast communications.

Another contribution is the proposing of the $MPLS_{comp}$ metric in order to indicate how much the proposed approaches in Chapter 3 are compatible with the MPLS protocol. In multicasting, it is a great advantage for any approach that supports MPLS multicasting to reduce the recourse to layer 3 to process the IP packets. As the value of $MPLS_{comp}$ increases, the switching-based forwarding of the IP multicast packets increases which means fast forwarding of these packets. The two approaches proposed in Chapter 3 are fully compatible with the MPLS protocol.

Secondly, an MPLS-based recovery architecture for multicast communication is proposed. The architecture is composed of a set of steps. One of these steps involves a novel tree division approach. The main contribution of the proposed architecture is that it handles both link and node failures in contrast with the other approaches that handle only link failure or the approach that cannot recover from all the node failures.

1.3 Organization of the Thesis

Chapter 2 and Chapter 3 form the first part of the thesis, and are dedicated to efficient label assignment and deployment of multicasting over MPLS networks.

- In Chapter 2, we present an overview of multicast communication and MPLS. We start by discussing multicast routing algorithms and protocols. Then we present an overview of MPLS. An introduction to the two problems that are handled in the thesis is given in Section 2.3.
- Chapter 3 of this thesis proposes two approaches to deploy multicast communication in MPLS networks. A performance analysis of the proposed approaches and a comparison with other approaches are presented.

Chapter 4 through Chapter 6 form the second part of the Thesis. We discuss the survivability in multicast communication based on MPLS and propose MPLS-based recovery architecture for multicast communication. These chapters are organized as follows:

- In Chapter 4, an introduction to MPLS-based failure recovery is presented. We discuss the main concepts and terminologies that are used in Chapter 5 and Chapter 6.
- In Chapter 5, we propose an MPLS-based recovery architecture for multicast communication. The proposed architecture basically depends on a suggested tree division approach that divides the multicast tree into several domains where each domain represents a sub-tree of the original tree. Two MPLS-based

recovery methods are then proposed to set up the backup paths inside the domains.

- Chapter 6 is dedicated to present the simulation results of the proposed recovery methods. The discussions of the results are reported in this chapter.

Finally, Chapter 7 concludes the thesis and introduces guidelines for future work.

Chapter 2

Background

This chapter is considered as the background for the remainder of the thesis. Section 2.1 presents an overview of IP multicasting. We discuss the advantage of multicast communication over unicast and broadcast communications. Then, we discuss multicast algorithms and protocols. Section 2.2 gives an overview of MPLS. We introduce the operation of MPLS, and discuss the Forward Equivalence Class (FEC) concept and the advantages of MPLS. In Section 2.3, we give an introduction to the two problems that are handled in this thesis.

2.1 Overview of IP Multicast Communication

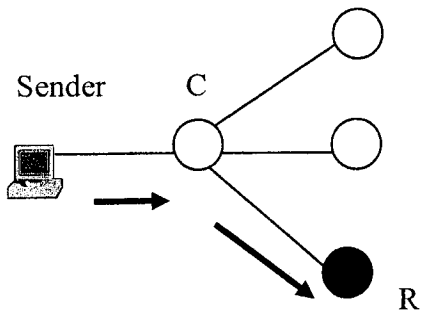
Multicast communication is defined as the ability of the network to send the data from one to many recipients, or from many to many recipients. Many applications rely on underlying multicast network services for delivering data between users and these applications are called multicast applications. Examples of multicast applications are: audio-video broadcast as an example of point to multipoint multicasting and video conferencing as an example of multipoint-to-multipoint multicasting. More details about

multicast applications are in [5, 6, 7, 8]. In order to address the main advantage of multicasting, we present the three types of communication generally used between computer systems:

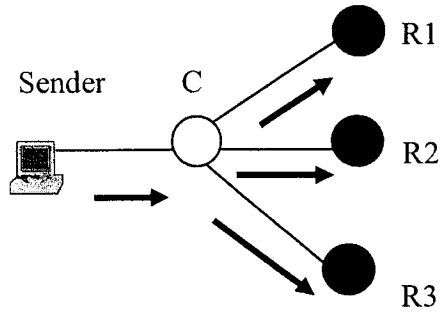
1. Unicast transmission where there are only one sender and one receiver.
2. Broadcast transmission where a sender sends the message to an entire network.
3. Multicast transmission where the message is sent to a set of hosts on the network.

Figure 2.1 shows the three types of communication.

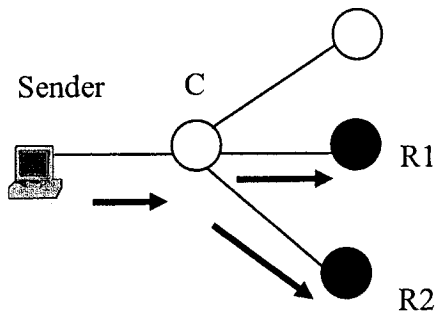
Multicasting can be implemented by using unicast communication where the multicast packet is sent to each member in the group using unicast addresses. However, this approach is expensive and inefficient in terms of bandwidth usage. For example, to send the multicast packet in Figure 2.1(c) to receivers R1 and R2 using unicast communication, the IP packet should be sent twice over the link that connects the sender and the router C as shown in Figure 2.2(a). On the other hand, multicasting can also be implemented using broadcasting where the multicast message is sent to all nodes in the network using a broadcast address, then any node that is not a member in the multicast group discards the packet. This approach is also inefficient such that some data packets will be flooded in the network and there is unnecessary processing of the packets on the machines of the non-members of the multicast groups. For instance, Figure 2.2(b) shows the implementation of multicasting using broadcast communication. In addition to unnecessary processing of the multicast packets at router A, part of the bandwidth of the link that connects the router C with router A is wasted. So it can be said that the main advantage of multicasting is that it reduces network traffic by efficiently utilizing the link



(a) Unicast Communication

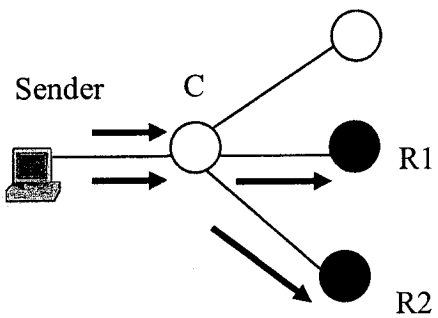


(b) Broadcast Communication

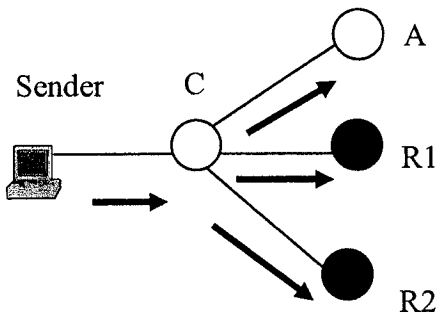


(c) Multicast Communication

Figure 2.1 Types of communications.



(a)



(b)

Figure 2.2 Implementation of Multicasting using (a) unicast communication and (b) broadcast communication.

bandwidth.

Multicast groups can be classified in different categories. They can be permanent or transient groups. The permanent groups remain in existence even if there is no member in the group where the transient groups remain in existence as long as there are members in the group. Moreover, multicast groups can be classified based on the distribution of the group members in the network. In this classification, multicast groups can be either dense or sparse groups.

Multicast communication in the Internet is implemented by employing three types of protocols:

1. Internet Group Management Protocol (IGMP) [9], which is responsible for maintaining the membership of the hosts that join and leave a multicast group.
2. Multicast Interior Gateway protocol (MIGP). This protocol is employed by multicast routers to enable multicast communication within the same Autonomous System (AS). Examples of MIGP are DVMRP, MOSPF and PIM-SM.
3. Border Gateway Multicast Protocol (BGMP), which is used by the border routers to enable multicast communication across ASs.

2.1.1 Multicast Routing Algorithms

In unicast communication, routing is done by finding the shortest path between the sender and the receiver and the packet is forwarded based on the destination IP address. However, the matter is different with multicast communication where a group of more than two nodes communicate with each other. In multicast communication, establishing a

tree that spans all the nodes in the multicast group is the first step to do the routing and packet forwarding may be based on both the source and group IP addresses.

The underlying routing algorithm determines the delivery tree for each multicast group. Multicast routing algorithms can be summarized as the following:

Flooding In this algorithm, when a multicast router receives the multicast packet, it checks if this packet has been received before or not. If the router has not received the packet earlier, the packet will be forwarded to each interface other than the one on which it has arrived. In the case where the packet was received before, it will be discarded. The main advantage of the flooding algorithm is its simplicity. The router does not have to maintain a routing table. It only needs to keep track of the recently seen packets. However, the flooding algorithm has the disadvantages that it can not scale widely over the Internet and it uses all the available paths in the network.

Spanning Tree This algorithm is an effective solution for the flooding algorithm. It creates a delivery tree such that there is only one active path between any two routers. A multicast packet will be forwarded to each interface belonging to the spanning tree except the one on which the packet was arrived. The spanning tree is considered a non group-specific algorithm. One of its advantages is that it guarantees to build a tree without loops in the network. However, the main disadvantage of this algorithm is that it may cause traffic concentration on relatively small number of available links.

Reverse Path Broadcasting (RPB) RPB creates a group-specific delivery tree for each source in the multicast group. In RPB, a router R forwards the multicast packet received from a source S if this packet arrives via the shortest path from the router R back to the

source S (i.e, the reverse path). Router R broadcasts the packet to all links except the one on which the packet arrived. The RPB algorithm can be enhanced to prevent unnecessary packet duplication. This could be done if the router making the forwarding decision could determine if the neighboring router on the outgoing link considers the local router to be on the shortest path back to the source. If this condition is achieved, the packet is forwarded to the neighbor, otherwise packet duplication is prevented. The main advantages of RPB algorithms are: it is efficient and easy to implement. Also, the router is not required to have knowledge about the entire spanning tree. However, one of RPB's limitations is that it does not take into account multicast group membership when building the delivery tree for a (source, group) pair.

Truncated Reverse Path Broadcasting (TRPB) TRPB is an extension to the RPB algorithm. It overcomes the weaknesses of the RPB algorithm by eliminating unnecessary IP multicast traffic. It achieves that by truncating the spanning tree if a leaf subnetwork does not have any group members. However, it still does not consider group memberships when building the branches of the delivery tree. Thus traffic is still forwarded to intermediate routers on the branches of the delivery tree toward leaf subnetwork with no group membership.

Reverse Path Multicasting (RPM) RPM is a further enhancement of the RPB and TRPB algorithms by taking into account the group memberships when constructing the branches of the delivery tree. The new feature introduced in RPM and not found in the previous two algorithms is the pruning mechanism. RPM allows the delivery tree to be pruned so that packets are forwarded to branches leading to members of the destination group. When an RPM router receives a packet for a (source, group) pair, the first packet

is forwarded using the TRPB algorithm to all routers in the internetwork. TRPB is used to guarantee that each router on a leaf subnetwork will receive the first packet. If there is a group member on one of its subnetworks, a router forwards the packet based on its IGMP information. If there are no group members, the leaf router sends a prune message on its parent link. Prune messages are sent only one hop toward the source of the delivery tree. If an upstream router receives a prune message from each of its child is interfaces, the router generates a prune message for the (source, group) pair and sends this message to its parent link.

Steiner Trees (ST) A Steiner tree is a delivery tree that spans all the members in a group with the minimal total number of links. The main idea of ST is to build a delivery tree that minimizes the use of network resources. The disadvantages of the ST algorithm are that it is NP-complete and it is unstable in cases where there are dynamic changes in group memberships.

Core Based Trees (CBT) In this algorithm, one of each multicast group members is selected as the core or as a Rendezvous Point (RP) for the group. A tree is then constructed such that the core is acting as the root of the tree. All members of the group will share this tree to deliver their packets among each other. The CBT algorithm is quite similar to the spanning tree algorithm except it allows a different core-based tree for each group.

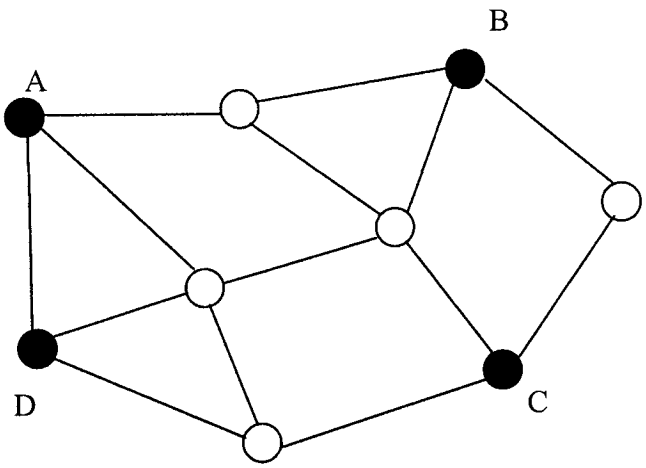
Multicast routing algorithms and protocols are discussed in more detail in [10, 11].

2.1.2 Multicast Routing Protocols

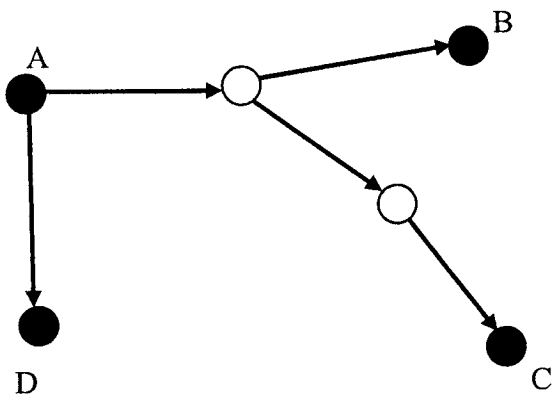
Generally, multicast trees are classified into two types:

1. Source-specific trees (also know as shortest path trees) build a separate tree for each source. Source-specific tree is efficient for high data rate sources. It provides minimal delay at the expense of cost. DVMRP, PIM-DM and MOSPF are example of multicast protocols that build source-specific tree.
2. Shared trees. In this type, a single tree is built and used by all the sources. The data communication in the tree can be one way or bi-directional. The shared tree is efficient for low rate sources and is efficient in the amount of state information that needs to be maintained at each router. However, the drawback of the shared tree is that it exhibits higher traffic concentrations. Shared trees use a single location in the network called the core or the Rendezvous Point (RP). The sources send their packets to the RP, which then forwards these packets to the receivers. CBT is a multicast protocol that establishes a shared tree to deliver the multicast packets. PIM-SM is a multicast protocol that allows switching of the receivers from the shared tree to the Source-specific tree.

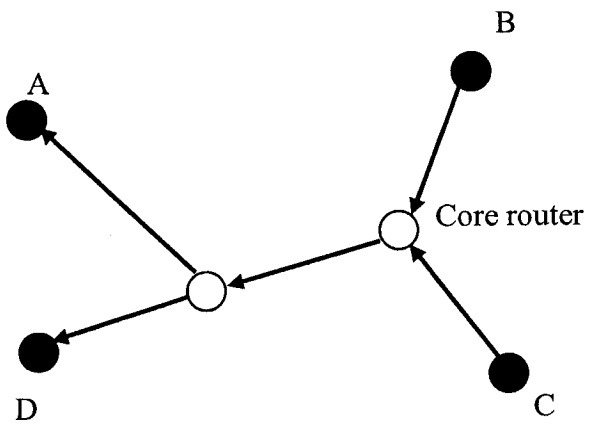
Figure 2.3(a) shows an example of a network with a multicast group composed of four members. Figure 2.3(b) shows the source-specific tree where node A is the source. Figure 2.3(c) shows the same group when the shared tree is built and one node in the network is selected as the core node. In Figure 2.3(c), nodes B and C are the sources in the group while nodes A and D are the receivers. The rest of this section introduces the basic operation of the multicast protocols.



(a) Network topology.



(b) Source-specific tree with A as a source.



(c) Group-shared tree of the group (A, B, C, D)

Figure 2.3 An example of a network with a multicast group session (A, B, C, D).

Distance-Vector Multicast Routing Protocol (DVMRP) DVMRP [12] is an implementation of the TRPB algorithm. It is the first protocol developed to support multicast routing. It constructs a different distribution tree for each source and its destination host group. Each distribution tree is a minimum spanning tree from the multicast source at the root of the tree to all the multicast receivers as leaves of the tree. The distribution tree provides a shortest path between the source and each multicast receiver in the group, based on the number of hops in the path, which is the DVMRP metric. A tree is constructed on demand, using a "broadcast and prune" technique, when a source begins to transmit messages to a multicast group.

Multicast Open Shortest Path First (MOSPF) MOSFP [13, 14] is an extension of the OSPF [15] unicast routing protocol. So it is a link-state protocol that uses OSPF to get the information about the network topology. MOSFP extends the OSPF protocol by providing the ability to route multicast IP traffic. A router in MOSPF uses IGMP to keep track of group membership information on its attached networks.

Protocol Independent Multicast (PIM) PIM is a multicast routing protocol that runs over an existing unicast infrastructure. PIM is different from the other protocols in two aspects:

1 It can operate in two modes: dense mode (PIM-DM [16]) and sparse mode (PIM-SM [17]).

2 It uses an explicit join model for sparse groups. Joining occurs on a shared tree and can switch to a per-source tree.

PIM-DM employs the same flood-and-prune mechanism that DVMRP and other dense mode routing protocols use. The main difference between DVMRP and PIM-DM is that PIM-DM is an independent protocol that can use the routing table populated by any underlying unicast routing protocol such as OSPF or IS-IS protocols to perform reverse path forwarding (RPF) checks. PIM-DM is employed in an environment where group members are densely distributed and bandwidth is plentiful.

PIM-SM In PIM-SM, group members are assumed to be located far away from each other. In addition to that, it is assumed that the bandwidth is not plentiful. PIM-SM uses per-group Rendezvous Points (RPs). An RP is a router in the PIM-SM domain that is used by senders to announce their existence, and by receivers to join the group.

Core Based Tree (CBT) The CBT architecture [18, 19] is introduced to overcome the shortcomings of DVMRP. It builds a shared tree that spans all group members and has a single node known as the core of the tree. The branches of the tree emanate from the core node and pass other routers called on-tree routers, which form a shortest path between a host's directly attached router and the core router. When a host wants to join a group, it expresses its interest in joining a group by contacting its local router which in its turn sends a join-request message to the next hop on the shortest path toward the core. The join-request message travels hop by hop until it reaches the core or an on-tree router. At that point, the core or the on-tree router sends a join-acknowledge message back along the reverse path. Each router on the path updates its forwarding cache to indicate that it now becomes an on-tree router and then forwards the join-acknowledge message back to the requesting router.

2.2 Overview of MPLS

MPLS has addressed the main problems faced by current IP-based backbone networks such as speed, scalability, quality-of-service (QoS) management, and traffic engineering. In this section, we give an overview of the main features of MPLS and its application and advantages.

2.2.1 MPLS Operation

The basic goal behind introducing MPLS approach is to speed up the process of packet forwarding without making any changes to the existing IP routing protocols. The idea of MPLS depends on assigning short, fixed length labels to the packets at the ingress node of the network. Inside the MPLS network, the MPLS capable routers, called Label Switching Routers (LSRs), only examine the label instead of the IP header.

The operation of MPLS is depicted in Figure 2.4. When an IP packet enters the MPLS domain, an ingress router called Label Edge Router (LER), LER A, pushes label L1 to the packet based on its Forward Equivalence Class (FEC) and then forwards the packet into the MPLS domain. During its way to the destination, LSR B and LSR C in the MPLS domain forward the packet based on a mapping process between the (incoming interface, incoming label) and (outgoing interface, outgoing label). LSR B performs the swapping process by replacing label L2 by L1 and then forwards the packet to LSR C, which does the same swapping process between L2 and L3. At the other edge of the MPLS domain, another router called egress LER, LER D, removes label L3 from the packet and forwards the packet based on its IP address. The path between the ingress

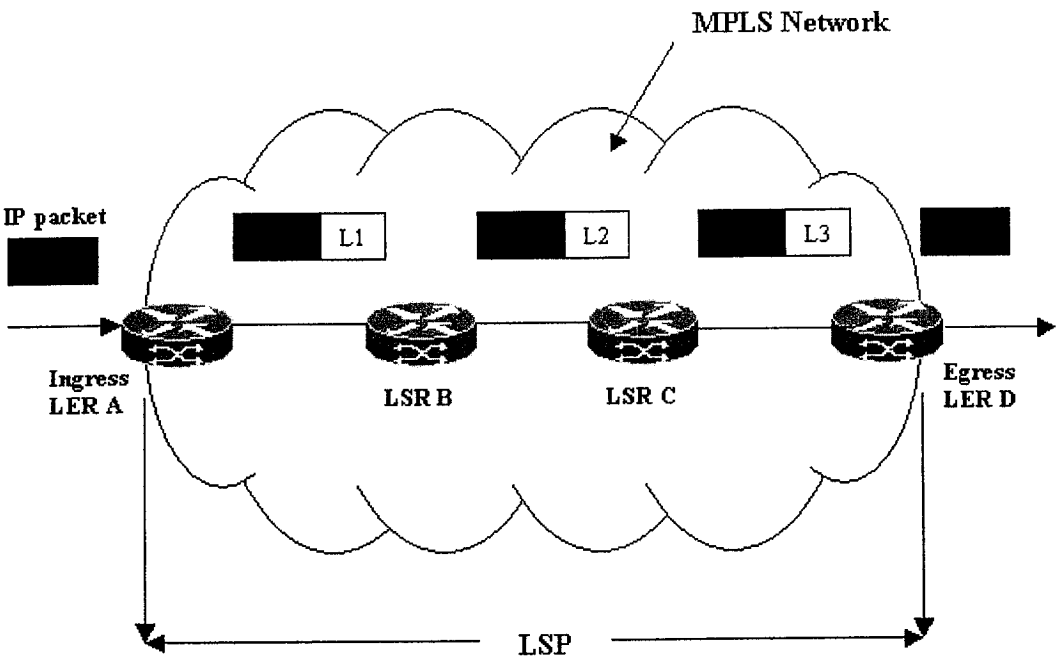


Figure 2.4 Operation of MPLS.

LER and the egress LER is called a Label Switching Path (LSP). The LSP is set up using signaling protocols such as the Label Distribution Protocol (LDP) and its extensions to support Constraint-based Routing (CR-LDP) [20], or a Traffic Engineering extension of the Resource Reservation Protocol (RSVP-TE) [21].

The generic format of the MPLS header is shown in Figure 2.5. It fits between the link layer header and the network layer header. The label field (20 bits) carries the actual value of the MPLS label. The detailed explanation of the label field and the other fields is given in [3, 4, 22].

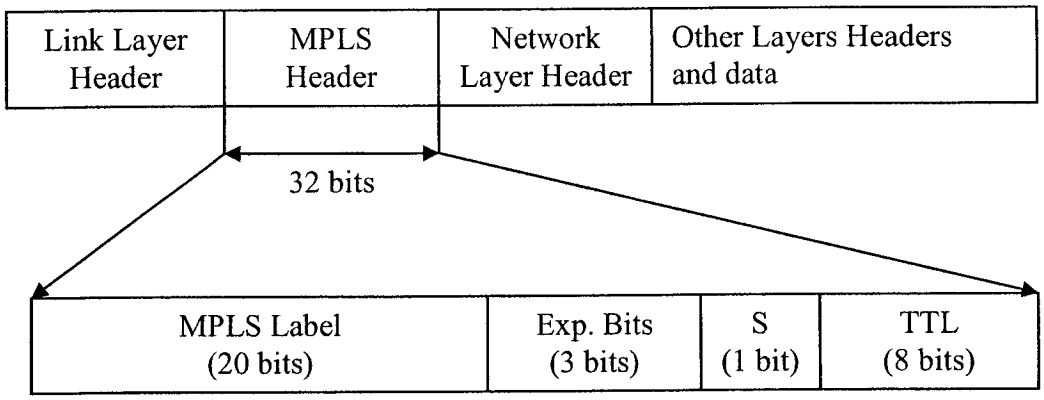


Figure 2.5 MPLS header format.

2.2.2 FEC

The Forward Equivalence Class (FEC) is a representation of a group of packets that share the same requirements for their transport [4]. All the packets that belong to the same FEC are provided the same treatment during their journey from the ingress LER to the egress LER in the MPLS domain. FEC is the basic feature of MPLS that offers the network administrators the capability to control the packet's path and the forwarding treatment it will receive along that path in the MPLS network [22]. Unlike traditional IP forwarding, which depends basically on the IP destination address, the assigned FEC takes into account many factors such as: the packet's application protocol, the IP source and destination addresses of the packet, and QoS requirements.

2.2.3 Applications and Advantages of MPLS

By transferring the IP backbone networks from connectionless mode into connection-oriented mode, MPLS accomplishes the following:

1. The performance of packet forwarding is improved in the MPLS network. This advantage is achieved by enhancing and simplifying the packet forwarding through using layer 2 switching paradigms.
2. The avoidance of traffic congestion in current IP networks is easily handled with MPLS by using traffic engineering process [23, 24, 25]. With MPLS, the overall network utilization is enhanced by attempting to create a uniform or differentiated distribution of traffic throughout the network.
3. MPLS has the capability to support QoS and Class of Service (CoS) for service differentiation.
4. One of the great advantages of MPLS is the integration between IP and ATM networks. By using the existing router/ ATM switch hardware, MPLS provides a bridge between access IP and core ATM networks.

2.3 Definition of the Problems

In this section, we introduce the two problems that are handled in this thesis. We start with state scalability problem in multicast communication and then we give the importance of providing a reliable multicast communication when a link/node fails in the multicast tree.

(S, G)	*I*	*O1*	*O2*	.....	*On*

Figure 2.6 The general format of a multicast entry in the MRT.

2.3.1 State Scalability Problem in Multicast Communication

In multicast communication when a new session is created in a network, a multicast tree is built to connect the group members of that session together. Each node in that tree should keep a multicast state entry for that session in the MRT (Multicast Routing Table). The general format of a multicast entry in the MRT is shown in Figure 2.6. In that figure, S represents the IP address of the source and G is the IP address of the multicast group. *I* is the incoming interface where the IP packets of session (S, G) will be received and *O1* through *On* are the outgoing interfaces where each IP packet of session (S, G) will be duplicated and sent out.

The number of entries in the MRT increases linearly with the number of multicast groups in the network. This means more memory requirement and slower forwarding process since each packet forwarding involves an address look-up in the MRT. MPLS overcome the speed problem by replacing the current forwarding process by a switching-based method. This is achieved by mapping the MRT at each router into a corresponding forwarding switching table. Each entry in the MRT is mapped into its corresponding entry in the forwarding switching table. Figure 2.7 shows the general format of the entry in Figure 2.6 when it is mapped into its corresponding entry in the forwarding switching

(S, G)	*I*	*O1*	*O2*	.....	*On*
	IL	*OL1*	*OL2*	.....	*OLn*

Figure 2.7 The general format of a multicast entry in the forwarding switching table.

table. *IL* is the label attached to the IP packet when it is received at interface *I*. and *OL1* through *OLn* are the MPLS labels that will be pushed to the IP packets that are send out over the outgoing interface *O1* through *On*.

Although MPLS reduces the processing time of the IP packet, multicast communication still suffers from the scalability problem when the number of sessions is very large. Further improvement of the processing time can be achieved if the size of the forwarding switching table is reduced. In MPLS, this objective can be achieved if two or more multicast sessions share the same entry in the forwarding switching table and then the same labels can be used for different sessions. This concept is called label aggregation.

In the next chapter, we explain the scalability problem in more detail and then propose two approaches that enable two or more multicast sessions, which deliver their IP packets over the same tree, to use the same MPLS labels.

2.3.2 Failure Recovery in Multicast Communication

The problem of providing reliable services in multicast communication has been gaining importance especially after the appearance of real-time applications. Providing

failure recovery approaches in multicast communication is more demanding than unicast because one link or node failure affects many receivers of the same group. Although, several MPLS-based failure recovery approaches have been proposed for unicast communication, relatively little research has been done that targets MPLS-based recovery for multicast communication. In chapter 5, we propose an MPLS-based failure recovery architecture for multicast communication.

2.4 Summary

We have introduced IP multicast communication and the MPLS protocol. An introduction to the two problems that are handled in this thesis is given in this chapter. In the next chapter, we discuss the problems and limitations of deploying multicasting over MPLS networks. Then, we propose two schemes that introduce efficient solutions to the scalability problem in multicasting.

Chapter 3

Deploying Multicast Communication over MPLS Networks

3.1 Introduction

Unlike unicast communication, multicasting still encounters some problems and limitations in MPLS networks. Some of these limitations are discussed in [26, 27] and itemized here briefly:

- The limitations on layer 2 technologies. As mentioned before, MPLS is used to speed up the processing time of the IP packet. It achieves this by replacing the IP forwarding by a simple label lookup. This is done by mapping the label onto various existing layer 2 switching capabilities, such as ATM and Frame Relay. However, such mapping on these technologies poses several limitations such as: limited label space, no merging capability and Time To Live (TTL) decrement.
- A key point of MPLS is that multiple flows may be assigned to the same label. This process is called aggregation [22, 28]. The main advantage of aggregation is

that it reduces the number of labels needed to handle a set of flows. Aggregation is an important issue for IP multicast in MPLS domain. This issue is related to the granularity of multicast traffic, which is (*, G) for a shared tree and (S, G) for a source tree.

- Flood and Prune issue. Some IP multicast routing protocols, such as DVMRP [12] and PIM-DM [16], flood the network with multicast data, and then the branches that do not want to receive the data of the specific multicast group are pruned. Although flooding is the simplest multicast algorithm, it is extremely inefficient because its principle is similar to broadcasting. The process of flooding and pruning is repeated periodically, thus generating a very volatile tree structure. This dynamic behavior resulted from flooding and pruning process makes also the mapping from a layer 3 point-to-multipoint tree to a layer 2 point-to-multipoint tree dynamic. This makes deploying multicasting in an MPLS network problematic because of the limited label space, the signaling overhead, and the time needed to set up the LSPs.
- Source/Shared Tree. This is a scaling issue, which may affect the label space in multicasting. As was discussed before, IP multicast trees are classified into source trees where a tree per source (S) and per multicast group (G) is created, or shared trees where one tree per multicast group is established. The advantage of using shared trees in MPLS is that they consume fewer labels than source trees.
- Coexistence of source and shared trees. Some IP multicast routing protocols, like PIM-SM, support both source and shared trees. In such a protocol, some routers can maintain both (*, G) and (S, G) states for the same group G.

Most of the above points are discussed in [29, 30]. This chapter handles the aggregation and scalability issue when the number of IP multicast sessions increases within an MPLS network. In general, multicast sessions can be classified into different FECs based on (S, G) information for each session, where S is the IP address of the source and G is the IP address of the multicast group. However, this classification does not provide a solution for the scalability issue and it does not allow aggregation of different sessions having the same multicast MPLS tree. The reason is that an IP multicast group address does not represent a specific host or a network prefix as it does for an IP unicast address. This makes it impossible to know if two different sessions would have the same tree over the MPLS network. Depending on the IP multicast address to classify the multicast sessions into FECs is not a scalable solution. In this chapter, we propose two schemes to overcome the scalability issue in multicasting over MPLS networks.

3.2 Motivation

In order to introduce the motivation of our approaches, Figure 3.1 shows a simple MPLS network with ingress LER E0 and three egress LERs E1, E2, and E3. If a hundred different multicast sessions pass through the network from the ingress LER E0, both LSR1 and the ingress LER E0 will maintain a forwarding table that contains the hundred entries. Each entry will consume a different MPLS label. However, the number of multicast trees that could be built over the network is seven. The trees can be identified

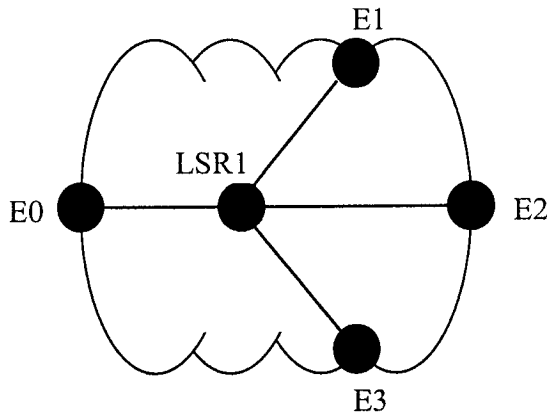


Figure 3.1: A simple MPLS network.

by the LERs as: {E0, (E1)}, {E0, (E2)}, {E0, (E3)}, {E0, (E1, E2)}, {E0, (E1, E3)}, {E0, (E2, E3)}, and {E0, (E1, E2, E3)} where the first term represents the ingress LER and the second term represents the egress LERs of a certain tree. If each tree in the network is distinguished from the other trees, then all multicast sessions having the same tree structure will be mapped to the same number (FEC) and only one MPLS label at E0 and LSR1 will be maintained for each tree. The same label then is used to forward the IP packets of the different multicast sessions (flows) delivered on the same tree. For the network shown in Figure 3.1, only seven MPLS labels are needed at both E0 and LSR1.

In the next two sections, we propose two approaches to deploy multicasting in MPLS networks. The first scheme is called State Encoding (SE) while the second one is called Tree Numbering (TN).

3.3 State Encoding (SE) Scheme

Assume we have an MPLS network with a number of LERs. Several multicast trees could be built for each ingress LER inside the MPLS network. Each tree has its structure which is different from the structures of the other trees. The assumed MPLS network is a backbone one that is either connected to other access networks through edge routers or it could be connected directly to local area networks (LANs) at some LERs as shown in Figure 3.2. The numbers shown in the figure represent the interfaces numbers.

3.3.1 SE Concept

The idea in this approach is to encode the state of each router that a multicast tree passes through [31]. Consider Figure 3.3, which shows a router with one incoming interface (d) and three outgoing interfaces (a, b, c). The arrows indicate all possible trees that could pass the router from interface (d) as a parent link and continue through interfaces (a, b, c) as possible child links in the tree. When a multicast session (S, G) builds a tree in the network and the tree crosses the router in Figure 3.3 from interface (d) as incoming interface, the state of the router can take one of seven states depending on which outgoing interface is active.

To encode the possible combinations of the active interfaces, we need seven codes as shown in the table in Figure 3.3.

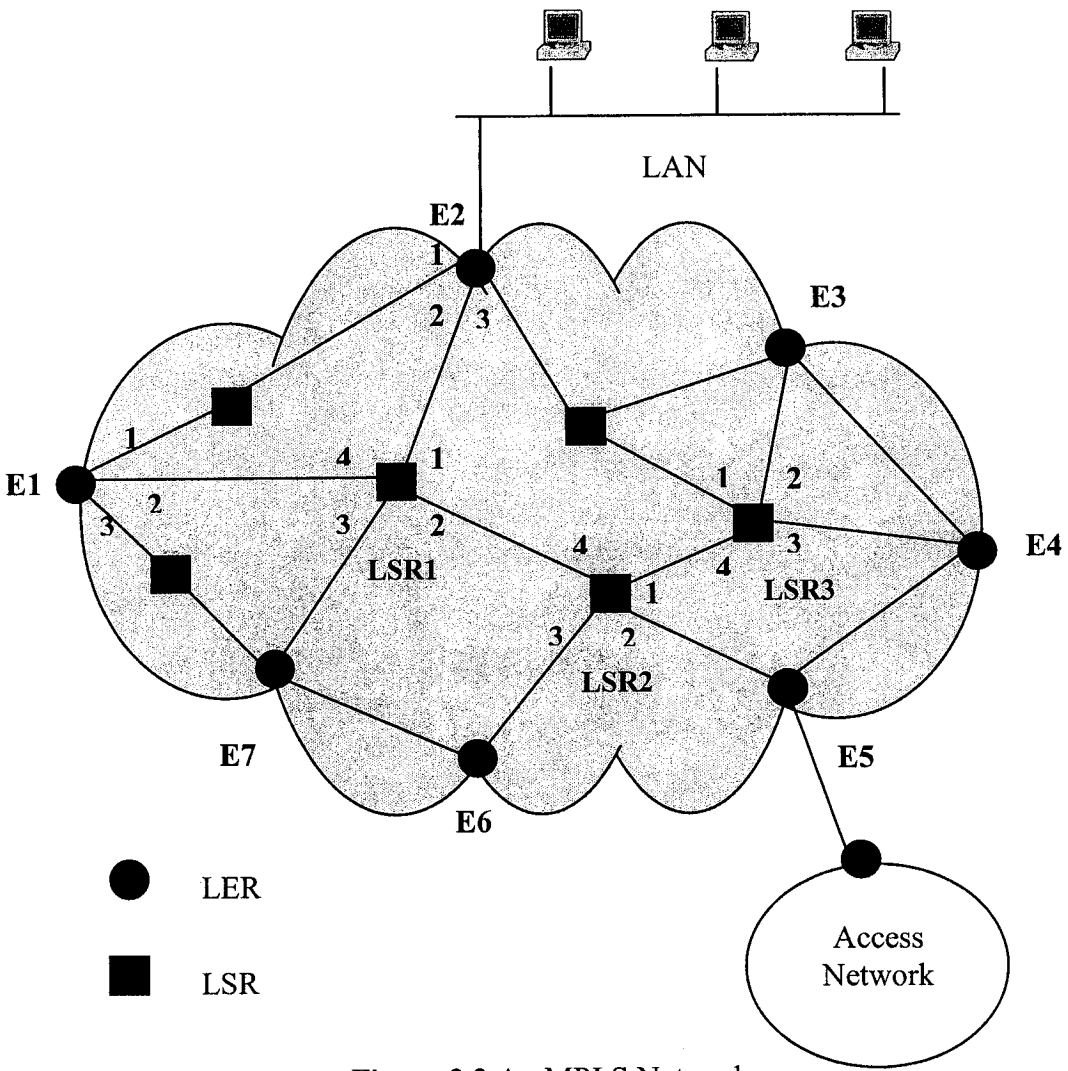
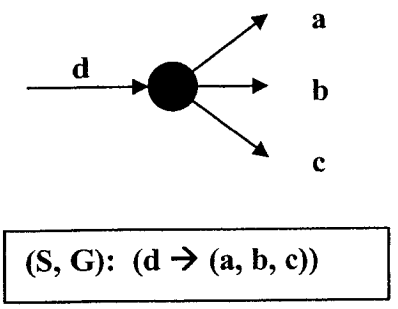


Figure 3.2 An MPLS Network.



c	b	a	*State Code*
0	0	1	C1
0	1	0	C2
0	1	1	C3
1	0	0	C4
1	0	1	C5
1	1	0	C6
1	1	1	C7

Figure 3.3 A state code table for a router with an incoming interface (d) and three possible outgoing interfaces (a, b, c).

When the corresponding interface is set to 1, this means that a multicast tree passes this router from the incoming interface and continues to another router thought that interface. For instance, C3 is a code for all multicast trees that pass the router from interface (d) as incoming interface and continue to two other routers using the outgoing interfaces (a) and (b). Note that state (0, 0, 0) is not included in the table which means that the corresponding tree does not cross the router.

State codes on all the corresponding routers of a multicast tree will be sent back to the ingress router of that tree. For this purpose, a path state is assumed in each router of the tree. One of the following two options can be used in order to maintain a path state in each router over a multicast tree:

1. Resource Reservation Protocol, RSVP [32], could be used for that purpose. In this case, a new object should be defined to collect the codes of the corresponding tree from each LSR and send the collected codes back to the ingress LER.

2. The second option is to define a specific protocol. The protocol is composed of a set of messages. One of these messages should be used to maintain the path state. Another message is defined to collect the tree code. If a new protocol is defined, it should use the multicast state entries already generated by the multicast protocol that is employed in the network.

The choice between these two options depends on the backbone network itself. If QoS were used in the network, the use of the RSVP would be preferable because the state codes can be collected by defining a new object, which can be sent back to the ingress LER using the same message used to reserve the path. In this case, the advantage of using RSVP is that there is no need to define new messages. However, the disadvantage of using RSVP in this case is that an overhead time will be added at each LSR if the message that contains the state codes includes other objects. Option 2 should be used in the case that the backbone network does not use RSVP. The main advantage of using option 2 is its simplicity. In that option, the main messages that should be defined are the one that maintains the path state and the one that collects the state codes.

Code generation starts at the LSRs that are directly connected to the egress LERs. At each LSR, no code will be sent upstream by that router until all the codes from its child routers are received. The ingress LER is the last router that concatenates its code to the received codes. During the exchange of the sub-codes between the routers, the following assumptions are made:

1. There is a mechanism that defines the orders of the concatenations of the received sub-codes.

Table 3.1: A pseudo code of the SE approach.

```
1   // In: Incoming interface for session (S, G) at router R.
2   // O1, O2, ..., Om : Outgoing interfaces for session (S, G) at router R.
3   CurrentCode = { }; // This is an initialization for the CurrentCode
4   // Session (S, G) has a state entry as: (S, G): In → (O1, O2, ..., Om)
5   for i =1 to m do        // m is the number of outgoing interfaces
6       begin
7       Receive sub-code[i] from the router connected to interface i;
8       If sub-code[i] is not received correctly then
9           begin
10          Ask the router connected to interface i to re-send sub-code[i];
11          goto 7;
12          end
13      CurrentCode ← concatenate sub-code[i] to the CurrentCode;
14      end
15  Code_R ← Generate the sub-code for (S, G) using the state code table;
16  // Code_R is the code generated by router R for session (S, G)
17  Code_R ← concatenate Code_R to the CurrentCode;
18  Send Code_R upstream through interface In.
```

2. There is a protocol of message exchange between any router and its child routers to make sure that the sub-codes are sent and received correctly.

At the ingress router, the codes are used as information to classify the IP packets of the multicast sessions into their appropriate FECs. If two or more multicast sessions share the same tree and have the same tree code, they will be mapped to the same FEC. The pseudo code of the SE approach is shown in Table 3.1.

3.3.2 Illustrative Example

To illustrate SE in an example, consider Figure 3.2 after removing the LANs and the access networks. Assume five multicast sessions are established in the MPLS network from the ingress router E1 and cross the network from one or more egress LERs as given in Table 3.2. Figure 3.4 shows the multicast trees that are built for each session. The state code table of the ingress router E1 as shown in Figure 3.4(d) is the same as the state tables maintained also at LSR1, LSR2 and LSR3.

As it is shown, session 2 and session 3 have the same tree. If the MPLS multicast tree is configured for each session, two labels will be used in the switching table at each router for the two sessions even though they have the same "tree structure (i.e., having the same ingress node and the same set of egress nodes)". If the ingress LER has the information that indicates that the two sessions have the same tree structure, then the multicast data packets from the two sessions will be assigned the same label and a saving in label usage is achieved.

The code generated for the multicast tree in Figure 3.4 (b) is as follows: at LSR3, the code is 010 because the multicast tree does not cross interfaces 1 and 3; their corresponding bits are set to 0. Only interface 2 is crossed by that tree, so its corresponding bit is set to 1. The other codes for the same tree generated at LSR2, LSR1, and E1 are: 001, 110, and 010, respectively. Table 3.3 shows the state codes of each tree at all the LSRs and the ingress LER. This table represents a tree code table (TCT) at the ingress router that maps each multicast session into its code and then into its FEC. For instance, session 4 (S4, G4) has a tree code = (010110111110).

Table 3.2 Five multicast sessions pass the MPLS network shown in Figure 3.2.

Multicast Session	Egress LERs
Session 1: (S1, G1)	E5
Session 2: (S2, G2)	E3 and E7
Session 3: (S3, G3)	E3 and E7
Session 4: (S4, G4)	E3 through E7
Session 5: (S5, G5)	E4, E5 and E7

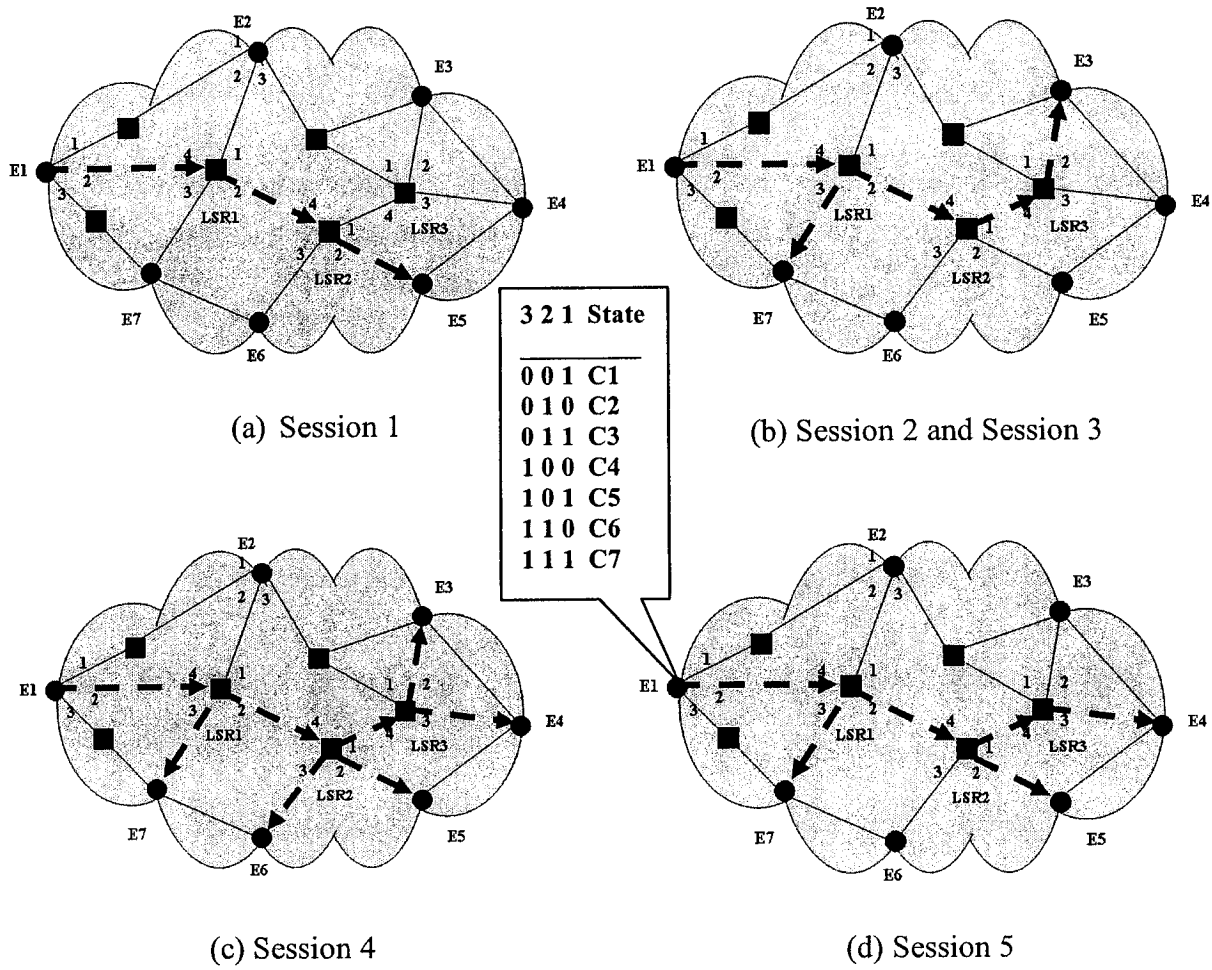
We assume that the generated sub-codes at all the routers are equal. The size of any sub-code should be equal to $MaxCode$, where $MaxCode$ is the maximum number of interfaces for a certain router in the MPLS network. If the number of interfaces at router i is less than $MaxCode$, that router should append zero bits to the left of the generated code. The number of these bits is equal to $MaxCode - SizeCode_i$, where $SizeCode_i$ is the size of the code generated from the state table at router i.

3.3.3 Tree Maintenance

When a current member of a multicast group leaves or a new member joins the multicast session (S, G), the code of the multicast tree of that session is recalculated. The recalculation of the tree code is triggered by the first router that receives the **Join-Request** or the **Prune-Request** message. That router generates a **Code-Request** message for the tree built by session (S, G). The **Code-Request** message should be received by all the routers on the corresponding tree. The recalculation also starts from the LSRs that are directly connected to the egress LERs and the process proceeds as discussed before.

Table 3.3 State codes of the five sessions at the ingress LER E1 and the LSRs LSR1 through LSR3

Session	State code at			
	E1	LSR1	LSR2	LSR3
	321	321	321	321
(S1,G1)	010	010	010	---
(S2,G2)	010	110	001	010
(S3,G3)	010	110	001	010
(S4,G4)	010	110	111	110
(S5,G5)	010	110	011	100



(a) Session 1

(b) Session 2 and Session 3

(c) Session 4

(d) Session 5

Figure 3.4 The multicast trees that are built for the sessions in Table 3.2.

If the rate of joining or leaving a multicast tree is high, the rate of generating the **Code-Request** messages will be high too. In the backbone networks, multicast sessions, such as video/audio conferencing, the number of participants in such sessions will be stable most of the life time of the session. However, the use of the **Code-Request** message can be replaced by the use of the **Update-Code** message. That message is generated when a router receives a **Join-Request** or a **Prune-Request** message. In this case, each router should keep a copy of the received sub-codes from all the outgoing interfaces for all the multicast sessions. When a router receives a **Join-Request** or a **Prune-Request** message for multicast session (S, G), the code of the tree that is built by that session is updated starting from the first router that has received the **Join-Request** or the **Prune-Request** message. That router generates the **Update-Code** message, which will be forwarded upstream until it is received by the ingress LER. The solution of using the **Update-Code** message instead of regenerating the tree code by all the routers adds extra cost in terms of using the memory resources at each router.

3.3.4 Uniqueness of the Tree Code

The uniqueness of each code generated by a certain tree is guaranteed from the fact that any router will not generate the same code for two different trees even though they might have the same tree structure. In Figure 3.5, for instance, two different trees T1 and T2 are generated and both of them are rooted at the ingress LER E1. Tree T1 has a code **baCx** and tree T2 has a code **baCy**, where Cx and Cy are the concatenated codes generated at routers RX and RY, respectively. Even if Cx is equal to Cy, the code generated by E1, which is appended to these as **(ba)** will distinguish them. The final

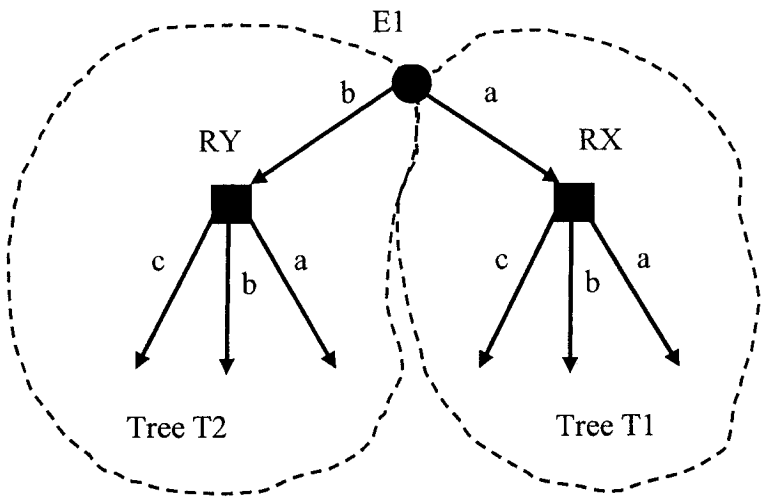


Figure 3.5 The code generated at E1 distinguishes any two trees.

codes then can not be equal at any case because the code generated for T1 is (01Cx) and the code generated for T2 is (10Cy). This justification is applied to any router in the network that contributes to the calculation of codes of the trees.

3.4 Tree Numbering (TN) Scheme

The code size that is generated for each tree by SE approach depends on the number of routers of that tree. This motivates us to propose another approach that generates a fixed size of the information to encode the tree information. This approach is called Tree Numbering approach (TN).

3.4.1 TN Concept

We use Figure 3.6 to explain the TN concept [33]. In Figure 3.6(a), we have a simple network where E0 is the ingress LER and E1 through E3 are three egress LERs that are

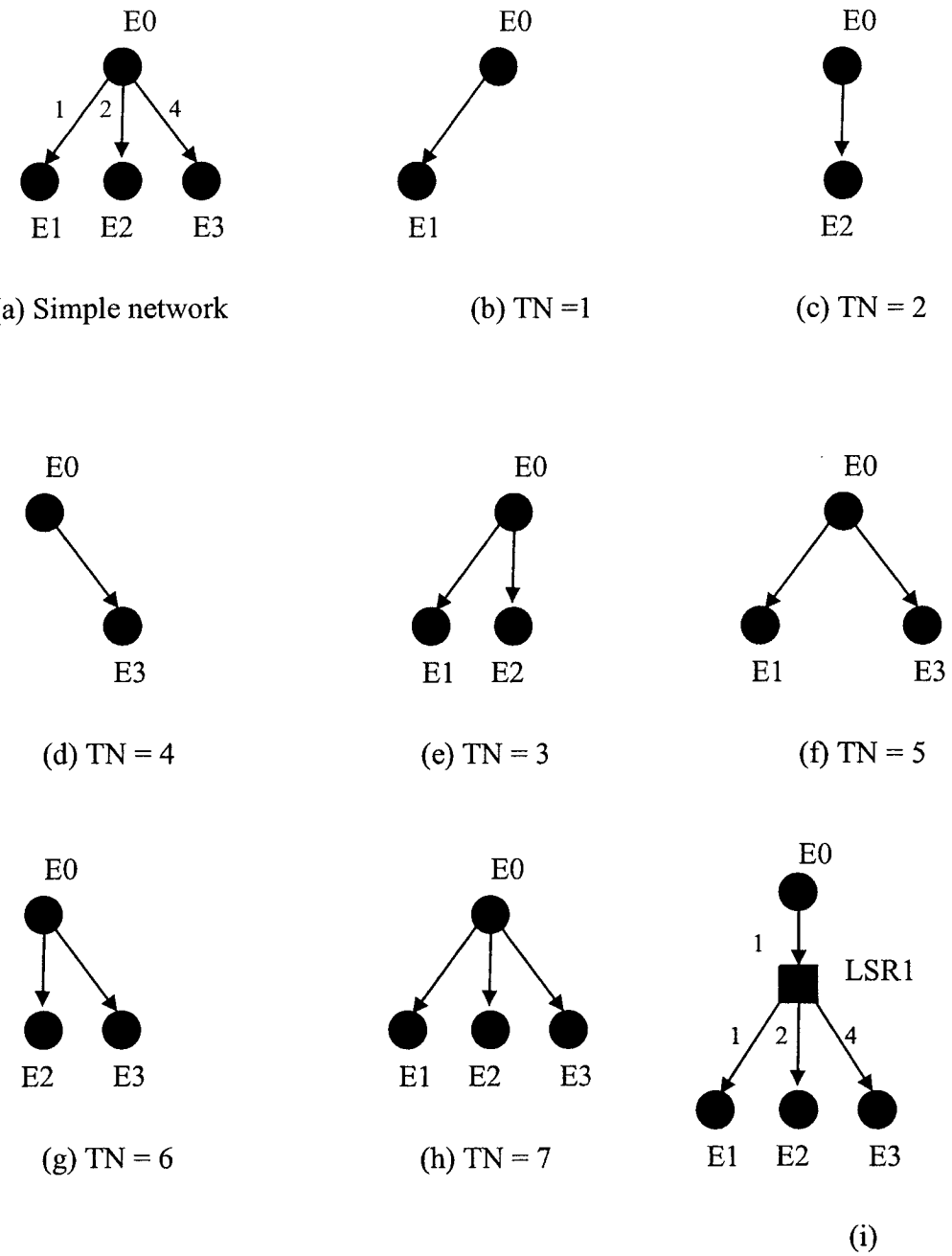


Figure 3.6 Explanation of Tree Numbering concept.

directly connected to E0. Figures 3.6(b) through 3.6(h) show the MPLS trees that can be built in the network. Each one of these trees has a structure that is different from the structures of the other trees.

Note that the number of trees depends only on the number of the egress LERs and not on the network topology. For instance, an addition of a new link to the network as in Figure 3.6(i) does not change the number of the trees; however the addition of that link changes the structure of each tree.

This allows us to distinguish each tree by a number from 1 to 7. The value of each tree number depends on how many egress LERs connected to that tree. If each interface of E0 is assigned a weight value that is a power of 2 and the weight values of these interfaces are different, then the tree number of each tree is calculated as the sum of the weight values of the interfaces connecting E0 with the egress LERs of that tree. Assume the interfaces of E0 are weighted as shown in Figure 3.6(a) with values 1, 2 and 4. Then, the seven trees are assigned the numbers 1 through 7 as given in the Figures 3.6(b) through 3.6(h).

Each egress LER can also be assigned a weight value. The weight value for an egress LER is calculated by multiplying the weight values of the interfaces over the path that connects the ingress LER with that egress LER. For example, in Figure 3.6(i) the weight value of LER E2 is equal to $(1 \times 2 = 2)$, which is the weight value of the interface that connects E0 with LSR1 multiplied by the weight value of the interface that connects LSR1 and E2.

Assume now a new egress LER E4 is added to the network in Figure 3.6(i). The new LER is connected directly to LSR1 as shown in Figure 3.7(a). In this case, the weight value of the interface that connects LSR1 with E4 is assigned value 8, which is 2 to the power of 3. The number of trees that can be generated in the network is equal to 15. Assume we change the topology of the network to the one shown in Figure 3.7(b). The

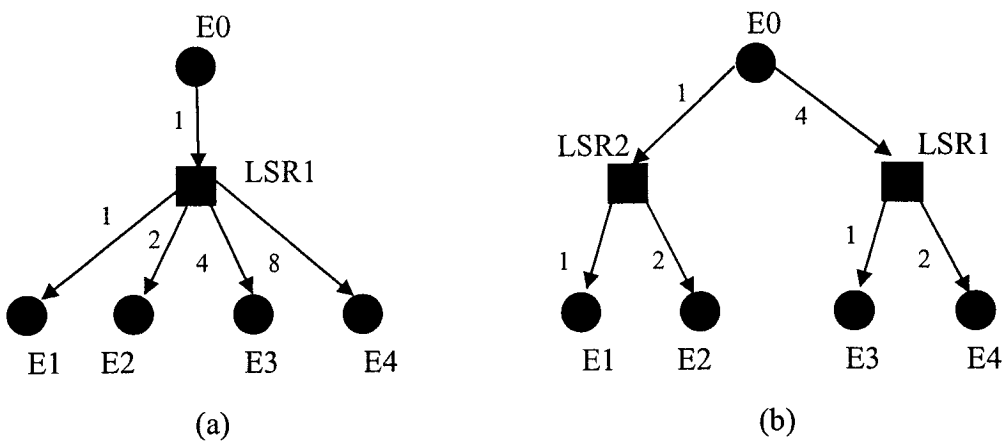


Figure 3.7 Two networks topologies of four egress LERs.

number of trees that can be generated in the network is still equal to 15; however, the structures of the trees are different. Moreover, the weighting procedure of the interfaces is different because E0 has two interfaces as well as LSR1 and LSR2. Both LSR1 and LSR2 use the same procedure to weight their interfaces because each node of them is connected directly to two egress LERs. The weighting procedure at E0 is done as follows:

1. For each interface of E0, the number of egress LERs that have paths through that interface is calculated.
2. Since the tree numbering depends on weighted values of power 2, start weighting one of the interfaces with number 1 (2 to power 0).
3. The weight value of the next interface should start from the next unused weight value. Since the interface that connects E0 and LSR2 is weighted by 1 and there are two egress LERs downstream that interface, which means weight values 1 and

2 are used, the next weight value should be 4 and this value is assigned to the other interface that connects E0 with LSR1.

4. Each path from E0 to any egress LER is assigned a value that differs from all the other values. For instance, the weight value of E4 is equal to ($4 \times 2 = 8$), which is the weight value of the interface that connects E0 with LSR1 multiplied by the weight value of the interface that connects LSR1 and E4.

3.4.2 Implementation of the TN Concept in a Large Tree

The implementation of TN concept in a large network involves message exchanges between the routers and modification of the MRT (Multicast Routing Table). The implementation is divided into three steps:

Step 1: Index of the state entries

At each router that maintains MRT for different multicast sessions, the multicast state entries in the MRT should be indexed by the address of the ingress LER of the corresponding multicast session. This step is necessary to gather all multicast state entries that are shared by the same ingress LER. The state entry of the multicast session would be modified as $E_i : (S,G) : (I, O_1, O_2, ..., O_m)$, (See Chapter 2, Figure 2.6), where E_i is the ingress LER of multicast session *(S, G)*, *I* is the incoming interface and O_1 through O_m are the outgoing interfaces. In order to maintain a path state at each router that has a multicast tree, we can use one of the two options mentioned in the case of SE: either RSVP protocol [32] can be used if a QoS is employed in the network, or a specific protocol can be defined. The path message in RSVP or in the new defined protocol

should also carry the IP address of the ingress LER to all the routers spanned by a certain tree.

Step 2: Weighting the outgoing interfaces

In this step, the outgoing interfaces of the ingress LERs as well as the ones of the LSRs are weighted. The weight values of these interfaces are then used in the next step to assign a number for each tree in the network. As discussed previously, the weight values of the outgoing interfaces at a certain router depend on the number of egress LERs downstream from each interface. For this purpose, each router (ingress LER or LSR) maintains a weighting table for its outgoing interfaces that have the same incoming interface and are indexed by the same address of the ingress LER. The weighting procedure of the outgoing interfaces is carried out as follows:

1. The weighting process starts by collecting the number of egress LERs for each outgoing interface. The collection process of these numbers starts at the LSRs connected directly to the egress LERs and proceeds in the reverse direction toward the ingress LER. Also, RSVP can be used to carry this information and distribute it to all LSRs and the ingress LER. RSVP uses the path state information maintained in step 1 to distribute this information in the reverse direction of the tree (i.e., from the egress LERs towards the ingress LER).
2. For each outgoing interface i where $1 \leq i \leq m$, and m is the number of outgoing interfaces of a router, the number of egress LERs, N^{i}_{egress}, is collected and recorded under the corresponding outgoing interface in the weighting table.

3. After collecting N^i_{egress} for each interface, the weight value of interface i is calculated as:

$$W_i = \begin{cases} 1, i = 1 \\ 2^{\alpha i}, i \geq 2 \end{cases} \tag{3.1}$$

where

$$\alpha_i = \sum_{k=1}^{i-1} N^k_{egress} \tag{3.2}$$

Equation (3.1) assigns value 1 to interface 1 and for any interface greater than 1, the weight value of that interface is equal 2 to the power of the sum of the egress LERs downstream from all the already weighted interfaces. The meaning of Equations (3.1) and (3.2) is illustrated by the weighting procedure given in Section 3.4.1.

Step 3: Tree Numbering and data forwarding

Each router on the tree contributes in the calculation of the tree number. For each multicast session, the partial weight value calculated by a router having m outgoing interfaces is:

$$W = \sum_{i=1}^{m} P_i \times W_i \times WDS_i \tag{3.3}$$

where

$$P_i = \begin{cases} 1; \text{If interface } i \text{ is active} \\ 0; \text{If interface } i \text{ is not active} \end{cases} \tag{3.4}$$

WDS_i is the partial weight value received from the downstream router connected directly to interface i, and W_i is the weight value of interface i. Interface i is active if there are downstream egress LERs from that interface for the corresponding session. Note that

WDS values for all interfaces connected directly to the egress LERs are equal to 1. After the calculation of the partial weight values by the LSR, this value is sent to the upstream router through the incoming interface. This process is repeated by all the LSRs until all the partial weight values have been received by the ingress LER, which finally calculates the weight value of the tree that indeed is the tree number for the corresponding multicast session.

During the calculation process of the tree number, the routers can exchange the MPLS labels. Once the tree number is calculated by the ingress LER, this number is then considered the FEC of the corresponding multicast session. If that FEC is already assigned a label, the same label will be used for the new multicast session; otherwise, a new FEC is created and assigned a new MPLS label. Then, the ingress LER starts attaching the MPLS label to every IP multicast packet of that session and the forwarding is achieved based on the MPLS mechanism.

3.4.3 Illustrative Example

As an illustrative example, consider again the same multicast sessions presented in Table 3.2 and their multicast trees shown in Figure 3.4. Assume $T_i = \{NT_i, LT_i\}$ is the multicast tree built by multicast session *i*, where NT_i is the set of routers on tree T_i and LT_i is the set of links that connect NT_i together. We define T_{shared} as a shared tree composed of the union of *m* trees as follows:

$$T_{shared} = \{NT_{shared}, LT_{shared}\} = T_1 \cup T_2 \cup ... \cup T_m \tag{3.5}$$

where $NT_{shared} = NT_1 \cup NT_2 \cup ... \cup NT_m$

and $LT_{shared} = LT_1 \cup LT_2 \cup ... \cup LT_m$

All the trees that compose T_{shared} must have the same ingress LER. Figure 3.8 depicts the shared tree of the multicast trees shown in Figure 3.4. The weight values of the interfaces are given in parentheses. The weighting tables at LSR1, LSR2, LSR3 and the ingress LER E1 are given in Figure 3.9. The tree number of sessions (S2, G2) and (S3, G3) is calculated as follows:

The calculations start at LSR3 by calculating its partial weight value as $W_{LSR3} = W_2 \times WDS_2 = 1 \times 1 = 1$, where W_2 is the weight value of interface 2 at LSR3 and WDS_2 is the downstream weight value received from E3. W_{LSR3} is then sent to the upstream LSR2, which in turn calculates its partial weight value as $W_{LSR2} = W_1 \times WDS_1 = 1 \times 1 = 1$. Note that WDS_1 is equal to W_{LSR3}. After that, LSR1 calculates its partial weight value as $W_{LSR1} = W_2 \times WDS_2 + W_3 \times WDS_3 = 1 \times 1 + 16 \times 1 = 17$. Finally, when the ingress LER E1 receives this value, it calculates the tree number as $W_{E1} = W_2 \times WDS_2 = 1 \times 17 = 17$. The tree numbers for all sessions are given in Table 3.4.

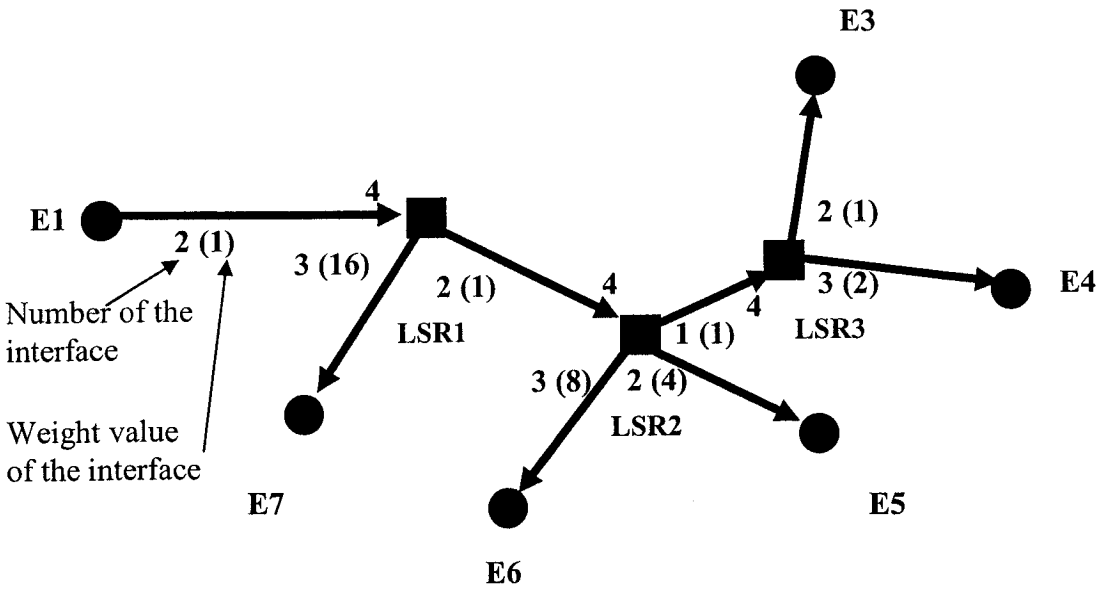


Figure 3.8 The shared tree of the multicast sessions in Table 3.2.

Interface	N_{egress}	W
2	1	1
3	1	2

(a) Weighting table at LSR3

Interface	N_{egress}	W
1	2	1
2	1	4
3	1	8

(b) Weighting table at LSR2

Interface	N_{egress}	W
2	4	1
3	1	16

(c) Weighting table at LSR1

Interface	N_{egress}	W
2	5	1

(d) Weighting table at E1

Figure 3.9 The weighting tables at LSR1, LSR2, LSR3 and ingress LER E1.

Table 3.4 Tree Numbers for the five sessions given in Table 3.2.

Session	Tree Number
(S1,G1)	4
(S2,G2)	17
(S3,G3)	17
(S4,G4)	31
(S5,G5)	22

3.4.4 Tree Maintenance

One of the main characteristics of multicasting is its dynamic behavior where at any time a current member of the multicast group can leave or a new member can join the group. In this section, we discuss the actions that should be taken when a new **Join-Request** or **Prune-Request** message arrives at a router from interface *i*. We define the access factor f_c^i as the number of multicast sessions that use *i* as an outgoing interface. For example, in Figure 3.8, the access factor of interface 2 of LSR3 is equal to 3, which is the number of multicast sessions that use that interface as outgoing interface. These sessions are: (S2, G2), (S3, G3) and (S4, G4). Before we discuss the detailed actions that take place as a response to the **Join-Request** or **Prune-Request** messages, we assume that a refresh cycle for the TN algorithm, which is composed of the three steps discussed previously, is run every defined period of time. Any **Join-Request** or **Prune-Request** messages discussed next is assumed to take place before running the refresh cycle.

Join-Request (S, G)

When a router receives a **Join-Request** message from interface i for multicast session (S, G), the response of the router to the message depends on whether the multicast state entry for (S, G) is maintained on the router or not. Three general cases are addressed as follows:

Case 1: The router has already an entry for (S, G). Furthermore, interface i has already been weighted before (i.e., $f_c^i \geq 1$). The only action that is done by the router is generating a message to update the weight value of the (S, G) session. As an example for this scenario, consider Figure 3.4(b), which shows the multicast tree for session (S2, G2). That figure is redrawn in Figure 3.10. Assume E5 sends a **Join-Request** message to LSR2 through interface 2 in order to join multicast session (S2, G2). After adding interface 2 to the interfaces list of (S2, G2) in the MRT, LSR2 responds to this message by generating an **Update-Weight** message for (S2, G2) and sends this message upstream to LSR1, which in turn sends the message to the upstream router (the ingress LER E1).

Each router on the path between the egress LER that initiates the **Join-Request** message and the ingress LER contributes in the calculation of the value carried on the **Update-Weight** message by multiplying the value received by the **Update-Weight** message with the weight value of the outgoing interface. The updated value generated at each router in the path is calculated as:

$$W_{update} = W_i \times WDS_i \tag{3.6}$$

Recall that WDS_i is the partial weight value received from the downstream router connected directly to interface i, and W_i is the weight value of interface i where the **Update-Weight** message was received. For the given example, the calculation of the

updated value starts at LSR2 by calculating its partial update weight value as $W_{LSR2} = W_2 \times WDS_2 = 4 \times 1 = 4$, where W_2 is the weight value of interface 2 at LSR2 and WDS_2 is the downstream weight value received from E5. W_{LSR2} is then sent to the upstream LSR1 which in turn calculates its updated weight value as $W_{LSR1} = W_2 \times WDS_2 = 1 \times 4 = 4$. Finally, when the ingress LER E1 receives this value, it calculates the final updated value as $W_{E1} = W_2 \times WDS_2 = 1 \times 4 = 4$, and adds this value to the current tree number. The new tree number for session (S2, G2) will be 17 + 4 = 21.

Case 2: The router has already the entry for (S, G) but $f_c^i = 0$. This means that a new egress LER En joins the shared tree for the first time. In this case, the following actions should take place:

1. After adding the interface *i* to the interfaces list for (S, G) to the MRT at the LSR that is directly connected to En, that LSR should generate an **Update-Number** message. This message is sent upstream along the path between that LSR and the ingress LER. Each router along that path should update the number of egress LERs for the interface where the **Update-Number** message is received.
2. Moreover, the weighting values of the interfaces of the routers on that path are re-calculated.
3. If the new calculation results in any change of the weighting value of any interface at any router, that router generates a message to activate step 3 of the tree numbering approach. Note that running step 3 could result in recalculations of all the tree numbers.
4. If the new calculation of the weighting values keeps the weighting values of the interfaces unchanged, interface *i* of En where the **Join-Request** message was

received should be weighted and then an Update-Weight message for (S, G) should be generated and sent to the ingress LER. An example for this scenario is the case when E2 in Figure 3.11 wants to join multicast session (S1, G1) through interface 1 of LRS1. There is a multicast state entry for (S1, G1) at LSR1 but interface 1 is not in the outgoing interfaces list of (S1, G1) and its f_c^1 is equal to 0. E2 is a new egress LER that joins the shared tree rooted at E1 for the first time. Because joining of E2 will not result in any recalculation of the weighting values at any router in the path between LSR1 and E1, LSR1 only weights interface1 using Equations (3.1) and (3.2). Then, it sends an **Update-Weight**(S1, G1) message to E1. Note that the weight value of interface 1 of LSR1 will be 32 and the new tree number of multicast session (S1, G1) will be 4 + 32 = 36.

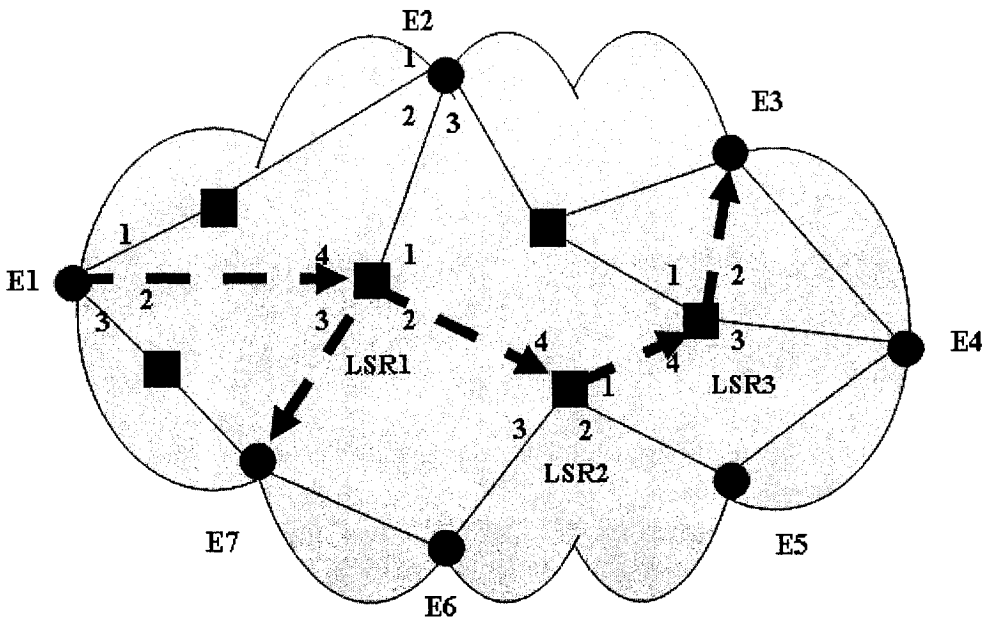


Figure 3.10 Multicast tree of Session 2(S2, G2).

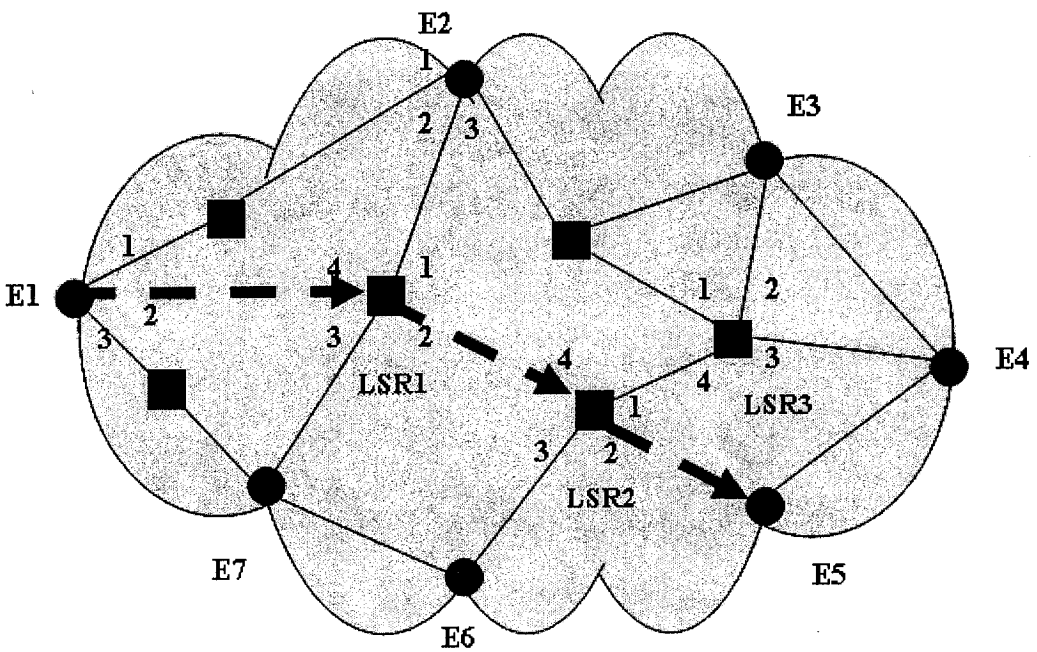


Figure 3.11 Multicast tree of Session 1(S1, G1).

Case3: There is no entry for (S, G) on the router. In order to explain this case clearly, assume En wants to join multicast session (S, G). Assume LSRn is directly connected to En. A **Join-Request**(S, G) message is sent by En to LSRn. Because LSRn has no entry for (S, G) in its MRT, the **Join-Request**(S, G) will be forwarded upstream until it is received by a router that has the multicast entry for (S, G) in its MRT. That router generates an acknowledge message and sends it back to LSRn. When LSRn receives the acknowledge message, it generates an **Index-Request** message for (S, G). The purpose of the **Index-Request** message is to get the IP address of ingress LER where the multicast tree for (S, G) is rooted. If the response message to **Index-Request** message carries an IP address that is already used in LSRn, one of the following two tracks will take place:

1. If the access factor of the interface of LSRn that is connected to En is greater than 1, the process proceeds as in case 1.
2. If the access factor of the interface of LSRn that is connected to En is equal to 0, the process proceeds as in case 2.

On the other side, if the response message to **Index-Request** message carries an IP address that is not used in LSRn, steps 2 and 3 of the Tree Numbering approach is run.

As an example for case 3, consider Figure 3.11 when E4 wants to join (S1, G1) from interface 3. A **Join-Request**(S1, G1) message is sent to LSR3. However, LSR3 has no entry for (S1, G1). The **Join-Request** message is then sent upstream to LSR2. Since LSR2 has a state entry for (S1, G1), an acknowledge message is sent back to LSR3 for multicast session (S1, G1). Furthermore, a path state is maintained at LSR3. Because LSR3 is connected directly to the egress LER E4, which initiates the **Join-Request**(S1, G1) message, LSR3 is then generates an **Index-Request**(S1, G1) message and sends it to

LSR2 which in turns sends it to LSR1. When the IP address of E1, as an ingress LER for (S1, G1) is returned to LSR3, it generates an **Update-Weight**(S1, G1) message. The new tree number for (S1, G1) is then updated as done in case 1 and case 2.

However, if for example, the returned message from LSR2 contains an IP address for an ingress LER that is not used in the weighting table at LSR3, the functions of step 2 and step 3 are activated.

Prune-Request (S, G)

When a router receives a **Prune-Request** message from interface *i* for multicast session (S, G), that router removes interface *i* from the outgoing interfaces list of the multicast state entry (S, G). If *i* is the only interface for (S, G) at the router, the **Prune-Request** message will be forwarded upstream until it reaches the ingress LER or it reaches a branching point router for (S, G). The only action that should be done is a generation of an **Update-Weight** message by the LSR that is directly connected to the egress LER that initiated the **Prune-Request** message. Then, the value of f_c^i should be decremented by 1. If f_c^i becomes 0, no further action will take place as the case in **Join-Request**. Note that f_c^i becoming 0 means that the egress LER that sent the **Prune-Request** message is leaving the shared tree. It is not necessary to send an Update-Number message as in case 2 of the **Join-Request**. The reason for that is when an egress LER leaves the shared tree; it can be assumed that the egress LER is virtually on the shared tree. This assumption has no effect on the tree numbers of all the trees that compose the shared tree. The tree maintenance will take place during the next refresh cycle. The complexity of tree maintenance in the case of Prune-Request is less than the case of Join-Request. That

advantage comes from the fact that when the Prune-Request results a leaving of an egress LER of the shared tree, no recalculation of the weighting values will be necessary as it is with Join-Request case.

Consider Figure 3.10 again when E7 wants to leave multicast session (S2, G2). E7 sends the **Prune-Request**(S2, G2) message to LSR1 through interface 3. The value of f_c^3 is then decremented by 1. Then, LSR1 generates the **Update-Weight** message for (S2, G2). the updated value is calculated using Equation (3.6) as $W_{LSR1} = W_3 \times WDS_3 = 16 \times 1 = 16$. LSR1 then sends the **Update-Weight** message to E1, which calculates the updated value as $W_{E1} = W_2 \times WDS_2 = 1 \times 16 = 16$. E1 then updates the tree number of (S2, G2) by subtracting the updated value from the current tree number. The new tree number of multicast session (S2, G2) will be 17 - 16 = 1.

Note that the **Update-Weight** message may be generated in both **Join-Request** and **Prune-Request**. For this reason, the **Update-Weight** message should indicate whether the value it carries should be added to the tree number in the case of **Join-Request** or it should be subtracted from the current tree number in the case of **Prune-Request.**

3.4.5 Uniqueness of the Tree Number

If a shared tree is composed of two or more trees, it is important that the assigned number to a certain tree be different from all the numbers assigned to all the other trees. In other words, any two different trees that have a common ingress LER should be assigned two different tree numbers. The uniqueness of tree numbers in the MPLS network guarantees the switching of IP multicast packets to the actual on-tree egress

LERs. In order to prove that each tree built from the ingress LER has a unique number, we present the following key points:

1. The weighting procedure introduced in TN algorithm is actually assigning a weight value for each egress LER in the shared tree. The weight value of egress LER E_i^w is the product of all the weight values of the outgoing interfaces that connect the ingress LER with E_i. For example, Figure 3.12 shows a shared MPLS tree with five egress LERs. The weight value of E_i^w is $E_1^w = W_3 \times W_5 \times W_6$.
2. The weight value of any egress LER can take a value that is a power of 2, (i.e, 1, 2, 4, 8, ...). This point is seen directly from Equation (3.1), which indicates that the weight value of each outgoing interface is a power of 2. Then, the result of multiplying any two or more weight values will also be a power of 2.
3. The tree number for each tree is the sum of all the weight values of the egress LERs of that tree. This point can be proved by expanding Equation (3.3) starting the calculation from the ingress LER. For example, if a multicast tree is built to deliver the packets to all the egress LERs in Figure 3.12, then the Tree Number, *Tnum*, of that tree is calculated as:

$$\begin{aligned} Tnum &= W_6 \times (W_5 \times (W_3 + W_4 \times (W_1 + W_2))) + W_{10} \times (W_9 + W_8 \times W_7) \\ &= W_5 \times W_6 \times (W_3 + W_1 \times W_4 + W_2 \times W_4) + W_9 \times W_{10} + W_7 \times W_8 \times W_{10} \\ &= (W_3 \times W_5 \times W_6) + (W_1 \times W_4 \times W_5 \times W_6) + (W_2 \times W_4 \times W_5 \times W_6) + (W_9 \times W_{10}) \\ &\quad + (W_7 \times W_8 \times W_{10}) \\ &= E_1^w + E_2^w + E_3^w + E_4^w + E_5^w \end{aligned}$$

Theorem 3.1 Given a shared tree that is defined as in Equation (3.5). If the shared tree is weighted as discussed in section 3.4.1, then each tree of the trees that compose the shared tree has a unique number.

Proof. Since the tree number for each tree is calculated as the sum of the weight values of the egress LERs of that tree and because the weight values of the egress LERs are powers of 2, it is necessary to prove that the weight value of each egress LER is different from the weight values of all the other egress LERs in the shared tree.

In order to prove the last statement, assume we have an MPLS shared tree where the ingress LER E0 has m outgoing interfaces. The weight values of these interfaces are arranged in order such that $Wi < Wi+1$ where $1 \le i \le m-1$. Assume all the egress LERs reached by the same interface j are grouped together in sub-tree G_j and all the egress LERs reached by the same interface $j+1$ are grouped together in sub-tree G_{j+1} as shown in Figure 3.13. Assume that G_j is connected to E0 through an outgoing interface j that connects E0 and LSRj and G_{j+1} is connected to E0 through interface $j+1$ that connects E0 and LSRj+1. Assume Ej is an egress LER in sub-tree G_j and Ej+1 is an egress LER in sub-tree G_{j+1}. Assume the number of egress LERs in G_j is X_j and the number of egress LERs in G_{j+1} is X_{j+1}. Then, the weight values of Ej and Ej+1 are given as: $E_j^w = W_a \times W_j = W_a \times 2^{\sum_{i=1}^{j-1} Xi}$ and $E_{j+1}^w = W_b \times W_{j+1} = W_b \times 2^{\sum_{i=1}^{j} Xi}$, (see Equations (3.1) and (3.2)), where W_a is the partial weight value that connects LSRj with E_j and W_b is the partial weight value that connects LSRj+1 with E_{j+1}. We use the contradiction principle to prove that E_j^w and E_{j+1}^w are not equal.

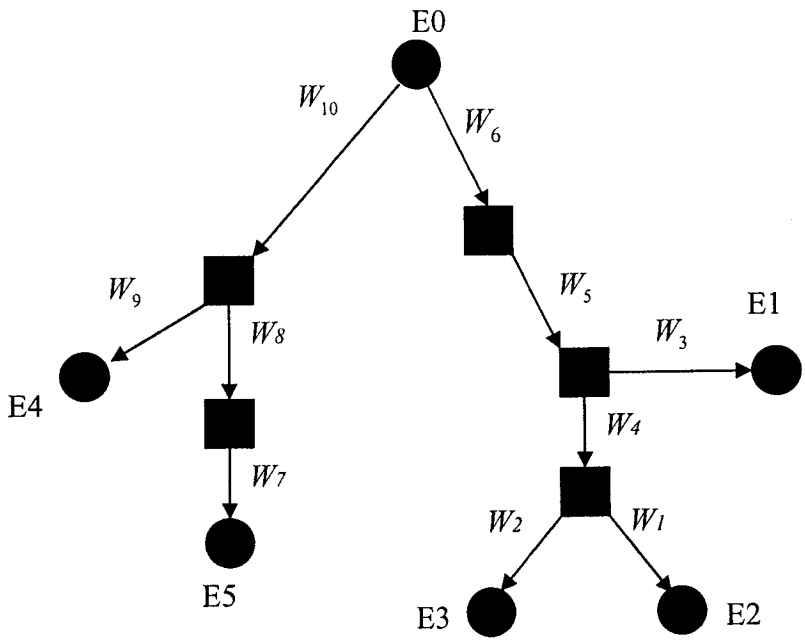
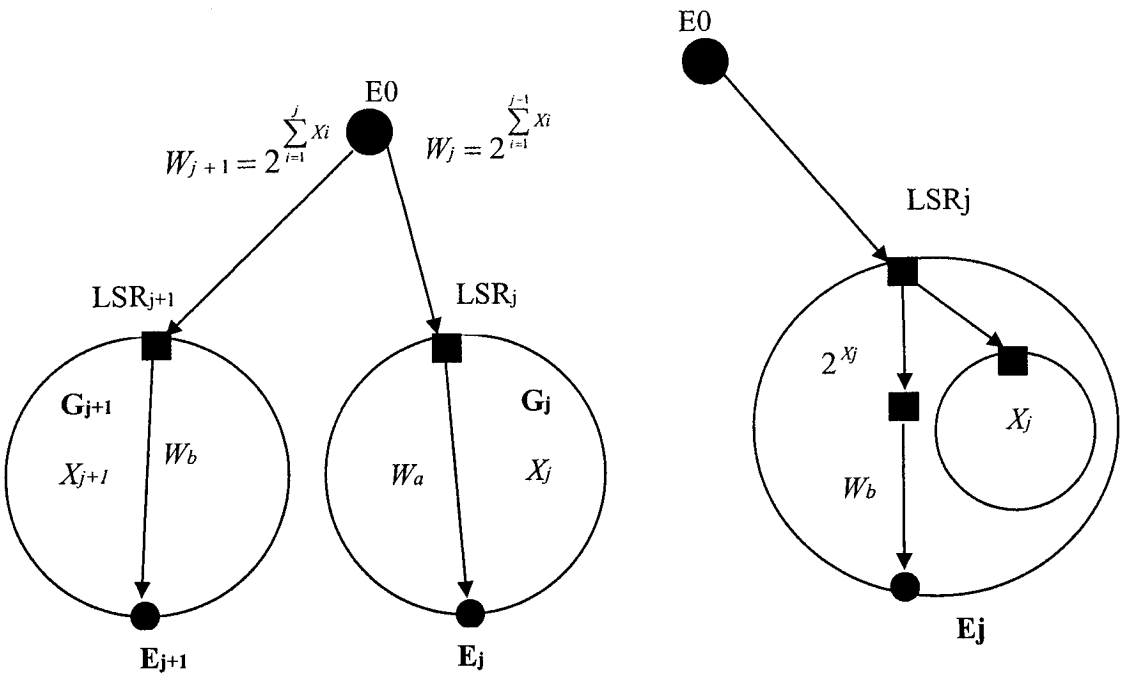


Figure 3.12 A shared MPLS tree of five egress LERs.

Assume that $E_j^w = E_{j+1}^w$. This assumption gives $W_a \times 2^{\sum_{i=1}^{j-1} Xi} = W_b \times 2^{\sum_{i=1}^{j} Xi}$, or $W_a = W_b \times 2^{Xj}$. From the last equation, we can consider Gj as a sub-tree rooted at LSRj which has two outgoing interfaces: the first one has $\alpha = 0$, (see Equation (3.1)), and connected to Xj egress LERs, and the second one has $\alpha = X_j$ and connected to E_j (See Figure 3.13(b)). This makes the total number of egress LERs in sub-tree Gj equal to 1+Xj. This result contradicts the assumption that Gj contains Xj LERs and our assumption that $E_j^w = E_{j+1}^w$ is not correct, hence each tree has a unique number. □



(a) Sub-trees Gj and Gj+1

(b) Sub-tree Gj when considering the value of $W_a = W_b \times 2^{Xj}$

Figure 3.13 An ingress LER connected to two sub-trees Gj and Gj+1.

3.5 Related Work

Recently, many approaches have been suggested to deploy multicast communication in MPLS networks. In order to support MPLS network multicasting functionalities, extensions to RSVP-TE and LDP are proposed in [34]. The proposed extensions concentrate on enabling the signaling mechanism to establish, maintain, and terminate multicasting connections. The paper descriptively discusses the implementation of multicasting by defining the extended objects and the necessary message types in RSVP-TE and by defining the extended messages format and multicasting Type-Length-Value (TLV) in LDP.

A mechanism that triggers label binding in MPLS multicasting based on traffic-driven instead of control-driven is suggested in [35]. The authors propose that the first packet belonging to any new FEC should trigger the label distribution between LSRs. The first packet is forwarded using traditional IP routing. Any packet that belongs to the FEC for which its first packet has already been received will be processed based on the label switching mechanism instead of traditional IP routing. The aforementioned mechanisms have addressed only the support of each multicast flow individually and they do not handle the aggregation and scalability issues as in the other approaches [31, 33, 36, 37, 38, 39, 40].

In [36, 37], an approach is proposed to construct multicast trees in an MPLS network where the multicast forwarding states are kept only in the branching routers of the multicast tree. Then, LSPs are built only between the branching routers. We call their approach LSP between Branching Points (LBP). However, building LSPs between the branching points means an extra time would be added at each branching point to process the IP packets which makes this approach not fully compatible with standard MPLS processing.

A multicast scheme called Edge Router Multicasting (ERM) [38, 39] has been introduced, which aims to build a multicast tree where branching points occur only on the edges of an MPLS domain. In this case, multicast flow is transformed into multiple quasi-unicast flows. One of the advantages of the ERM approach is that it allows the aggregation between multicast and unicast flows. However, to build the ERM trees, different multicast routing algorithms would need to be developed. Moreover, the

proposed ERM scheme would not usually create a fully optimized multicast delivery tree in terms of the least number of links.

Moreover, one of the drawbacks of the LBP and ERM approaches is that they depend on another router in the MPLS domain to coordinate the building of the MPLS tree. For example, in [36, 37], a network information manager system (NIMS) for each group is introduced to collect the join and leave messages from all group members. Then NIMS computes the multicast tree for each group. Based on this computation, each branching router is determined. After that, NIMS sends BRANCH messages to all branching routers to inform them about their next branching router. The same happens in [38, 39] with the extended version of ERM called ERM2. The authors define a Multicast Management (MM) router. The function of the MM is to keep a record of the current active on-tree edge routers.

Y. Oh et al. [40] have proposed an approach that handles the scalability issue in multicasting. The authors consider an MPLS multicast mechanism under Internet broadcasting systems. They propose a label aggregation scheme to solve the scalability problem of multicast streams inside the MPLS domains. Their approach depends on maintaining a table at the ingress LER called Tree Node Table (TNT). The purpose of this table is to store the IP addresses of the all routers spanned by the multicast tree. If two or more multicast sessions have the same entry in the TNT table, they are assigned the same MPLS label. A comparison between different MPLS multicasting approaches that handle the aggregation and scalability issues is given in Table 3.5.

Table 3.5: A comparison between different MPLS multicasting approaches.

Approach	Building the MPLS tree	Recourse to L3 processing	Needs changes in the routing protocols	Aggregation Capability
LBP	Centralized	Yes (at branching points)	No	With unicast flows
ERM	Centralized	Yes (at edges of MPLS Network)	Yes	With unicast and multicast flows
TNT	Distributed	No	No	With multicast flows
SE	Distributed	No	No	With multicast flows
TN	Distributed	No	No	With multicast flows

LBP: LSP between Branching Points.
ERM: Edge Router Multicasting.
TNT: Tree Node Table
SE: State Encoding.
TN: Tree Numbering.

3.6 Performance Evaluation

We evaluate the performance of the proposed approaches in terms of the number of labels used versus the number of multicast sessions, and in terms of the memory size needed to maintain the FECs at the ingress LER. In addition to these metrics, we define a new one that indicates how much any proposed MPLS multicasting approach matches the objective of MPLS protocol by reducing the recourse to layer 3 processing (i.e., traditional routing of the IP packet).

Assume we have a shared MPLS tree with N_{LER} egress LERs. If u is the number of multicast trees that can be generated and shared by the same ingress LER then u is given as:

$$u = 2^{N_{LER}} - 1 \tag{3.7}$$

Assume N_{LER}^{j} and N_{LSR}^{j} are the numbers of egress LERs and LSRs for tree j, respectively, where $1 \leq j \leq u$. Assume there are s multicast sessions that pass through the MPLS network from the ingress LER.

3.6.1 Number of Labels

Without label aggregation, each session of s requires a label. The total number of labels needed at the ingress LER is a linear relationship given as:

$$L_{NoAgg} = s \tag{3.8}$$

In order to illustrate the number of labels that could be reduced by multicast label aggregations approaches (SE, TN and TNT), we introduce a simple analysis. Assume the number of trees that can be built by s sessions is k where $1 \leq \mathrm{k} \leq \mathrm{s}$. Let q_k be the probability of building k trees by the s sessions. Then, we define the average number of labels as:

$$L_{ave} = 1 \times q_1 + 2 \times q_2 + \ldots + s \times q_s = \sum_{k=1}^{s} k \times q_k \tag{3.9}$$

Moreover, we define the L_{sav} as the percentage of labels reduced by aggregation related to labels for no aggregation as:

$$L_{sav} = 100 \times (1 - \frac{L_{ave}}{s}) \tag{3.10}$$

We show the advantage of label aggregation in three cases. Case 1 is an example of the average case where the probability of building k trees by s sessions is uniformly distributed. The other two cases are examples for the extreme cases. Case 2, for instance, is an example of the best case when the probability of building the same tree for all the

sessions is high. On the other hand, case 3 is an example of the worst case when the probability of building a tree for each session is high. The detailed analysis of these cases is as follows:

Case 1: when the probabilities are uniformly distributed. In this case

$$q_k = \frac{1}{s}, 1 \le \mathrm{k} \le \mathrm{s} \tag{3.11}$$

For this case, Equation (3.9) can be written as:

$$L_{ave} = \sum_{k=1}^{s} k \times q_k = \sum_{k=1}^{s} k \times \frac{1}{s} = \frac{1}{s} \sum_{k=1}^{s} k = \frac{s+1}{2} \tag{3.12}$$

Figures 3.14 and 3.15 show the relationship between the number of labels and L_{sav} versus the number of multicast sessions when $q_k = \frac{1}{s}, 1 \le \mathrm{k} \le \mathrm{s}$.

Case 2: In this case, it is assumed that all the sessions will be on the same tree with probability q_1. For this case, Equation (3.9) can be written as:

$$L_{ave} = q_1 + \sum_{k=2}^{s} k \times q_k \tag{3.13}$$

Moreover, if we assume that $q_k = \frac{1-q_1}{s-1}$, for $2 \le \mathrm{k} \le \mathrm{s}$. Equation (3.13) can be reduced to:

$$L_{ave} = q_1 - \frac{1-q_1}{s-1} + \frac{s(s+1)(1-q_1)}{2(s-1)} \tag{3.14}$$

Case 3: In this case, it is assumed that all the sessions will have s trees with the probability qs. For this case, Equation (3.9) can be written as:

$$L_{ave} = s \times q_s + \sum_{k=1}^{s-1} k \times q_k \tag{3.15}$$

If we assume that $q_k = \frac{1-q_s}{s-1}$, for $1 \leq k \leq s-1$, Equation (3.15) can be reduced to:

$$L_{ave} = s \times q_s + \frac{s(1-q_s)(s+1)}{2(s-1)} - \frac{s(1-q_1)}{(s-1)} = s \times q_s + \frac{s(1-q_s)}{2} = \frac{s}{2}(1+q_s) \qquad (3.16)$$

Figures 3.16 and 3.17 show the relationship between the average number of labels and L_{sav} versus the number of multicast sessions for the case 2 while Figures 3.18 and 3.19 depict the relations in case 3. A heuristic analysis and the derivations of q_k for $1 \leq k \leq s$ are given in detail in [40]. We aim by presenting the above analysis to show the advantage of label aggregation. For example, the assumption in case 2 represents the best case when most of the sessions build the same multicast tree. In that case, the saving in label usage is close to 70 percent (Figure 3.17), even though the value of q_1 is equal to 0.4. On the other hand, case 3 represents the worst case when most of the sessions build different trees. However, a reasonable saving in label usage is achieved, which is equal to 15 percent when q_s is equal to 0.7 as shown in Figure 3.19.

In order to ensure the advantage of label aggregation, we carried out a simulation program on three network topologies as: Network 1 is composed of 25 nodes and 55 links, Network 2 is composed of 28 nodes and 45 links, and Network 3 is composed of 63 nodes and 142 links. For each network, a fixed node is selected as ingress LER, and among the other nodes we select *SEL* as a set to represent the potential egress LERs of that network. Then we generate multicast sessions from 1 to 2000. For each session, we select randomly i nodes from *SEL* as the egress LERs of that session, where

$2 \leq i \leq SEL$. Then we compare the structure of the tree that is built for each session with the trees that are already built in the network for the current active sessions. If the generated tree is already deployed in the network, no extra MPLS label is used. If the tree is a new tree, then a new MPLS will be consumed. Figure 3.20 shows the result obtained from the simulation for the three network topologies.

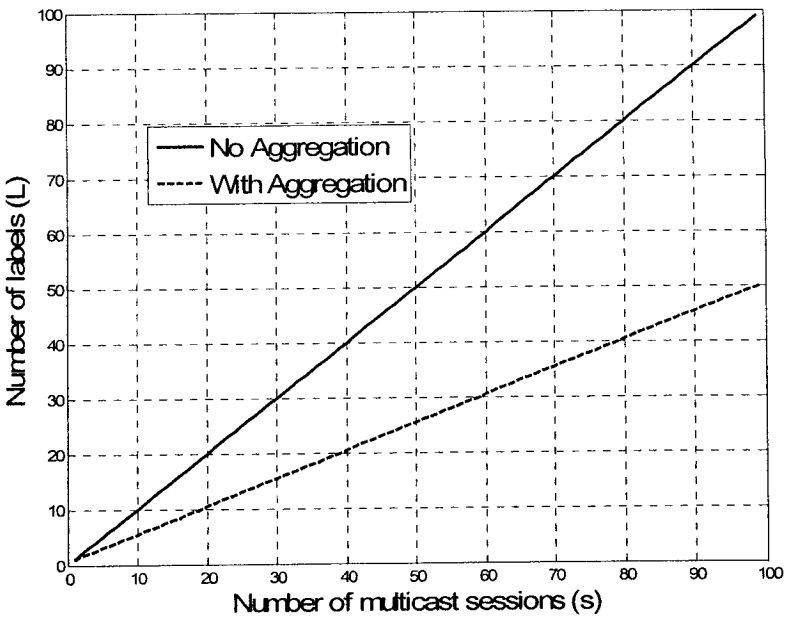


Figure 3.14 Number of labels vs. Number of multicast sessions (case 1).

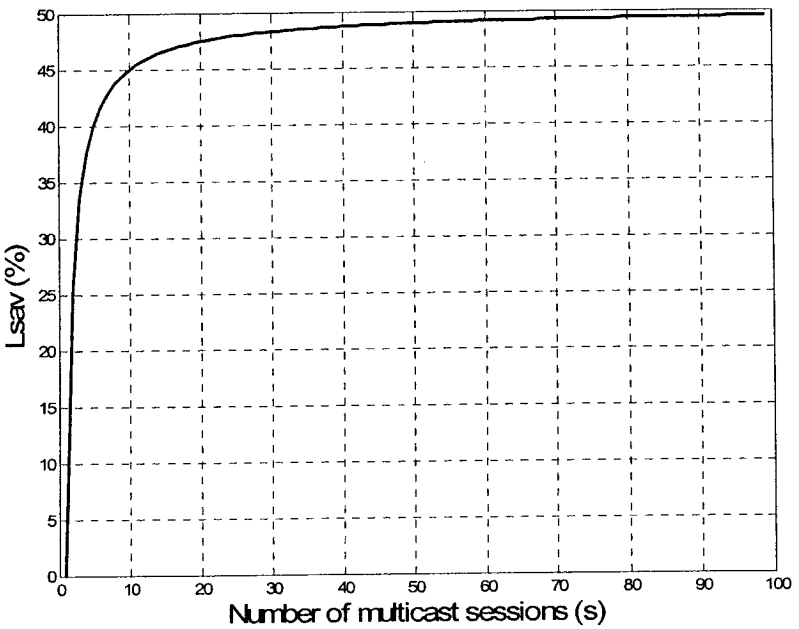


Figure 3.15 L_{sav} vs. Number of multicast sessions (case 1).

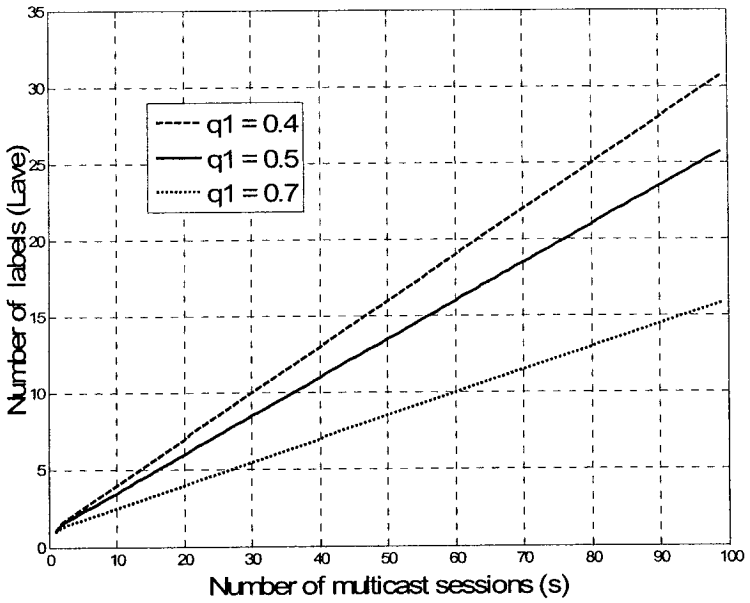


Figure 3.16 Number of labels vs. Number of multicast sessions (case 2).

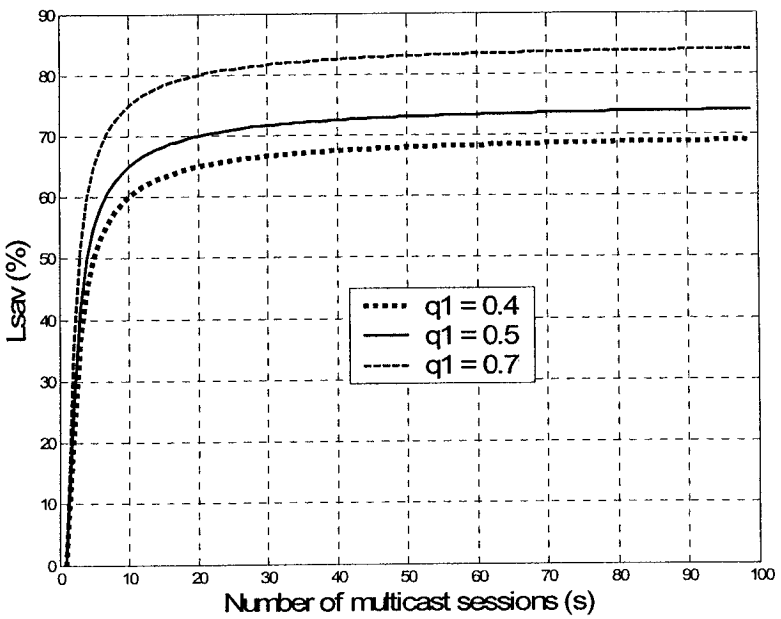


Figure 3.17 L_{sav} vs. Number of multicast sessions (case 2).

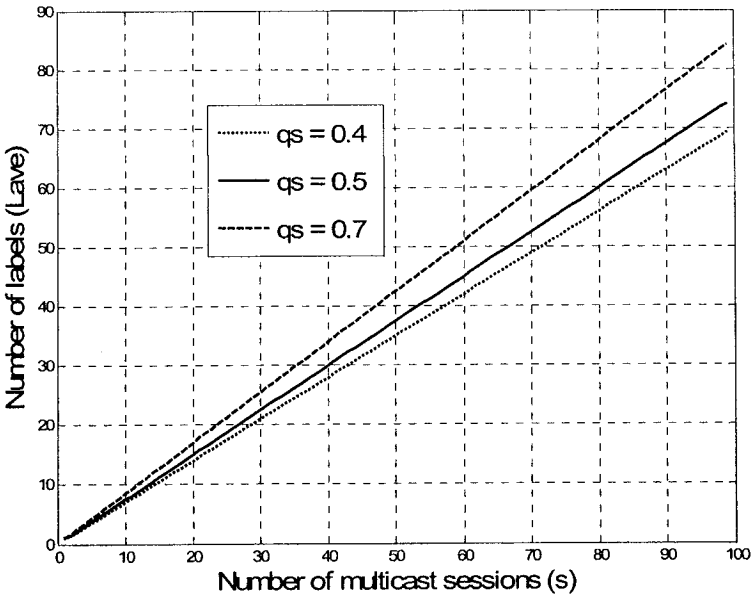


Figure 3.18 Number of labels vs. Number of multicast sessions (case 3).

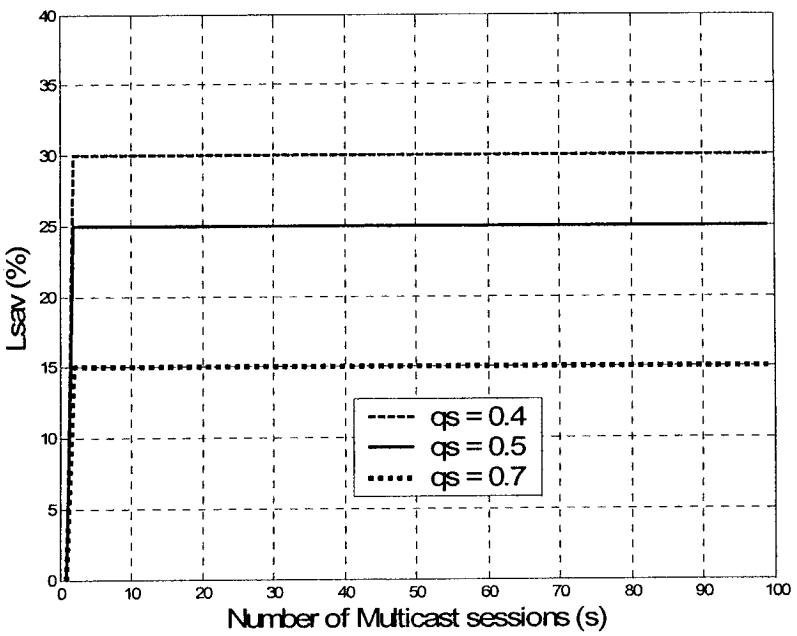


Figure 3.19 L_{sav} vs. Number of multicast sessions (case 3).

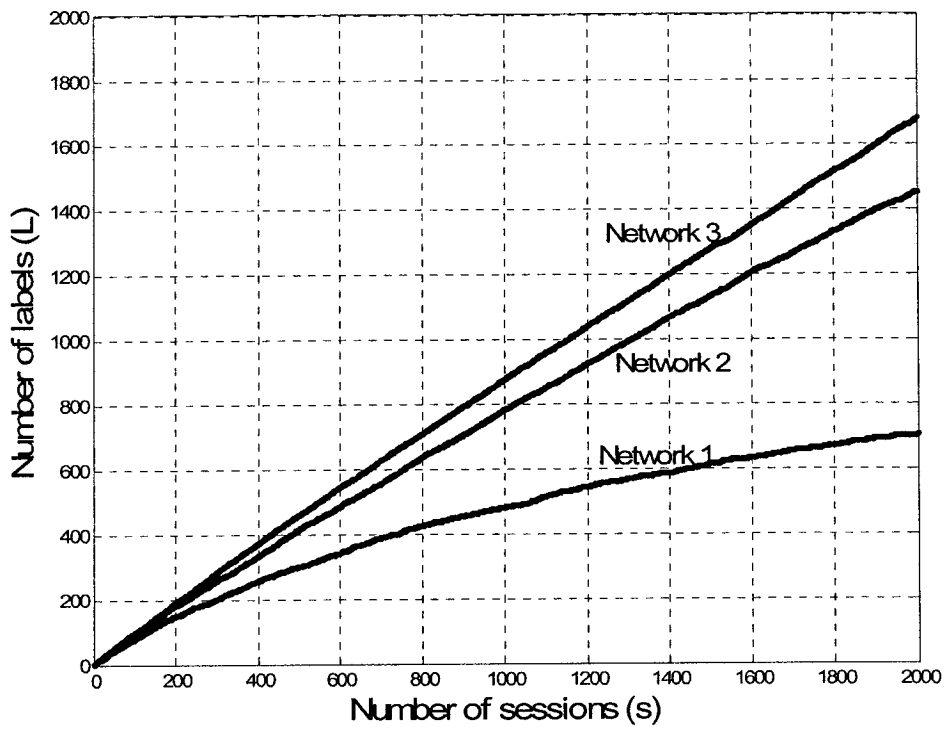


Figure 3.20 Number of labels (L) vs. Number of multicast sessions (s) generated by simulation on three network topologies.

3.6.2 Memory Size

One motivation of MPLS is to speed up the forwarding process of IP packets inside the MPLS networks. MPLS achieves this objective by doing most processing at the edge routers and relieving the core routers from the heavy processing of IP packets by forwarding them using switching-based principle. When a multicast packet arrives to the ingress LER, many tasks take place to process the packet before being switched into the network. One of these tasks is the classification of the packet into its appropriate FEC. The FECs in our approaches are the tree codes if SE approach is used or the tree numbers if TN approach is used. If TNT approach is used, the FEC for each tree is then the IP addresses of the routers of the corresponding tree. Further improvement of the processing time at the ingress LER can be achieved by reducing the size of the memory that maintains the FEC information. Since each IP multicast packet requires a look-up in the FEC table, reducing the size of that table means speeding up significantly the forwarding process for each multicast packet.

In this section, we compare the performance of our approaches with the one that has been offered in [40] in terms of the memory size needed to maintain the information about each tree at the ingress LER. The reason of choosing [40] is that it is the only reference among the others that propose a mechanism to distinguish the structure of the tree as SE and TN approaches. As was discussed in the previous section, TNT, SE and support label aggregation and provide a significant saving in the usage of the labels. However, these approaches maintain different memory sizes at the ingress LER.

In [40], the authors use Tree Node Table (TNT) that maintains the IP address of each egress LER and LSR spanned by a certain multicast tree. The number of entries in the TNT table for each tree j is calculated as:

$$EY_j = N_{LER}^j + N_{LSR}^j \tag{3.17}$$

where N_{LER}^j and N_{LSR}^j are the numbers of egress LERs and LSRs for tree j, respectively, as defined before. The size of the TNT table in bits is then given as:

$$Z_{TNT} = 32 \times \sum_{j=1}^{u} EY_j = 32 \times \sum_{j=1}^{u} (N_{LER}^j + N_{LSR}^j) \tag{3.18}$$

where 32 is the number of bits for each IP address. Using the state encoding (SE) approach, the size of each code for each tree depends on the number of interfaces that participate in generating that code at each LSR router and at the ingress LER. The size of the table needed to maintain all the codes of all the trees in the SE approach is given as:

$$Z_{SE} = (f-1) \times \sum_{j=1}^{u} (1 + N_{LSR}^j) \tag{3.19}$$

where f is the number of interfaces at each router where it is assumed that all the LSRs generate the same size code.

The memory size needed to maintain the number of the trees in the TN approach depends on the number of the trees and not on the number of LSRs. If we assume that each entry needs 32 bits, then the memory size for the table is:

$$Z_{TN} = 32 \times u \tag{3.20}$$

where u is defined in Equation (3.7). This means that only one entry in the memory is needed to maintain the number of each tree.

In order to compare between the TNT, SE and TN approaches in memory size, four network topologies are used in our simulation as given in Table 3.6. Network 1 represents Toronto Metropolitan [41] while network 2 represents U.S. Long-Distance network [42]. Network 3 and network 4 are generated using the Waxman model [43]. For network 1 and network 2, the number of egress LERs in the shared tree is increased from 2 to 10 while in the other two networks it is increased from 6 to 16. The results are depicted in Figures 3.21 through 3.24. In Equation (3.19), the value of f is assumed to be 9, such that each router will produce a sub-code of 8 bits.

Table 3.6 Network topologies used in our simulation.

Network	Number of nodes	Number of links
Network 1	25	55
Network 2	28	45
Network 3	63	142
Network 4	87	175

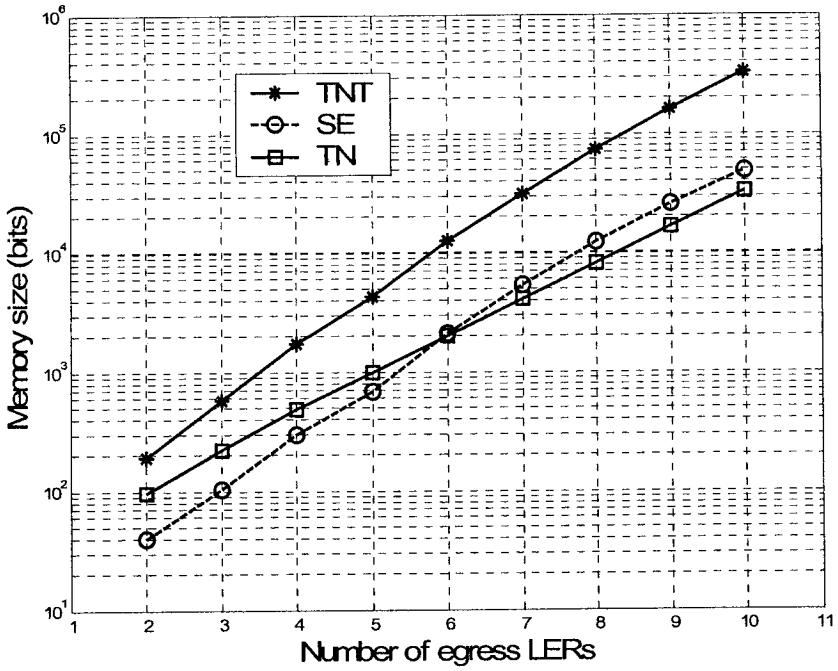


Figure 3.21 Memory size vs. Number of egress LERs (Network 1).

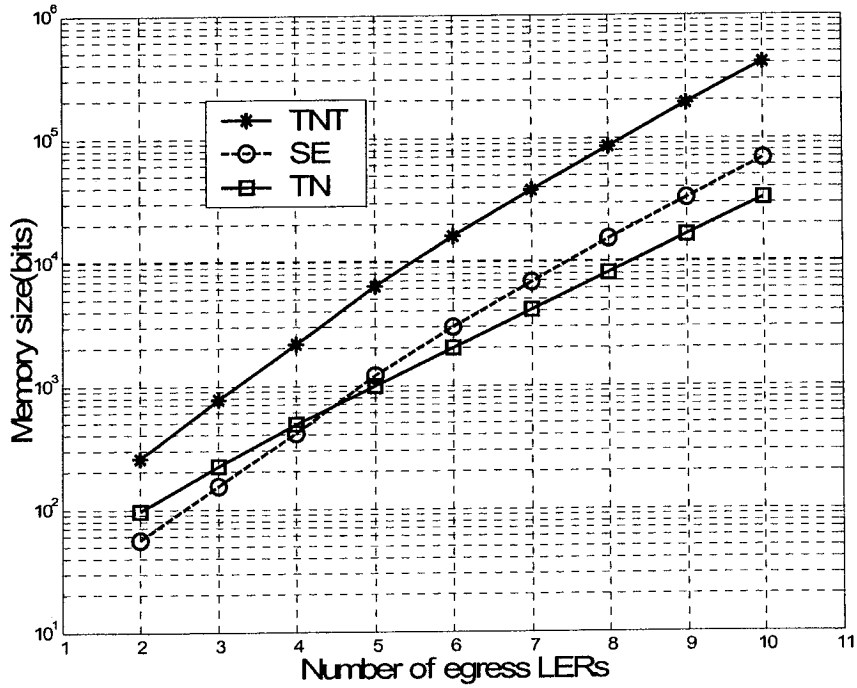


Figure 3.22 Memory size vs. Number of egress LERs (Network 2).

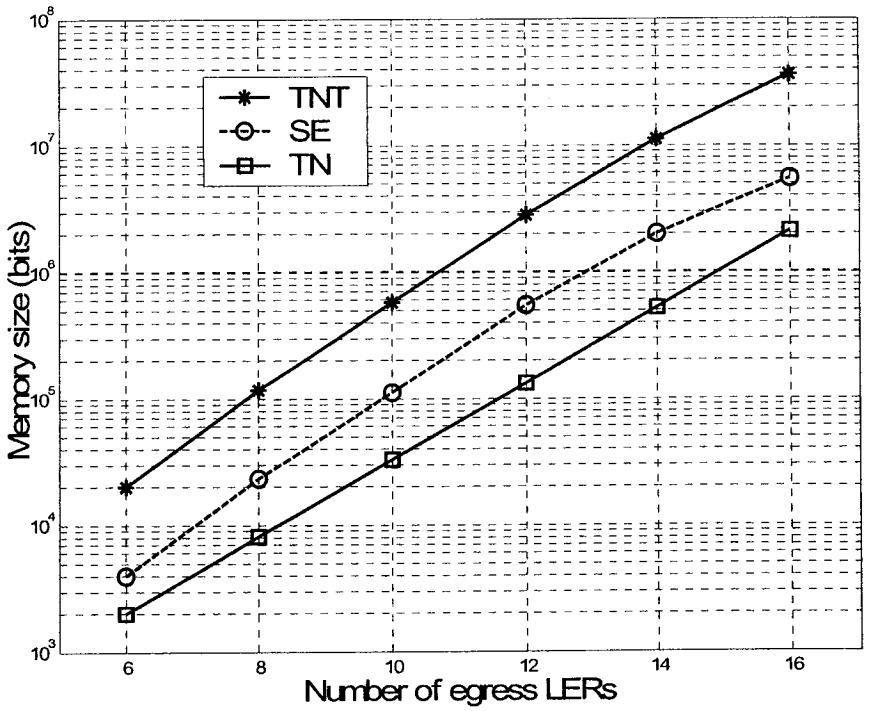


Figure 3.23 Memory size vs. Number of egress LERs (Network 3).

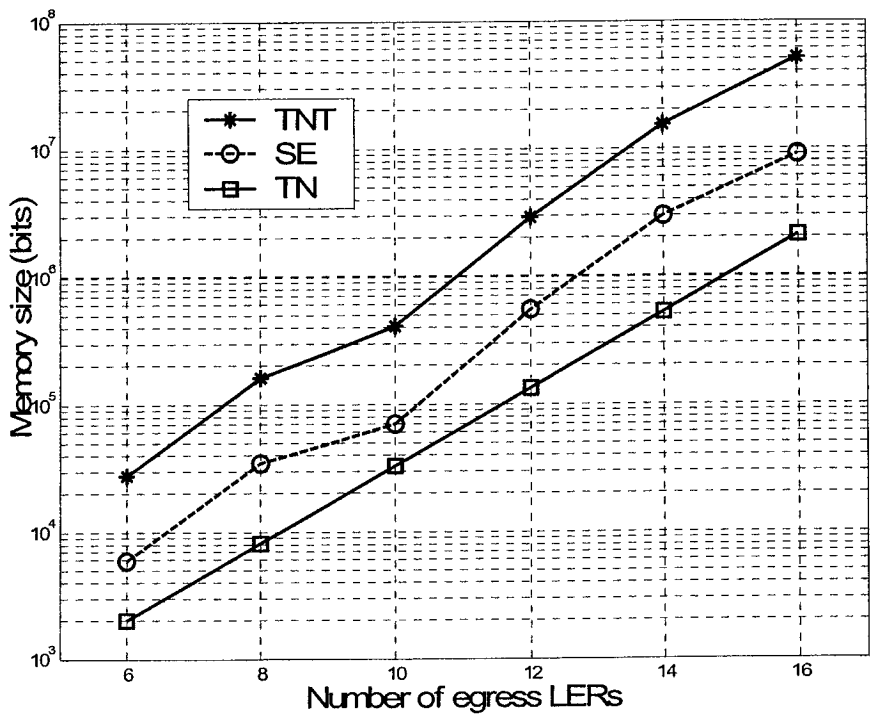


Figure 3.24 Memory size vs. Number of egress LERs (Network 4).

3.6.3 Recourse to Layer 3 (L3) Processing

The basic motivation of MPLS is to speed up the forwarding process of IP packets inside the MPLS networks and to introduce a scalable approach that handles the growing increase in IP flows. MPLS achieves the first goal by doing most processing at the edge routers and relieving the core routers from the heavy processing of IP packets by forwarding them using switching-based principle. The second goal is achieved by classifying several data flows into the same FEC, then each FEC will be mapped to a label and each label is allocated to an LSP. On the other hand, the duplication process of IP multicast packet is a necessary step at each branching point of the delivery tree. It is a great advantage for any approach that supports MPLS multicasting to keep the standardized method of MPLS protocol as much as it is possible. This means that it is better to reduce the recourse to layer 3 (L3) to process the IP packet. In order to measure the compatibility of any proposed approach with MPLS protocol, we define $MPLS_{comp}$ for a certain tree as

$$MPLS_{comp} = 100 \times (1 - \frac{N_{IP}}{N_{LSR}}) \tag{3.21}$$

where N_{IP} is the number of routers that require recourse to traditional IP processing and N_{LSR} is the number of LSRs in that tree. In TNT, SE and TN approaches, $MPLS_{comp}$ for any tree is 100 percent. However, in LBP and ERM approaches, $MPLS_{comp}$ is mostly less than 100 percent. In the LBP approach, $MPLS_{comp}$ for a certain tree depends on the

number of branch routers in that tree and in ERM it depends on the number of upstream edge routers [38].

Table 3.7 The seven trees generated in Network 3.

Tree	N_{LER}	N_{LSR}	N_{IP}	
			LBP	**ERM**
T1	16	24	10	4
T2	15	29	10	4
T3	18	30	12	7
T4	19	31	12	6
T5	20	32	13	5
T6	23	31	14	8
T7	29	29	16	9

Table 3.8 The seven trees generated in Network 4.

Tree	N_{LER}	N_{LSR}	N_{IP}	
			LBP	**ERM**
T1	4	15	2	1
T2	10	22	5	1
T3	13	22	8	4
T4	18	42	12	4
T5	19	42	14	6
T6	22	35	13	4
T7	22	43	15	7

Tables 3.7 and 3.8 illustrate the N_{IP} in LBP and ERM approaches for seven multicast trees generated in Network 3 and Network 4. The $MPLS_{comp}$ for these two approaches is shown in Figures 3.25 and 3.26.

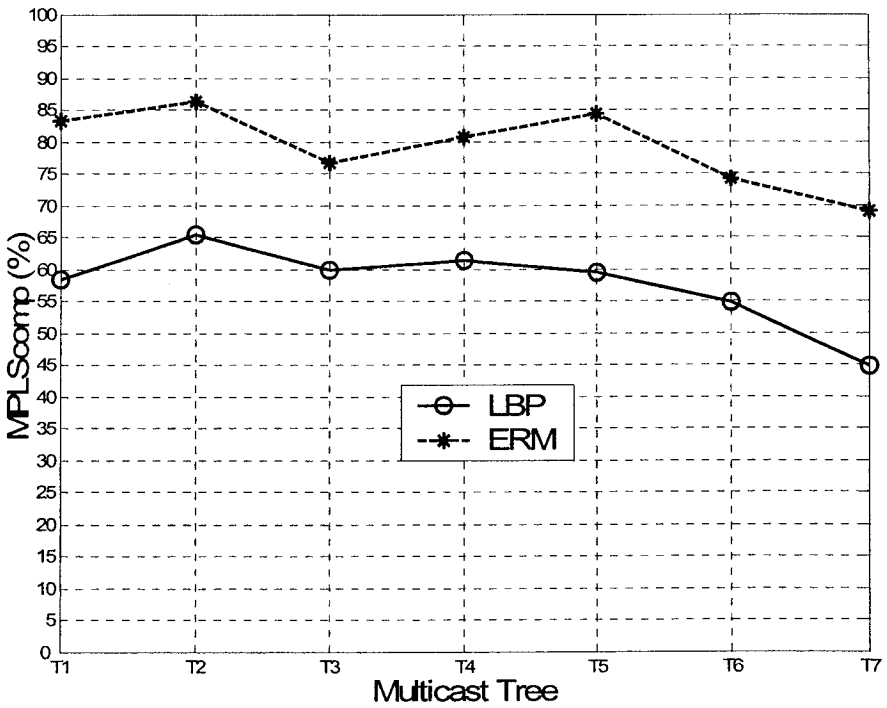


Figure 3.25 $MPLS_{comp}$ for the multicast trees generated in Network 3.

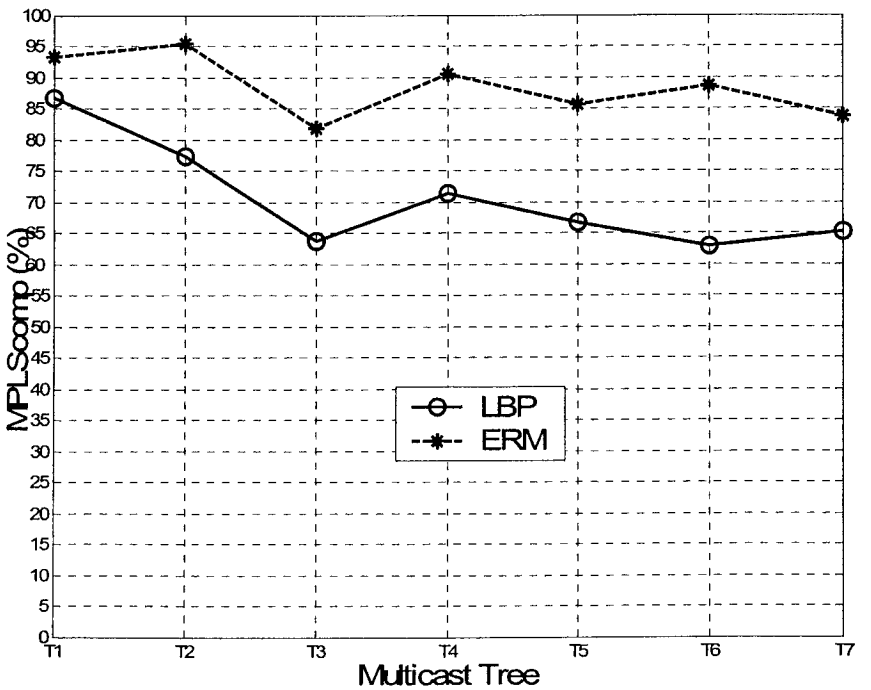


Figure 3.26 $MPLS_{comp}$ for the multicast trees generated in Network 4.

3.6.4 Discussion of the Results

From Figures 3.14 through 3.20, it is clearly shown that label aggregation approaches (TNT, SE, TN) provide a significant saving in the usage of the labels. However, these approaches maintain different memory sizes at the ingress LER. As shown in Figures 3.21 through 3.24, SE and TN approaches outperform the TNT approach. In TNT approach, the memory size increases rapidly than the SE and TN approaches. Furthermore, the TN approach uses less memory than the SE approach when the number of egress LERs in the tree increases as shown in Figures 3.23 and 3.24. A typical MPLS network has large size and the average number of LSRs and egress LERs for each tree could be in the range of hundreds, which makes the adoption of TN approach the best choice if the memory size at the ingress LER is a significant factor.

On the other hand, while SE and TN approaches are completely compatible with the MPLS protocol, LBP and ERM always record less compatibility with MPLS, which may reach under 50 percent as the case in T7 of Figure 3.25. Furthermore, LBP could reach the worst case when the value of $MPLS_{comp}$ reaches zero. This happens when the value of the N_{IP} is equal to the value of N_{LSR} as in the multicast tree shown in Figure 3.27. In that case, the speed of packet forwarding based on switching paradigm is completely cancelled and a recourse to layer 3 to use traditional IP routing is activated at every LSR over the tree.

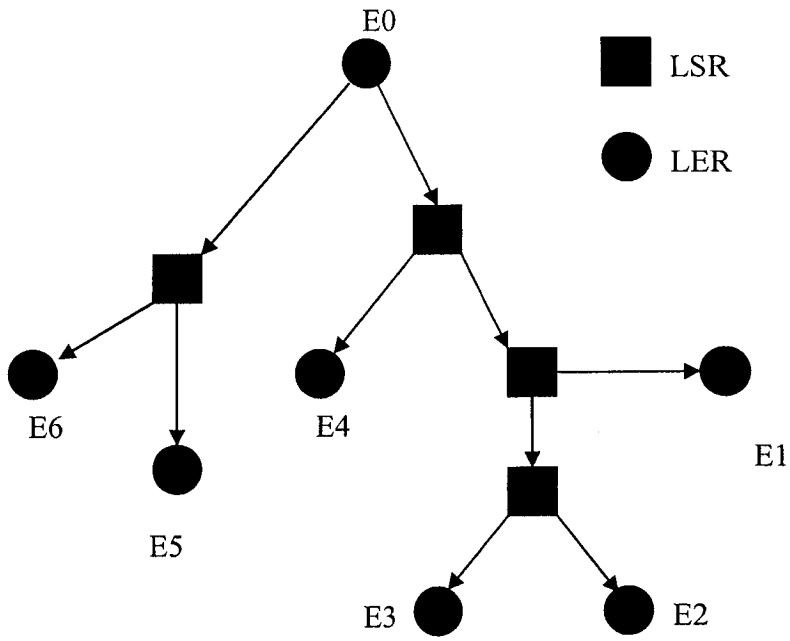


Figure 3.27 An MPLS tree with $MPLS_{comp}$ equal to 0 in LBP.

3.7 Complexity of SE and TN Approaches

In this section, we discuss the complexity of SE and TN approaches in terms of the time required to calculate the SE or the TN and in terms of the memory resources needed at each router to maintain the state code table in SE approach or the weighting table in TN approach.

In both SE and TN approaches, the time it takes to calculate the tree code if SE approach is used or the tree number if TN approach is used depends generally on the longest path in the corresponding tree. Another factor that contributes in the time is the number of outgoing interfaces at each router. Each router will not send the sub-code or

the partial weight value to the upstream router until it receives the sub-codes or the partial weight values from the downstream routers.

The memory resources required for maintaining the state code table or the weighting table at each router depend on the number of interfaces of that router. If router R has n interfaces, then a state table of each interface as an incoming interface should be maintained. The number of entries (i.e., number of states) in each table will be 2^{n-1}. Then, the size of the state tables will consume $n \times 2^{n-1}$ entries. In TN approaches, there is another table for each incoming interface. That table maintains the number of egress LERs downstream from each interface. This makes the size of the memory needed in TN approaches equal to $2n \times 2^{n-1}$.

3.8 Summary

We have proposed two efficient schemes for the multicasting scalability problem in MPLS networks. These solutions are called State Encoding (SE) and Tree Numbering (TN). MPLS labels are significant resources in MPLS networks. It is advantageous if different multicast sessions can share the same multicast tree in the network and re-use the MPLS label. SE and TN schemes support the aggregation of different multicast flows that use the same tree to deliver their IP packets in the MPLS network. In terms of the memory size required to maintain the tree codes or the tree numbers, the proposed schemes outperform the TNT approach, which also supports MPLS label aggregation for multicasting.

One of the objectives behind introducing MPLS is to speed up the forwarding process of the IP packets inside the MPLS networks. That objective is achieved by relieving the LSRs from the heavy processing of the IP packets, which is left for the LERs. Unicast communication is fully compatible with that objective. Once a LSP is set up, all the IP unicast packets that use that path are switched based on the labels they carry. In multicasting, most of the delivery trees have branching points. At each branching point, a multicast packet is duplicated and sent over each link of that point. Some approaches that have been proposed to support multicasting in MPLS networks, require the recourse to the traditional IP forwarding. As the number of branching points in a certain tree increases, the traditional IP forwarding increases too. For the purpose of measuring the compatibility of any proposed approach with the MPLS protocol, we have proposed $MPLS_{comp}$ as metric in order to give an indication to that compatibility. While SE, TN and TNT approaches are fully compatible with the MPLS protocol, some approaches such as LBP and ERM mostly record less than that.

Chapter 4

MPLS-based Failure Recovery

This chapter is the background for the next two chapters. We give an introduction to network survivability. A discussion for failure recovery at network layer and lower layers is presented in Section 4.2. The motivation, the objectives and the terminologies of MPLS-based recovery are then discussed in Section 4.3. Section 4.4 is dedicated to present the work that has been done for proposing MPLS-based schemes both in unicast and multicast communications.

4.1 Network Survivability

In networking, survivability is defined as the ability of a network to withstand and recover from failures [44]. With the growing development of the Internet in its new services and applications, survivability has become one of the most important requirements in networks. A survey on the design of survivable network architectures and recovery approaches for the different transport technologies is given in [45]. Any failure

of any part in a network will cause an interruption of a service. Service interruption will leave unacceptable consequences to the end users. For this reason, preventing service interruption and reducing the loss of service to a minimum in the case of inevitable interruption is an essential issue. In general, two types of failures are considered when network survivability is addressed: link failure and node failure. While node failure occurs as a result of equipment failure at network routers, link failure is usually related to cable cuts. Some of the reasons that lead to the interruption of transmission in a network are: damage as a result of building or because of fire, environmental events such as earthquakes and natural disasters [46]. Sometimes, intentional breakings of links can be necessary to carry out maintenance or expansion operation.

Backup paths (BPs) must be built to protect any element of the working paths (WPs). In general, two types of failures modes are considered when network survivability is addressed [47, 48]:

1. Single link failure mode: in this mode, a backup path is provided to protect the working path against all single link failures.
2. Single element failure mode: a backup path is provided to protect the working path against all single link and node failures.

When a links fails, the two nodes at the end points of that link detect the failure. In the case of a node failure, it is assumed that all the link interfaces at that node fail and therefore all the links that are incident on the node fail. For any WP, a single node failure model takes care of all the single link failures except the last link. For that

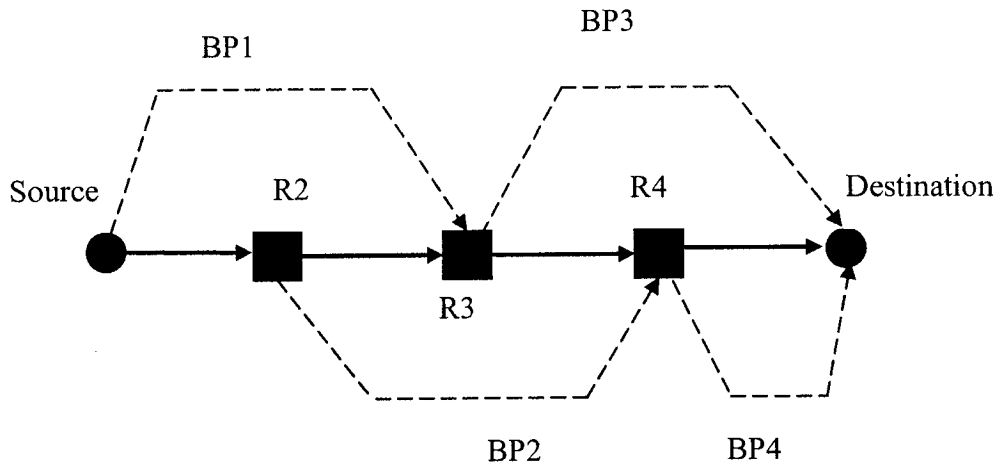


Figure 4.1: Backup paths in single element failure model.

reason, when a single node failure model is used, a BP is built to protect each node in the WP between the source and the destination, and a separate BP is built to protect the last link in the WP as shown in Figure 4.1. This makes the number of BPs required to protect the WP equal to $n + 1$, where n is the number of nodes between the source and the destination.

4.2 Failure Recovery at Different Layers

Current IP networks are connectionless. They use a best-effort approach to deliver data packets from the source to the destination. This means a least-cost route is calculated based on the routing information distributed by the routing protocol. If there is a network failure, some packets may be lost or dropped. However, the change in the network topology due to a link or node failure is distributed by the routing protocol and after a

period of time, the routing tables are updated and stabilized at each node in the network. This period of time is in the range of several seconds, which is not suitable for critical-time applications that impose a recovery time from a failure within milliseconds (60 ms is the common standard for the longest acceptable disruption to voice traffic). For example, Open Shortest Path First (OSPF) [15], a link state routing protocol has three timers affecting the failure detection: The Hello interval, the Router Dead interval, and the Advertisement interval. The accumulation of these time intervals may reach to several seconds.

Moreover, multicast routing protocols that rely on an underlying unicast routing protocol are as survivable as the underlying unicast routing protocol. For example, an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP) [12] may employ unicast routes in the underlying network to perform multicast routing. Thus, in the event of a failure, if the unicast routing tables are updated appropriately, DVMRP will also function correctly. On the other hand, if a multicast routing protocol is independent of the unicast routing protocol, it must implement its own restoration mechanism.

Failure recovery in networks can be implemented at the lower layers. For example, current backbone networks rely on SONET/SDH, which is a physical layer technology used in optical transmission [44, 49, 50]. In SONET/SDH, protection is done based on self-healing rings. Although SONET/SDH has the capability to provide service restoration within less than 50 milliseconds, it has the disadvantages that the scope of the protection is limited and it is expensive and requires specific hardware. In addition to that, it wastes up to half of the available bandwidth in the case it will provide full redundancy. A study of survivability in multilayer transport networks is presented in [51].

A mechanism that trades off between the slow recovery time provided by the network layer and the expensive cost provided by the lower layers should be used. Fortunately, MPLS as a connection-oriented approach achieves that objective. The slow time is reduced by building BPs before a failure occurs. In MPLS-based recovery, as will be discussed in Section 4.3, establishing BPs in advance before a failure takes place is called pre-planned recovery or protection switching.

4.3 MPLS-based Recovery Framework

The limitation on improving the recovery time of current routing algorithms is one of the basic motivations behind developing a standard framework for MPLS-based recovery [52]. In addition, many objectives should be taken into consideration when proposing any MPLS-based recovery scheme. Some of the objectives for MPLS-based recovery are:

- It should maximize network reliability.
- It should be applicable for entire end-to-end path or for segments of an end-to-end path.
- It should minimize the loss of data and packet reordering during recovery operations.
- It should minimize the signaling overhead to set up and maintain recovery paths.
- It should preserve the constraints on traffic after switchover.

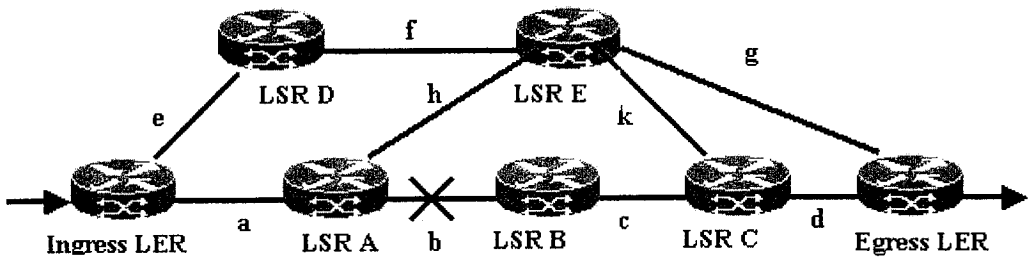


Figure 4.2: MPLS-based recovery.

MPLS-based recovery introduces two models for path recovery, either rerouting or protection switching. While the former provides traffic protection by establishing new paths or segments of path on demand after the occurrence of failures, the later relies on mechanisms that pre-establish a recovery path or path segments based on many factors such as routing policies and administrative considerations.

As defined before, the protected path that carries traffic before the failure takes place is called the working path while the path that carries traffic after recovery is called recovery path or backup path. For example, in Figure 4.2 the path (a, b, c, d) may be the working path and the path (e, f, g) is the recovery path. An LSR that is responsible for switching the traffic between the working path and the recovery path is called a Path Switch LSR (PSL) and the LSR that is responsible for merging the traffic back onto the working path is called a Path Merge LSR (PML). The ingress LER in Figure 4.2 is the PSL for the path (a, b, c, d) while LSR C or the egress router could be the PML.

Recovery can be local or global. Local recovery is used to protect segments of the working path and it is used to protect against link or node failures. In local recovery, the node immediately upstream of the fault will be the PSL. For example, in Figure 4.2 when

(a, b, c, d) is the working path and a failure takes place either for link b or for LSR B, LSR A will be the PSL that initiates the recovery process and in this case path (h, k) will be the recovery path and LSR C will be the PML, or path (h, g) could be the recovery path and the egress LER will be the PML. The main advantage of local recovery is that it minimizes the amount of time required for failure propagation.

On the other hand, global recovery is used to protect traffic against any link or node failure on a path. Global recovery is called end-to-end path recovery. In Figure 4.2, when path (a, b, c, d) is the working path, the end-to-end global recovery path will be (e, f, g). In global path recovery, the ingress LER will be the PSL and the egress LER will be the PML. If global path recovery is used, the PSL must be notified by a special signal called a Fault Indication Signal (FIS). This may make global path recovery slower than local path recovery especially if the link or node failure is far from the PSL.

Sometimes, it is not necessary that the PSL be different from the node that detects the failure. In this case, the PSL is itself the point that detects the failure and then directs the MPLS data to the pre-established recovery path. This process is known as fast reroute and has the advantage that it cancels the time required to propagate the error to the PSL. A comparison between protection switching and fast reroute options is found in [53].

The resources on the recovery path may be used to carry either a copy of the working path traffic or a low priority traffic that is displaced when a protection switch takes place [54]. Based on that, path protection switching is classified into two variants. The first one is called 1:1 protection switching. In 1:1 model, the recovery path is pre-established and low priority traffic is carried on this path. When a failure on the working path occurs, the low priority traffic is displaced and the protected traffic is directed to the recovery path.

Typically, the 1:1 protection switching is initiated by a PSL. The second type of path protection switching is called 1+1 path protection. In the 1+1 model, the resources on the recovery path are fully reserved, and carry the same traffic as the working path. The selection between the two traffic flows is made at the PML.

4.4 Related Work

In this section, we present some of the schemes that have been proposed to employ MPLS-based recovery in unicast communication. Then we discuss the schemes that have been suggested for designing MPLS-based recovery models in multicast communication.

4.4.1 MPSL-based Recovery in Unicast Communication

A lot of research has been done in the field of designing MPLS-based models for unicast communication. A study of label switching networks is given in [55]. The author focuses on ATM and MPLS networks. In order to show how to implement the path restoration strategy, the study targets three objectives: effective control, fast restoration and economic network capacity placement where the capacity is related to the amount of bandwidth used to transmit the data over a certain link. In [56], a restoration technique has been proposed to reroute a problematic tunnel to a backup one by concatenating the two tunnels at the local node. In [57], a design for a fast recovery mechanism in MPLS networks is proposed, where the objective is to quickly respond to network faults and reduce loss as much as possible. To achieve the intended objectives, the author proposes a centralized scheme where he assumes that the network topology, the primary, and the

backup tunnels are pre-established at the ingress router. The proposed approach uses the RSVP-HELLO extension in order to detect a link failure at the MPLS level.

In [58] and [59], the authors propose algorithms that aim at providing a guaranteed restoration using MPLS local recovery. An algorithm for dynamic routing for local restoration is presented in [48]. An MPLS restoration scheme that divides a primary LSP into several segments is proposed in [60]. The authors in [61] propose an approach that deals with dynamic reconfiguration of spare capacity for MPLS-based recovery in the Internet backbone networks.

4.4.2 MPSL-based Recovery in Multicast Communication

As we have discussed before, the two common protection switching recovery mechanisms are local protection and global (end-to-end) protection, and variant schemes that fall between these two common models are also proposed as discussed above. However, and to the best of our knowledge, little research has addressed the issue of designing recovery models for multicasting in MPLS networks. Indeed, the issue of building backup paths to protect a multicast tree is not similar to the case in unicast communications. Building BPs to protect a multicast trees is not as easy as protecting a LSP that carries unicast traffic. Some multicasting-specific properties should be considered when building backup paths for a multicast tree. Some of these properties are:

1. While unicast communication is one-to-one approach, multicasting could be one-to-many or many-to-many approaches. From this property, it can be said that any recovery algorithm that would be used to protect a multicast tree should reduce the number of users dropped from the communication when a failure takes place.

2. Multicast communication has a dynamic behavior, which means that the number of users is not constant along the life period of the multicast session. A change in the number of group members could take place from time to time when new users join a multicast session or current users leave the session. Any proposed MPLS-based recovery for multicasting should take care of the tree maintenance when the topology of the tree changes.
3. Distribution of the egress LERs of a certain multicast tree over a backbone MPLS networks) is an attractive property. For example, distribution of the egress LERs determines the type of the recovery whether it should be local or global. Figure 4.3, for instance, shows three multicast trees. Each tree has four egress LERs. A different recovery model is built for each tree depending on the distribution of the egress LERs.

The last property leads us to define the types of the BPs that can be built for a multicast tree. These types of BPs are given in Table 4.1, and Figure 4.4 gives an example for each type.

Table 4.1: Types of the BPs that can be built in a multicast tree.

Type	Example
Ingress LER to LSR BP	BP1
Ingress LER to egress LER BP.	BP2
LSR to LSR BP.	BP3
LSR to egress LER BP	BP4
Egress LER to Egress LER BP.	BP5

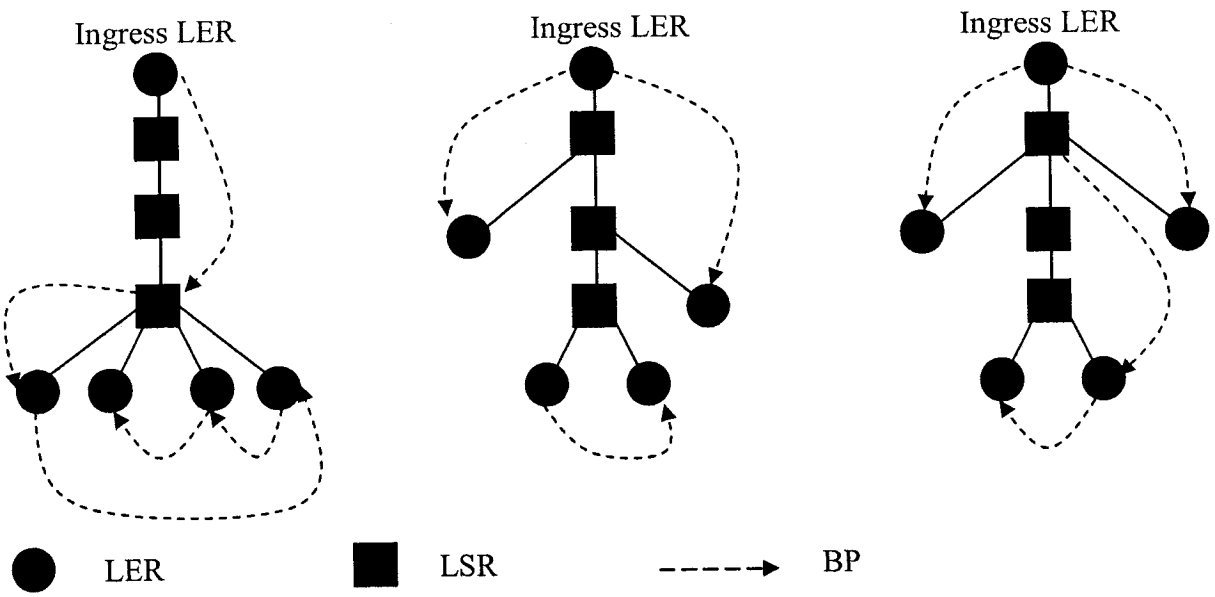


Figure 4.3: Different recovery models depending on the distribution of the egress LERs over the multicast tree.

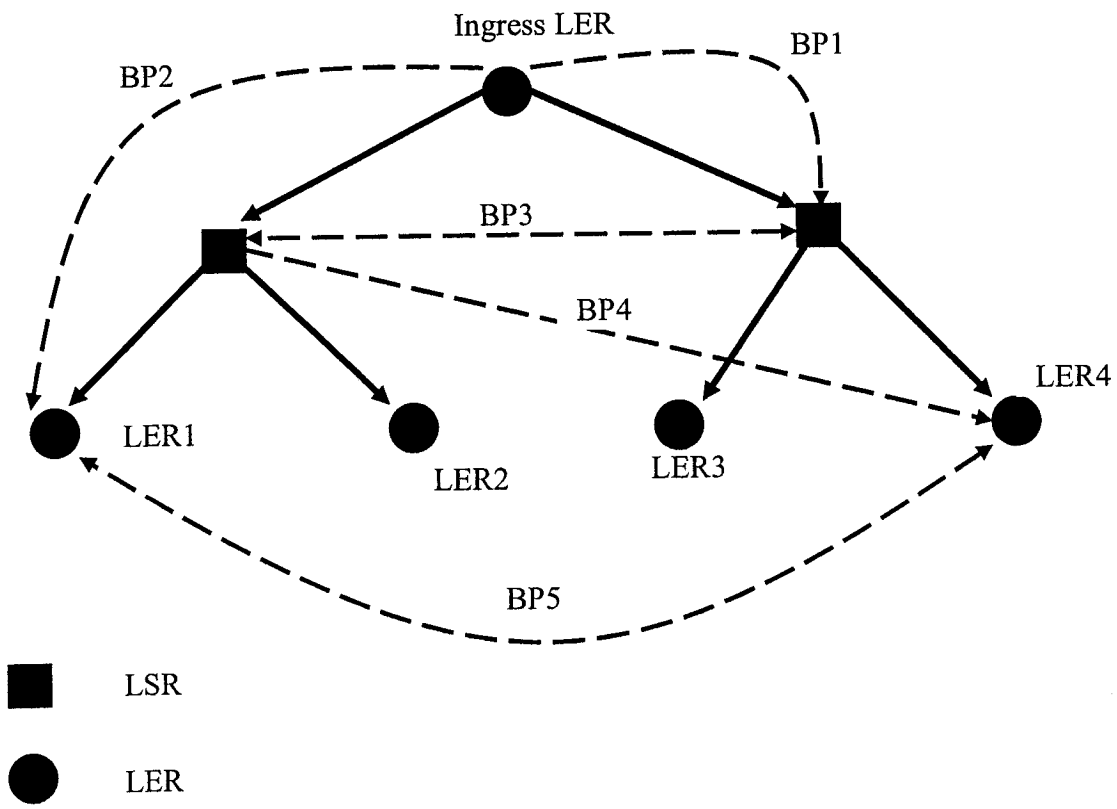


Figure 4.4: Different types of BPs in a multicast tree.

In [62], the author proposes a graph algorithm that takes the first property mentioned above into consideration. The proposed algorithm selects a backup path in a multicast routing tree that minimizes the number of receivers dropped from the communication when a failure in a multicast delivery tree occurs. The failure mode that was assumed is the single link failure. A heuristic algorithm that supports on-line routing of bandwidth guaranteed multicasting is proposed in [63]. For the purpose of setting up BPs, the authors consider the single local recovery model (see Figure 4.1) for a multicast tree. The primary objective of the proposed algorithm is the minimization of the backup capacity required to set up the backup paths. In order to study the performance of the proposed algorithm, a set of experiments is set up in two networks topologies under the assumption of different information models.

An MPSL-based architecture is proposed in [64]. The authors employ the concept of multicast aggregation [65] in the design of their proposed architecture. The idea in that concept is to "force" several multicast groups to share a single distribution tree. That concept is extended to find a backup recovery tree for each primary aggregated tree. If a failure occurs in the primary tree, all the multicast traffic that is assigned to that tree is switched to the recovery tree. The advantages of that architecture are its efficiency and scalability. However, its drawback is that it is not always guaranteed to find edge disjoint trees. The authors in [66, 67] use the concept of segmentation defined in [60] and apply that concept in multicasting. The BPs are built between the branching point of a multicast tree as shown in Figure 4.5. Segmentation-based recovery is considered as a method that supports local recovery for multicasting.

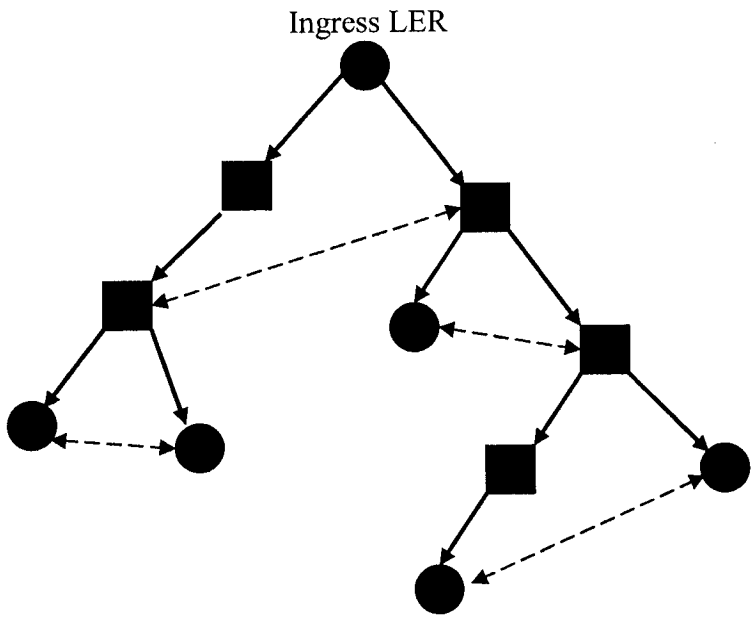


Figure 4.5: Segmentation-based recovery in a multicast tree.

4.5 Summary

We have discussed the importance of network survivability in current backbone networks. Also, we have explained the tradeoff between providing recovery at the network layer and at the lower layers. The motivations, the objectives as well as the terminologies of the MPLS-based framework are discussed. The research that has been done based on MPLS for unicast and multicast communications is presented.

In the next chapter, we propose an MPLS-based recovery architecture. We discuss the main steps that compose the architecture. It is worthwhile to mention at this point that the proposed methods that are presented in the next chapter will be compared in Chapter six with the schemes suggested in [66, 67].

Chapter 5

MPLS-based Recovery Architecture for Multicast Trees

5.1 Introduction

This chapter presents an MPLS-based recovery architecture for multicast communication. The architecture is composed of a set of steps, which are shown in Figure 5.1. The main objective behind this architecture is to support a local failure recovery model in multicast communication such that the notification time after the occurrence of the failure is reduced. Assume a multicast tree T = {E0, SEL} is deployed over an MPLS network where E0 is the ingress LER, SEL = {E1, E2,..., Ej} is the set of egress LERs and j is the number of egress LERs of tree T. The first step in the architecture is to collect the tree topology, T_{top}. The collection starts from the egress LERs and goes backward to the ingress LER. For this purpose, the same messages that are used to collect the tree code or the tree number (see Chapter 3) can also be used to piggyback information on T_{top}.

When the ingress LER receives T_{top}, the next step of the architecture is triggered. In this step, a tree division algorithm is run to divide the multicast tree T into several domains. Each domain is considered as a stand-alone tree. The detailed discussion of the tree division algorithm is given in Section 5.2. The next step is the distribution of the topologies of the domains to the corresponding routers. The purpose of this step is to let each router know its type (the definition of each router in tree T is given in the next section) and to which domain it belongs.

The last step in the architecture is the setup of the backup paths inside the domains. Two methods are proposed for this purpose. The backup paths fall in the category of preplanned (protection switching) approach, which means a sufficient backup capacity should be reserved in each link of the backup paths. Note that the capacity is the amount of the bandwidth that should be reserved over each link in the backup paths. Section 5.3 explains the two methods used to set up the backup paths in addition to the discussion of the notification and activation mechanism used inside the domains,

Before we start the detailed discussion of the architecture, we point out the following two assumptions:

1. Only a single failure can take place at any time in the network [44]. In other words, a single failure model is assumed.
2. We assume that no failure will take place until the backup paths are set up in all the domains.

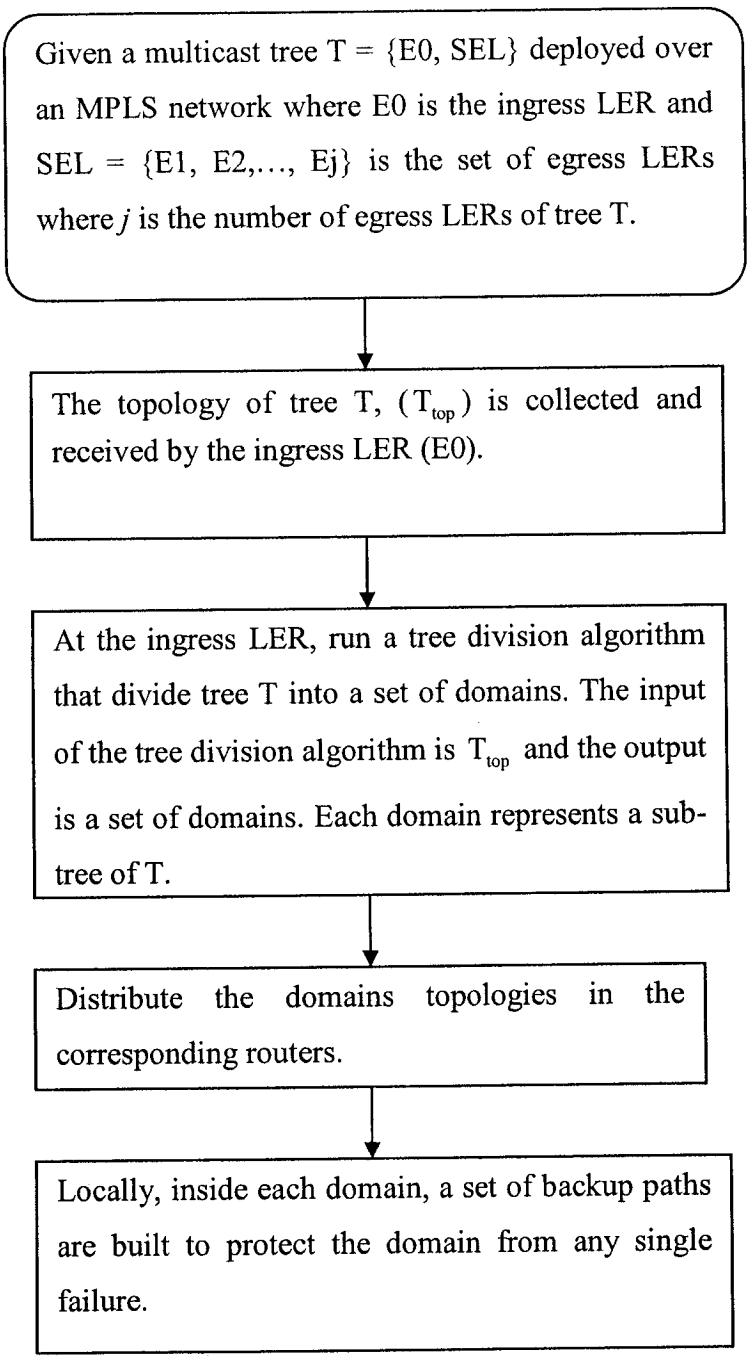


Figure 5.1 The main steps of the architecture that provides the MPLS-based recovery for multicast communication.

5.2 Tree Division Approach

In large backbone networks such as MPLS networks, depending on the global recovery model to restore from a link/node failure is not an efficient solution, especially if there is a significant consideration for QoS parameters such as delay. In this section, we introduce an approach to divide the multicast tree into several domains. Each domain is considered as a sub-tree of the original multicast tree. The root of each domain is called domain root (d^{root}), and each leaf of the domain is called a border router (BDR).

Assume a network represented as a graph $G = (N, L)$ where N is the set of nodes (routers) in G and L is the set of links that connect N together. Assume a multicast session is deployed over the network. This session builds its multicast tree $T = (NT, LT)$, where NT represents the routers of the tree and LT is the set of links that connects NT.

Definition 5.1 For each router $R \in NT$, we define the outgoing degree of R, $od(R)$, as the number of downstream routers that are directly connected to R.

Definition 5.2 A router R that has outgoing degree greater than 1, (i.e., $od(R) \geq 2$), is called a Branch Router (BR).

Definition 5.3 A router R that has outgoing degree equal to 1, (i.e., $od(R) = 1$), is called an intermediate router (IR).

Definition 5.4 A path segment in tree T that connects two BRs is called a Segment between BRs (SBR).

Definition 5.5 The length of SBR, l_{SBR}, is defined as the number of links between the two BRs connected by that SBR. Any SBR is called a partitioning segment if its l_{SBR} is greater than 1 (i.e., there is at least one IR between the two BRs connected by that SBR).

In MPLS networks, the root of tree T (T^{root}) is the ingress LER (E0) and SEL = {E1, E2,..., Ej} is the set of egress LERs where j is the number of egress LERs of tree T. The algorithm that divides the tree into domains starts from the T^{root} (E0) and ends when all the routers in the tree T are tested. A router is allocated to the current domain if it is an IR. A router is considered a BDR if it is an egress LER or if it is a BR and its parent router is an IR. A router is considered as a root of a new domain if it is an IR whose child router is a BR. Two main advantages are achieved by this division:

1. This division of the tree takes into consideration both link and node failures (single failure model).
2. The division makes the restoration mechanism local, which results in largely reducing the propagation time of the signaling message.

Figure 5.2 shows an MPLS tree divided into six domains. Note that there is an overlap between a domain and its downstream domains. This overlap takes place in the link that connects the BDR router of the domain with its upstream IR router. These two routers maintain the topologies of the two domains they belong to. Within a certain domain, the root and the BDRs of the domain are considered non-prone to failures.

Any partitioning SBR in a tree increases the number of domains by one. Thus, the number of domains that can be produced from tree the division algorithm is given as

$$ND = N_{SBR} + 1 \tag{5.1}$$

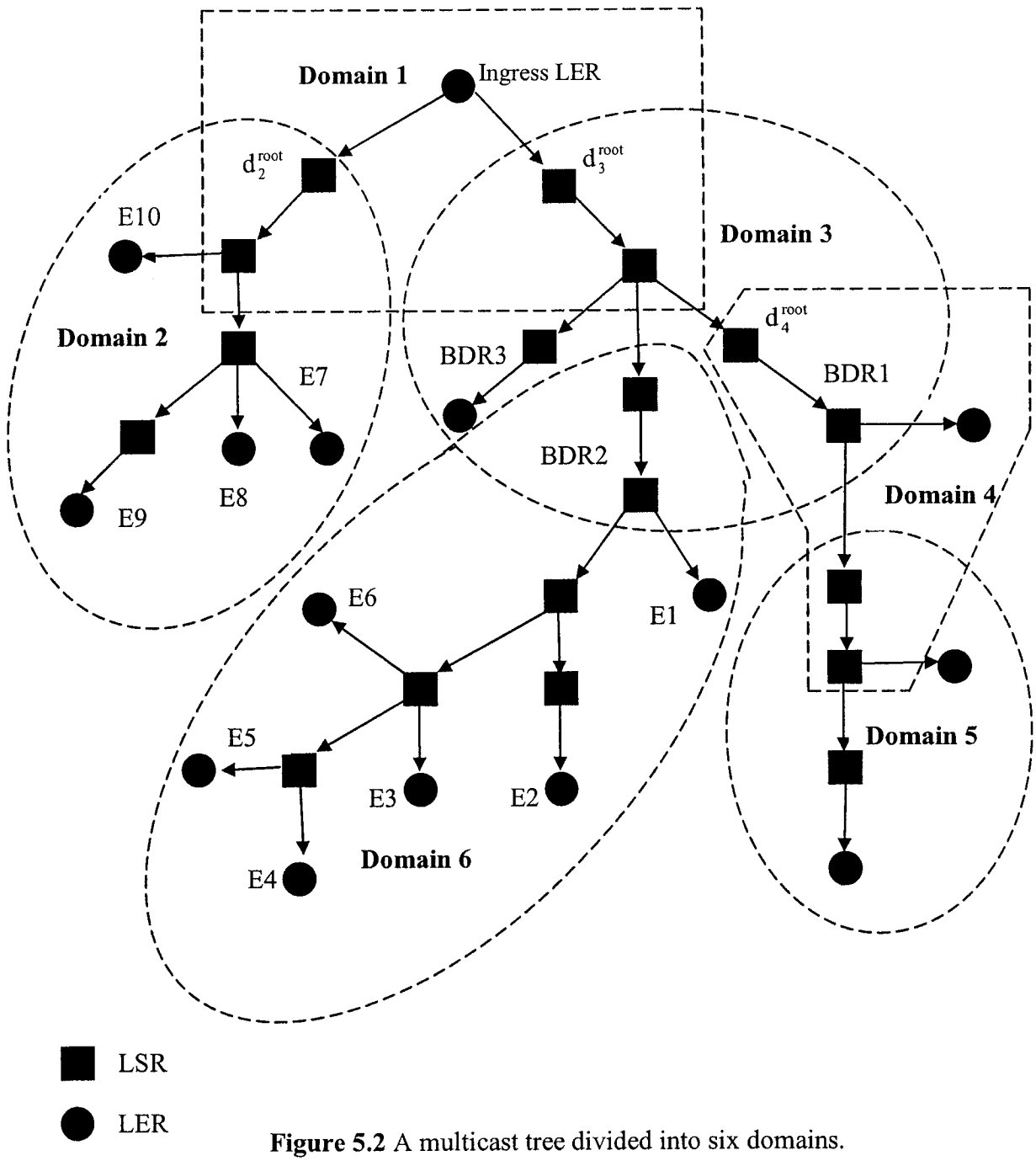


Figure 5.2 A multicast tree divided into six domains.

where ND is the number of domains and N_{SBR} is the number of partitioning SBRs. N_{SBR} itself depends on the number of BRs (see Definition 5.5).

5.2.1 Tree Division Algorithm

Figure 5.3 shows the algorithm used to divide a multicast tree T into several domains. The input of the algorithm is tree *T = (NT, ET)*. RSet and CSet are two local variables. RSet is a set that maintains the nodes that are candidates to be roots for the domains. The initial value of RSet is the ingress LER E0. CSet is a set used to contain the child routers of the node under test. When a node is chosen to determine its type (IR, BDR, BR), its child routers are added to CSet. The algorithm uses the following functions; their full codes are not shown in the algorithm:

1. ***get-node-from(***RSet***):*** this function is used to return a node from the RSet. After that, the node is removed from the RSet. The use of this function is shown in lines 8 and 47 of the algorithm
2. ***get-node-from(***CSet***):*** this function is used to return a node from the CSet. After that, the node is removed from the CSet. The use of this function is shown in line 17 of the algorithm.
3. ***Add-children(***nodeB,CSet***):*** this function is used to add the child routers of a node to CSet. This function uses the tree topology *T = (NT, ET)* to get the child routers of the node under test. This function is used in lines 27, 40 and 51 of the algorithm.

```
1   // RSet = {Contains the nodes that are candidates to be roots for the sub-trees}
2   // CSet = {Children of the node under test}
3   // the initial value of RSet is E0.
4
5   begin
6     i = 1
7     d[i] = new (domain);
8     nodeA = get-node-from(RSet);
9     RSet ←RSet/{nodeA}; // remove nodeA from RSet
10    d[i].root = nodeA
11    d[i].Add-to-NodesList (nodeA)
12    Add-children(nodeA, CSet)
13    Do
14     Begin
15        while CSet ≠ Φ do
16         Begin
17            nodeB = get-node-from(CSet);
18            CSet ← CSet/{nodeB}; // remove nodeB from CSet
19            if (od(nodeB) = 0 ) // nodeB is a LER
20               begin
21               d[i].Add-to-BorderNodesList(nodeB)
22               d[i].Add-to- NodesList (nodeB)
23               End
24            if (od(nodeB) = 1 ) // nodB is IR
25              Begin
26              d[i].Add-to- NodesList (nodeB)
27              Add-children (nodeB, CSet)
28              End
29          if (od(nodeB) > 1 )
30            Begin
31            nodeC = get-parentNode(nodeB)
32            If (od (nodeC) = 1)
33               begin
34               Add-node(nodeC, RSet)
35               d[i].Add-to-BorderNodesList(nodeB)
36               End
37            else
38               begin
39               d[i].Add-to- NodesList (nodeB)
40               Add-children(nodeB, CSet)
41                end
```

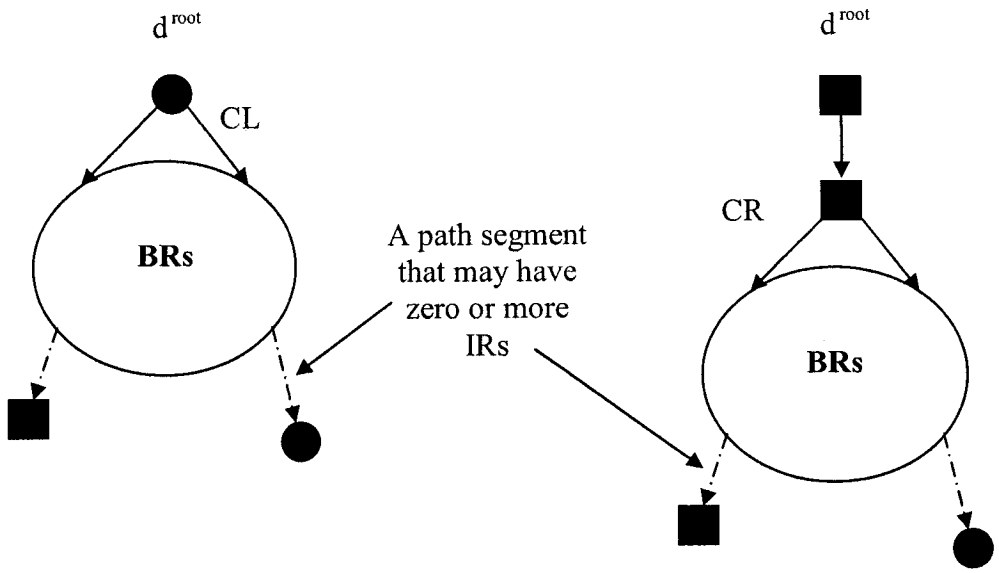
Figure 5.3 An algorithm that divides multicast tree T into several domains.

```
42              End
43      end  // End of the second while statement (line 16)
44  loop_1:
45  i++;
46  t[i] = new (domain)
47  nodeD = get-node-from(RSet);
48  RSet ←RSet/{nodeD}; // remove nodded from RSet
49  t[i].root = nodded
50  nodeA=get-child(nodeD)
51  Add-children(nodeA, CSet)
52  t[i].Add-to-NodesList (nodeD)
53  t[i].Add-to-NodesList (nodeA)
54  while (RSet!={} OR (CSet!={}) // End of the do statement (line 13)
55  end // End of the entire code
```

Figure 5.3 An algorithm that divides multicast tree T into several domains (continued).

The output of the tree division algorithm given in Figure 5.3 is a set of domains d[i] for $1 \leq i \leq ND$ where ND is the number of domains. Each domain is represented by three data components: the domain's root, the NodesList and the BorderNodesList. After dividing the tree, the ingress LER sends the topology of each domain to the root of that domain. Each domain's root then should inform the routers in that domain about the topology of the domain they belong to. At the end of this step, each router knows whether it is a BDR, d^{root} or just IR.

Each domain can take one of two patterns. These patterns are shown in Figure 5.4. Pattern 1 is produced when the d^{root} is the ingress LER of the MPLS tree. Pattern 2 is the general one and is produced when the d^{root} is an IR. In Figure 5.2, domain 1 is an example of pattern 1 while all the other domains are examples of pattern 2.



(a) Pattern 1 (b) Pattern 2

Figure 5.4 The two patterns that could be produced by tree division algorithm.

Definition 5.6 In pattern 2, the BR connected directly to the d^{root} is called the Critical Router (CR).

Definition 5.7 In pattern 1, each link connected directly to the d^{root} is called a Critical Link (CL).

Note that the failure of any CL in a certain domain which has pattern 1 results in the disconnection of all the BDRs in that domain connected to the d^{root} through that link. Also, the failure of a CR in a domain that has pattern 2 results in the disconnection of all the BDRs in that domain.

5.2.2 Maximum Number of Domains

Definition 5.8: The degree of any router R in tree T is equal to $od(R)+1$.

Let N_i be the number of nodes that have a degree equal to i in tree T.

Theorem 5.1 The number of leaves of a nontrivial tree is given by the formula

$$N_1 = 2 + \sum_{i=3}^{\infty}(i-2)N_i \tag{5.2}$$

Proof. See [68, pp.88]

Theorem 5.2 If a multicast tree has N_{LER} egress LERs, then the maximum number of domains that can be generated by applying the tree division approach introduced in the algorithm in Figure 5.3 is given as $ND = N_{LER}$, where ND is the number of domains.

Proof. If the multicast session has N_{LER} egress LERs, then these routers can be connected together as a tree in different ways. The simple tree that could be built to connect these routers together is the star tree where all the LERs are connected to one central router, which in turn is connected to the ingress LER as shown in Figure 5.5(a). In this case, the number of domains is $ND = 1$. If the path from the ingress LER to the central router contains at least one IR router, then ND is equal to 2 as illustrated in Figure 5.5(b).

In order to get the maximum number of BRs over a tree, the maximum degree for any BR in that tree should not be greater than 3. In this case, the multicast tree has only three types of routers: the routers which have a degree equal to 1 (the leaf routers) and the routers which have a degree equal to 2 (The IR routers) and the routers that have a

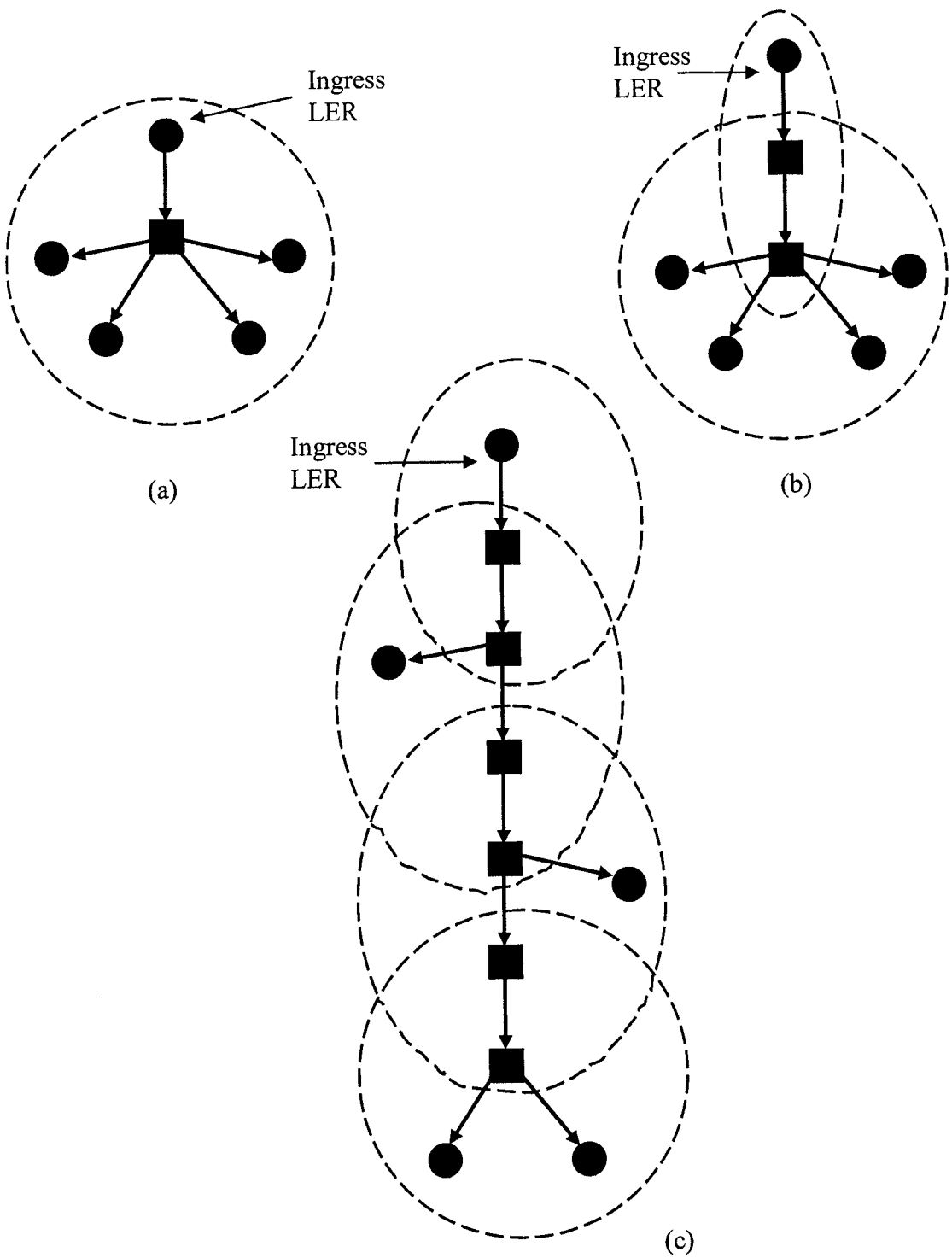
degree equal to 3 (The BR routers). By expanding the formula in Theorem 5.1, we get the following equation:

$$N_1 = 2 + \sum_{i=3}^{\infty} (i-2)N_i \tag{5.3}$$

Note that the term N_2 is not originally presented in Equation (5.2). Furthermore, the terms N_i for $i \geq 4$ will not appear in the tree under assumption. Then, Equation (5.3) can be rewritten as

$$N_3 = N_1 - 2 \tag{5.4}$$

where in this case N_3 represents the number of BRs and N_1 represents the number of leaf routers. In order to get the maximum number of domains, the ingress LER should be connected to one BR. This condition means that the degree of the ingress LER is equal to 1. Thus N_1 is equal to $N_{LER} + 1$. And the number of BRs as given in Equation (5.4) will be $N_3 = N_{LER} - 1$. If there is at least one IR between any two BRs, then the number of partitioning SBRs is equal to the number of BRs, which is N_3. By Substituting the value of N_3 in Equation (5.1) we get the maximum number of domains as $ND = N_{LER}$. Figure 5.5 (c) shows the case when four egress LERs are divided into four domains, which represents the maximum number of domains in this case.



Figurer 5.5 A tree composed of four egress LERs can produce different domains.

(a) Four egress LERs divided into one domain.
(b) Four egress LERs divided into two domains.
(c) Four egress LERs divided into four domains.

5.3 Building the Backup Paths inside the Domains

In order to reduce the cost of reserving the backup paths inside the domains, Integer Linear Programming (ILP) optimization can be used. To achieve this ILP optimization, sufficient information about the used and the residual capacity of each link has to be available at each node in the network. The authors in [48] have proposed three types of information models that can be used to obtain the ILP optimization in our case. However, ILP-based optimization problems have been proved to be NP-complete [69]. For that reason, we propose two heuristic methods to set up the backup paths inside the domains. We present first the ILP model and then we introduce the two recovery methods that can be used to set up the backup paths inside the domains. A discussion of the detection and notification mechanism that is used in these methods is also presented.

5.3.1 Integer Linear Programming (ILP) Model

In order to get the optimal solution for reducing the reserved capacity using ILP model, a set of linear equations must be formulated that defines the problem. Recall that in Section 5.2, we have represented a network as a graph $G = (N, L)$, where N is the set of nodes (routers) in G and L is the set of links that connect N together. Moreover, we have assumed that a multicast session is deployed over the network. That session builds its multicast tree $T = (NT, LT)$, where NT represents the routers of the tree and LT is the set of links that connects NT. Furthermore, the following parameters are used in the equations for the linear model:

μ_{ij} Unit cost of link (i, j).

S_{ij} Spare capacity at link (i, j).

N_N Number of nodes in the network.

ND Number of domains that tree T is divided to.

$\mathrm{d}_i^{\mathrm{root}}$ The root of domain i.

We assume that the links are directed between the nodes (routers) and each link is identified by the two nodes connected by that link. For example, (1, 3) is a link that is directed from node 1 to node 3. N_{BDR}^i is the number of BDRs in domain i. Each domain has a set of working paths that connect the domain's root with the BDRs in that domain. The working path j in domain i is written as $WP_j^i = (v_{1j}^i, v_{2j}^i, ..., v_{mj}^i)$ where $v_{kl}^i \in \{1, 2, 3, ..., N_N\}$ and m is the number of nodes over that path. Note that $v_{1j}^i = \mathrm{d}_i^{\mathrm{root}}$ for all $1 \le i \le ND$.

On the other hand, the backup path that must be set up to protect the working path j in domain i is written as $BP_j^i = (y_{1j}^i, y_{2j}^i, ..., y_{nj}^i)$. For all the backup paths, the values of $y_{kl}^i \in \{1, 2, 3, ..., N_N\}$ are to be fixed by the ILP model. Hence, the objective function is:

$$Min\left\{\sum_{k=1}^{ND} \sum_{(i,j)\in\{1,2,...,N_N\}} \mu_{ij} .S_{ij}\right\}, \quad i \neq j$$

The constraints that must be met in this model are:

1. The spare capacity on the backup links must be integer and non-negative.

$$S_{ij} \in Z^+$$

If the demand request for the multicast session that belongs to tree T is equal to b_k, then the value of S_{ij} in the objective function should equal to b_k if link (i, j) is not

reserved before, otherwise, the value of S_{ij} will equal to zero. In [48], a shared protection is assumed and depending on the information available at link (i, j), the value of S_{ij} can range from zero to b_k.

2. $y^i_{nj} = v^i_{mj}$, for all $1 \le i \le ND$ and all the working paths in each domain. This constraint states that the last router in each backup path in all the domains should be the same as the last router in the corresponding working path.
3. $\{v^i_{2j},....,v^i_{m-1j}\} \cap \{y^i_{2j},....,y^i_{n-1j}\} = \Phi$, for all $1 \le i \le ND$. This constraint states that the working path and its corresponding backup path should be link-disjoint and node-disjoint except from the first node that could be the domain's root or the last node that must be a border router as in given in constraint 2.

5.3.2 Detection and Notification Mechanism

In order to detect link or node failure, we assume that routers in a network G regularly send small messages called *probes* on all the links on which they send traffic, and listen for probes on the links from which they receive traffic [62]. These messages consume a small bandwidth of the link on which they are sent. Furthermore, probe messages should be sent frequently at a high rate to keep the detection time fast. The faster the detection time is, the lower the total recovery time will be. If a probe message is not received on a link within a certain period of time, that link is assumed to have failed and a detection procedure on the node that detects the failure is activated. In the case of node failure, all the links incident to that node are considered failed links. A detailed analysis of the detection time based on probe messages is given in [62].

In the proposed methods, we assume the downstream router in the failed link that detects the failure is responsible for failure notification mechanism. In the case of node failure, all the child routers of that node are responsible for the failure notification mechanism. The notification procedure is composed of two parts:

1- Generation of notification messages: a router that detects the failure will generate a notification message. This message should be forwarded to all the BDR routers downstream from that router in the same domain.

2- When a BDR receives the notification message, it generates an activation message which is sent in the reverse direction of the backup path to the PSL. The purpose of the activation message is to activate the switching tables in all the routers along the backup path and to inform the PSL router to start switching the traffic over the backup path.

An illustration of a notification procedure is shown in Figure 5.6, which represents a domain with pattern 2. The domain has two BDRs: BDR2 is an egress LER while BDR1 is a BR in an original tree. R1 is the CR of the domain. When R1 fails, all the BDRs are disconnected from the d^{root}. In this case, BDR2 detects the failure when the probe message over link (R1, BDR2) is missed and R2 detects the failure when the probe message over link (R1, R2) is missed. Since BDR2 is directly connected to R1, it immediately generates the activation message in the reverse direction of its backup path (BP2) while R2 forwards the notification message to BDR1. When the notification message is received by BDR1, it generates an activation message to activate its BP1.

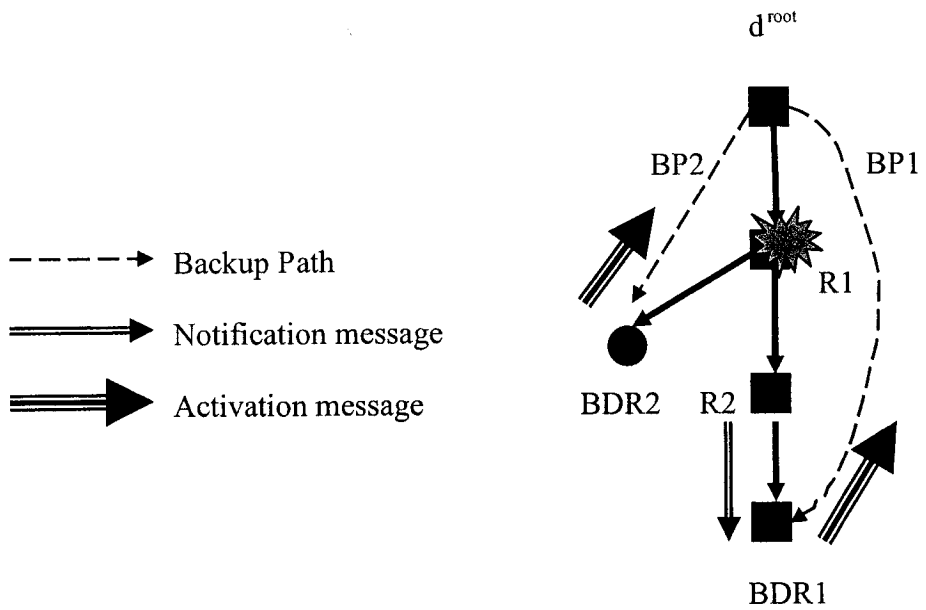


Figure 5.6 An illustration of a notification and activation procedure.

A flowchart that describes the detection and notification mechanism is shown in Figure 5.7.Once each domain is defined and its topology is known at all the routers in that domain, setting up the backup paths is triggered. In the next two sub-sections, we propose two methods to set the backup paths.

5.3.3 Method 1: Common PSL

In this method, the backup paths are built between the root of the domain and each BDR in that domain, i.e., the PSL is common for all the backup paths. Figure 5.8 shows domain 3 of the original multicast tree shown in Figure 5.2. This domain has three BDR routers. Note that BDR3 is an egress LER in the tree while BDR1 and BDR2 are BRs in the original MPLS tree. Domain 3 in Figure 5.8 has three working paths that connect d_3^{root} with the BDRs. These paths are: P1 = $\{d_3^{root} \rightarrow R1 \rightarrow R2 \rightarrow BDR1\}$, P2 = $\{d_3^{root} \rightarrow$

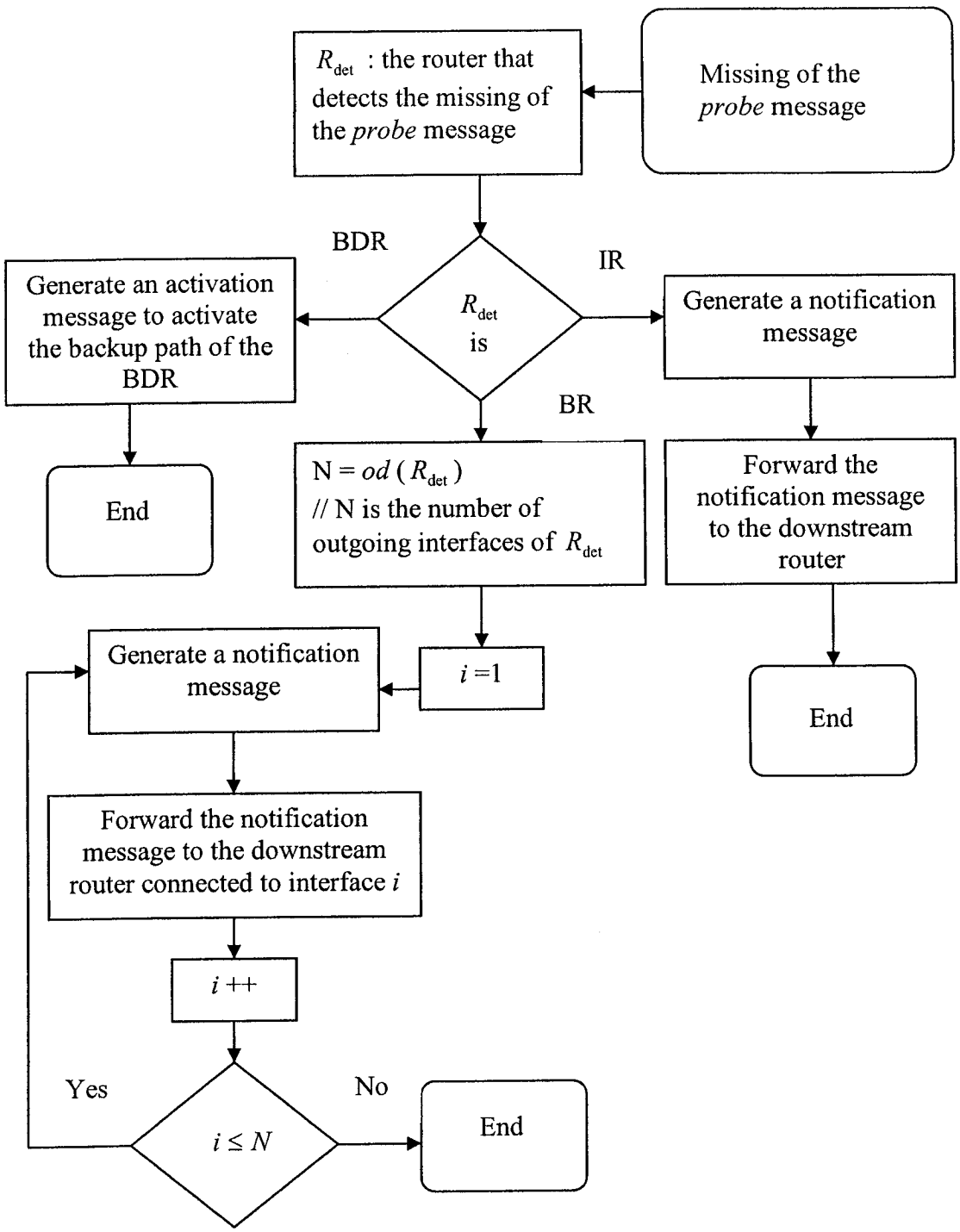


Figure 5.7: A flowchart for the detection and notification mechanism.

R1→ R3→ BDR2} and P3 = { d_3^{root} → R1→ R4→BDR3}. For each working path in the domain, a backup path is set up such that this path and its working path are link- and node-disjoint. To carry out this objective, a signaling protocol that uses constraint-based routing such as CR-LDP [20] or RSVP-TE [21] should be used. The constraint in this case is that a backup path must avoid any router over the corresponding working path.

Once the domain topology is known at all routers in that domain, d_3^{root} starts reserving a backup path with each BDR. This process involves exchanging a set of messages to assign MPLS labels over the backup paths in both directions. One direction is labeled to switch the traffic over that direction when a failure takes place. The other direction is labeled in the reverse direction of the first one. The second direction is used to forward the activation message from the BDR to the domain's root. Assuming the infinite capacity of all the links in the network, the shortest path between d_3^{root} and each BDR will be selected as a backup path that is link- and node-disjoint with the corresponding working path.

Assume the delay of each link is equal to one time unit. Moreover, assume the delay of the backup paths BP1, BP2 and BP3 in Figure 5.8 are 5, 4, 6; respectively. If the processing times for the notification and activation messages at the routers are ignored, the maximum time it takes to activate all the backup paths after the detection of a failure inside the domain depicted in Figure 5.8 is equal to 7 time units. This is the time it takes after the failure detection of R1, which is the CR in the domain. Table 5.1 illustrates the time sequence when the failure of R1 occurs.

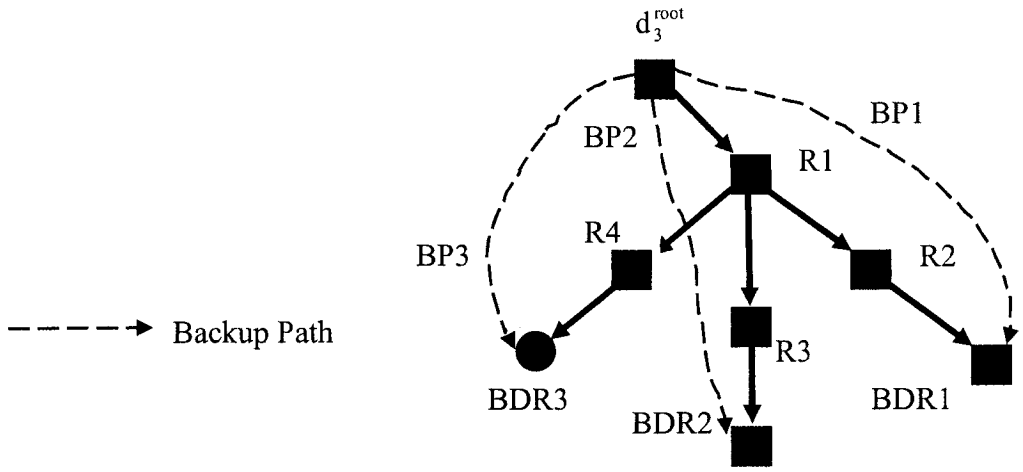


Figure 5.8 A domain with three BDRs.

Table 5.1 The time sequence when CR R1 fails.

Time Sequence	Events
0	R1 fails
T_{det}	R2, R3, and R4 detect the failure as a result of missing probe messages. R2, R3 and R4 then generate notification messages, which are forwarded to BDR1, BDR2 and BDR3.
$T_{det}+1$	Notification messages are received by the BDRs which then generate the activation messages to activate BP1, BP2 and BP3.
$T_{det}+5$	BP2 is active. d_3^{root} starts switching the traffic over that path.
$T_{det}+6$	BP1 is active. d_3^{root} starts switching the traffic over that path.
$T_{det}+7$	BP3 is active. d_3^{root} starts switching the traffic over that path.

5.3.4 Method 2: PSL at the High Level

In this method, after the domain topology is received by all the routers in that domain, each BDR runs the algorithm depicted as a flowchart in Figure 5.9. The idea in this method is to let each BDR select its PSL. The output of the algorithm in Figure 5.9 is a set of PSL candidates. After obtaining the set that maintains the PSL candidates, each BDR builds a backup path with one of those candidates. That PSL is indeed a BDR which is in a higher level in the domain relative to the level of the BDR that runs the algorithm.

The algorithm in Figure 5.9 is used to find all the PSL candidates for the BDR that runs the algorithm. The output of the algorithm is the S_{PSL} set that contains the PSL candidates. The input of the algorithm is the address of the BDR that runs the algorithm. The address of the BDR router is assigned to a local variable in the algorithm called *nodeA* with a level equal to zero. Then, the level of any router in the domain is calculated with reference to *nodeA* level. Any BDR that has a level greater than *nodeA* level is added to the S_{PSL} set. During the searching, the level is incremented when the algorithm goes up towards the root of the domain and the current router under test is a BR router. The level is decremented if the test router is a BR and the searching goes down toward the BDRs. The level is not changed if the router under test is an IR router. The algorithm stops searching when the S_{PSL} set is not empty. S_C is a set that maintains the routers that will be tested for PSL candidacy. The following functions are used in the algorithm but their codes are not completely shown:

1. ***Get_Parent(nodeA)***: returns the parent router of *nodeA*.

2. ***children_of (nodeC)***: returns a set that contains the children routers of *nodeC.*

3. ***get_node_from(S_C)***: returns a member from S_C.

Figure 5.10 shows domain 6 of the original tree given in Figure 5.2. The domain has six BDRs where all them are egress LERs. The PSL router for E2 is E1 while d_6^{root} is the PSL router for E1. Some BDRs may have more PSL candidates but only one of them is selected as a PSL. For instance, E4 has two PSL candidates, E3 and E6, which are also the PSL candidates for E5. However, only E3 is selected as the PSL router for E4 and E6 is selected as the PSL router for E5. The selection is based on the assumption that the backup paths between E3 and E4 and between E6 and E5 are shorter than the backup paths between E6 and E4 and between E3 and E5, respectively.

When a BDR selects its PSL, it starts the process of reserving a backup path with that PSL. As in method 1, a signaling protocol that provides constraint-based routing is used to set up the backup path between the BDR and its PSL. The constraint is the avoidance of the backup path to any router, except the BDR and its PSL, in the corresponding working path. For instance, in Figure 5.10, BP2 is a backup path that is set up between BDR E2 and its PSL E1. That path must avoid any router of R1, R2 and R5.

If we assume that the capacity of each link in the network is infinity, any BDR will select the closer PSL from the PSL candidates. As is shown from the discussion, the difference between the two methods is that in method 1, the domain's root triggers the process of building the backup paths while in method 2, each BDR alone runs the algorithm in Figure 5.9 to select its PSL router.

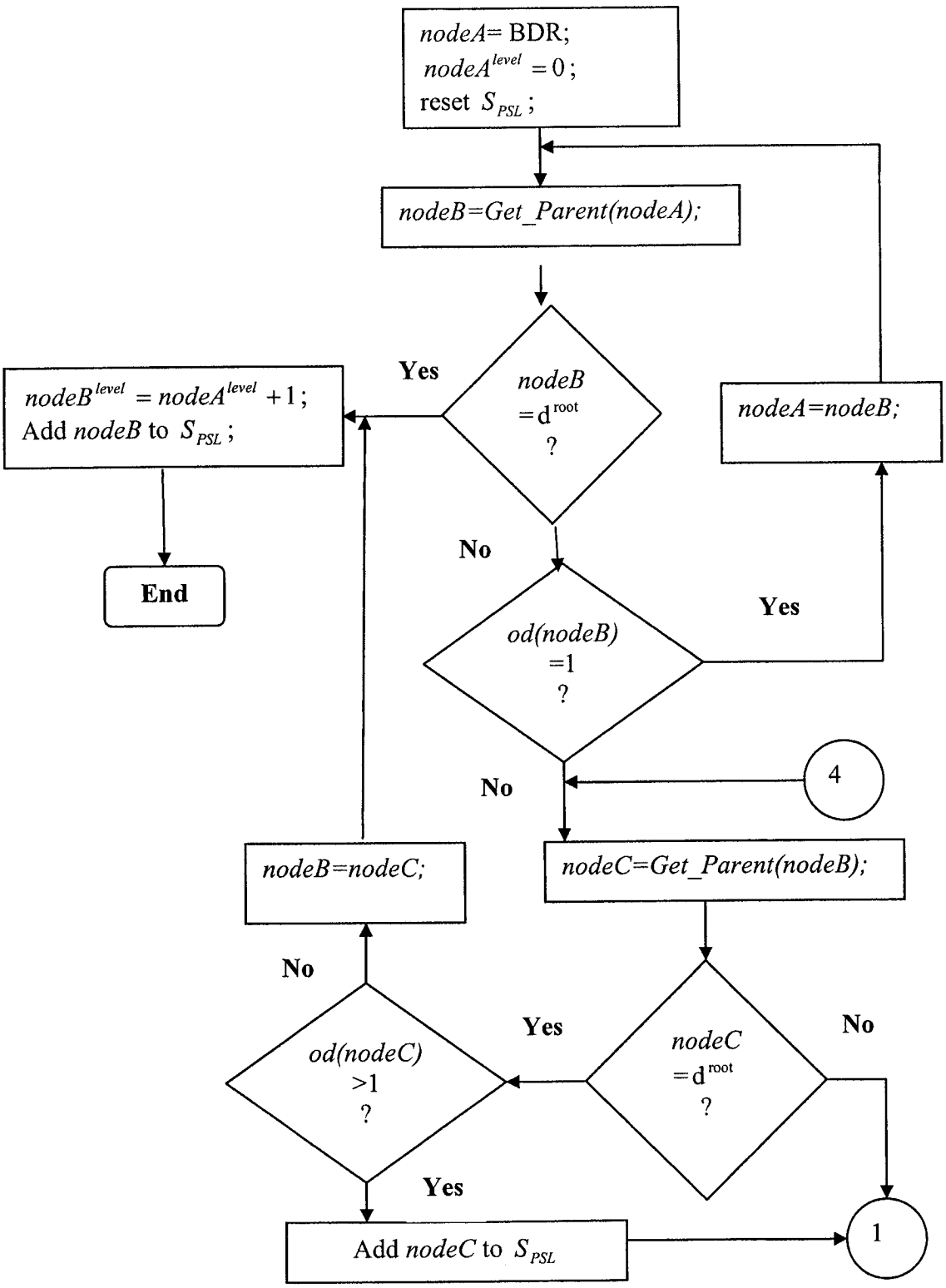


Figure 5.9 Flowchart of the algorithm that finds the PSL candidates for the current BDR

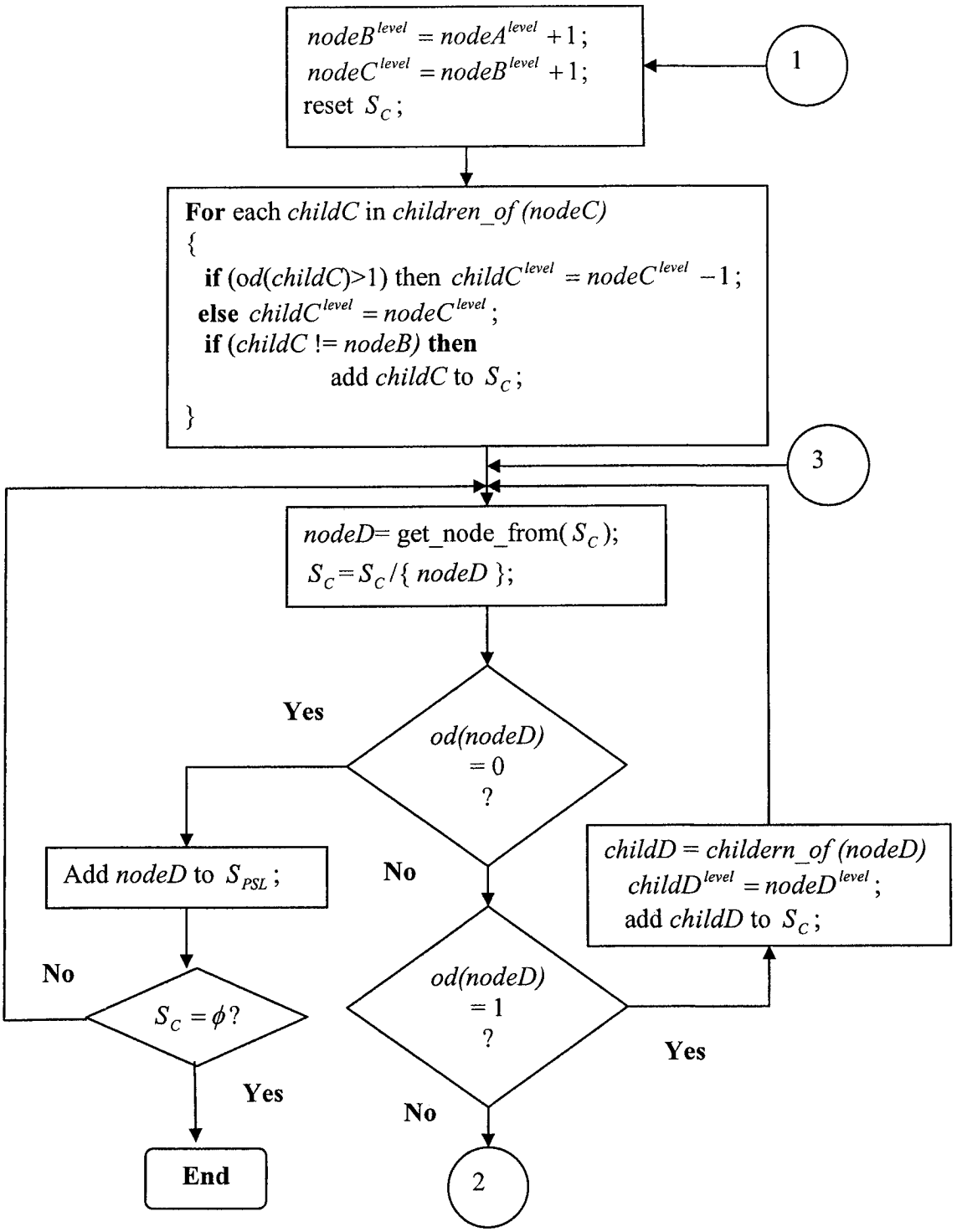


Figure 5.9 (continued).

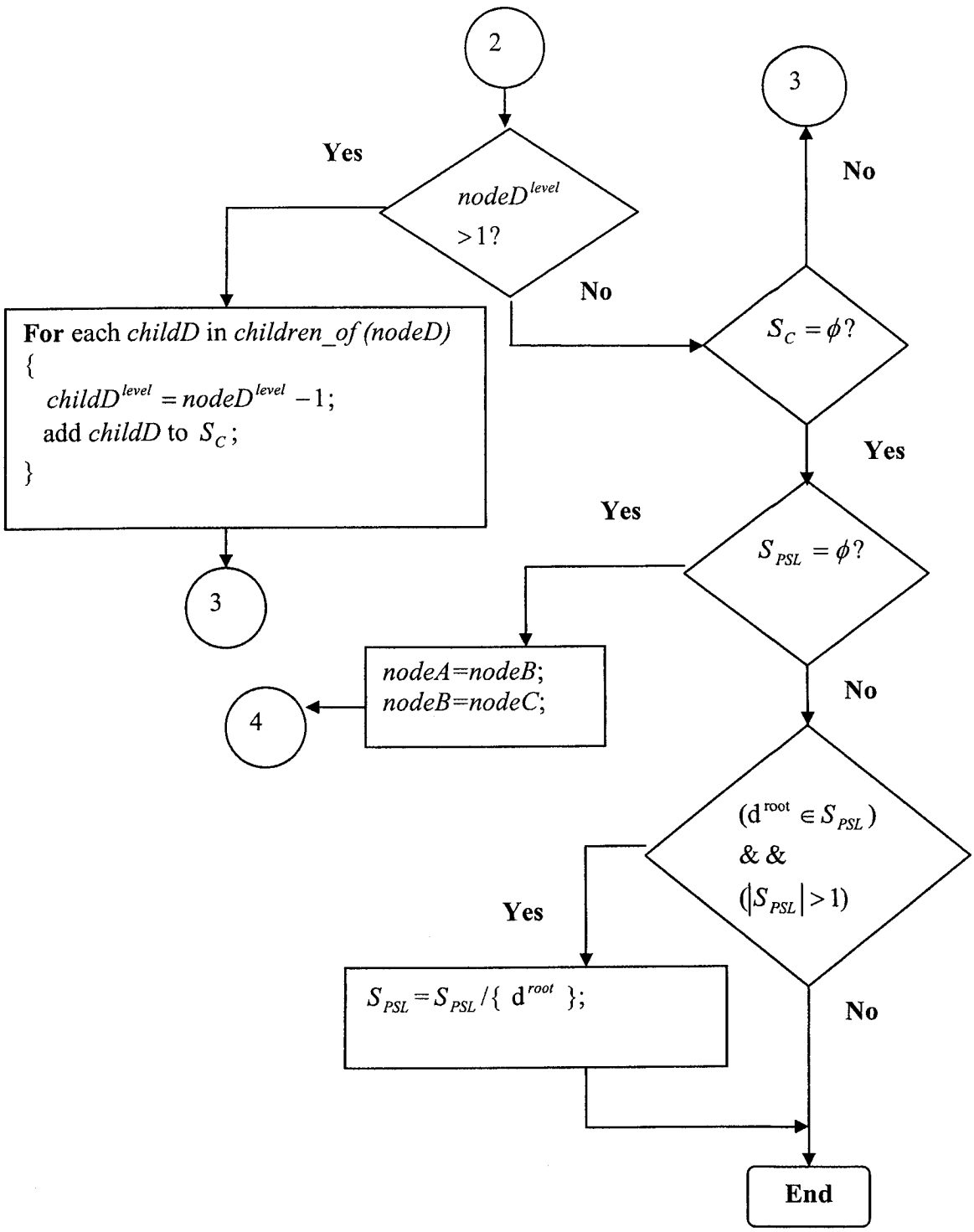


Figure 5.9 (continued).

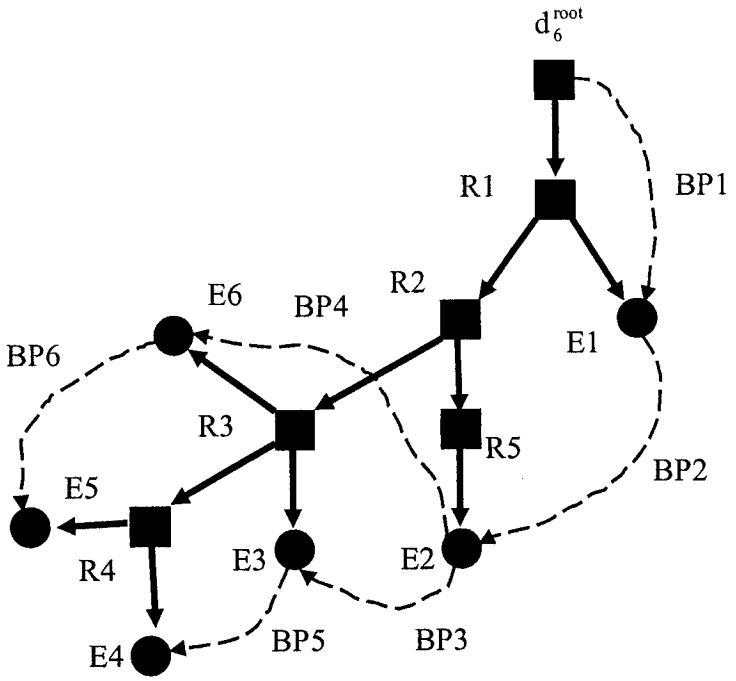


Figure 5.10 Domain 6 of the multicast tree in Figure 5.2 with the backup paths set up between the BDRs.

5.3.5 Number of Labels

The number of MPLS labels that are required when failure protection service is used will be three labels for each multicast tree in the network. The first label is used in order to switch the IP packets in the normal forwarding of the traffic when there is no failure. The other two labels are used when a failure takes place. One of the last two labels is used to switch the notification and activation messages. The third label is used after the recovery paths are activated.

5.4 Tree Maintenance

We discuss the effect of receiving a Join-Request or a Prune-Request message on the topologies of the domains. The discussion handles the necessary steps that should be triggered in order to keep the multicast tree divided as we have proposed in this chapter. When a new egress LER joins multicast session (S, G), a backup path must be set up to protect the new member. On the other hand, when a current egress LER leaves multicast session (S, G), the backup path that was set up with that egress LER must be released. This step is necessary to make the reserved capacity over the links of that path available for other flows.

The main changes that take place on the topologies of some domains as a result of receiving a Join-Request or a Prune-Request message are:

1. A domain is divided into two sub-domains.
2. A merging between two domains.
3. A shift in a domain.

These changes depend on three values:

1. The outgoing degree of R, $od(\mathrm{R})$.
2. The outgoing degree of $\mathrm{R_P}$, $od(\mathrm{R_P})$, where $\mathrm{R_P}$ is the parent router of R.
3. The outgoing degree of $\mathrm{R_C}$, $od(\mathrm{R_C})$, where $\mathrm{R_C}$ is a child router of R.

Before we go further in the detailed discussion, we point out the following:

1. The advantage of making each router maintain a copy of the domain where it belongs is clarified in the following discussion.

2. It should be noted that the BDRs and the roots of the domains maintain the topologies of two domains. Each domain's root maintains the topology of the domain where it acts as a root and the topology of the domain where its failure is handled. Each BDR maintains the topology of the domain where it acts as a BDR in that domain and the topology of the domain where its failure is handled.
3. As we have assumed in Chapter 3, the steps of the architecture proposed in the previous section are triggered every period of time. During the time between two consecutive periods, maintenance of the domains that will be affected should take place as discussed in this section.

5.4.1 Join-Request

When a new egress LER E_J sends a Join-Request message to join a multicast session (S, G), that message will be forwarded upstream until it is received by router R that has a multicast entry for (S, G), i.e., $od(R) \geq 1$. Any router between R and E_J will create a multicast entry for (S, G) and add that entry to the MRT (Multicast Routing Table). A set of steps must be carried out in order to set up a backup path for E_J based on the used method. In addition to that, a set of messages will be distributed to the routers in the domains where their topologies will change.

Let $P_{R \to E_J}$ denote the path segment from R to E_J . If $od(R) > 1$ and irrespective of the values of $od(R_P)$ and $od(R_C)$, the following steps are carried out to set up the backup path for E_J:

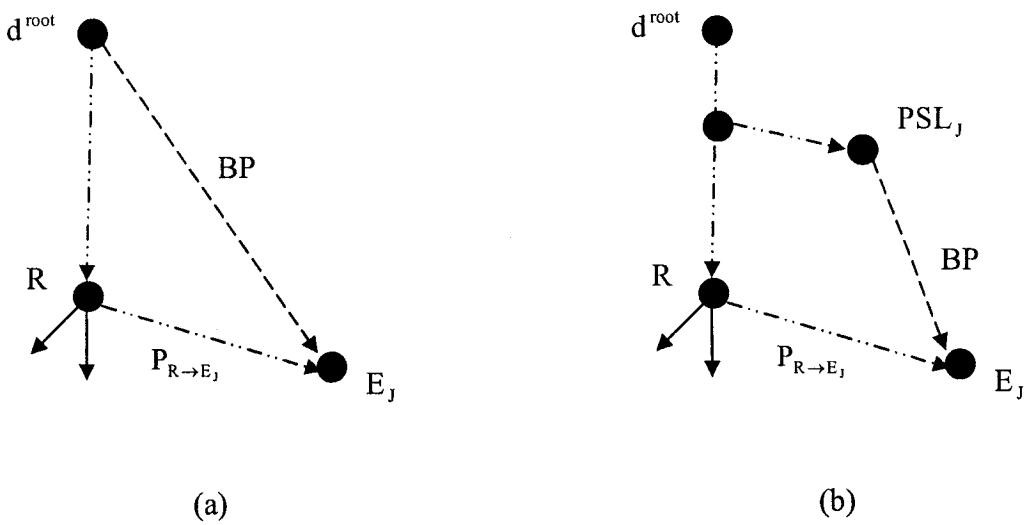


Figure 5.11 The situation when R receives a the Join- Request message $od(\mathrm{R}) > 1$.

1- R receives the path segment $P_{R \to E_J}$. This information is distributed to all the routers in the domain which it belongs to.

2- R forwards the domain's topology to every router over the path from R to E_J.

3- If method 1 is used, a backup path is set up between the domain's root and E_J as shown in Figure 5.11(a). If method 2 is used, E_J will run the algorithm given in Figure 5.9 to find the PSL candidate set. Then, one of the PSL candidates, PSL_J, is selected to set up the backup path with as shown in Figure 5.11(b). Furthermore, for each BDR that finds that E_J would be better than the current PSL, the backup path that was set up with that PSL is released and a new backup path is set up with E_J.

If $od(R) = 1$, one of the following four cases may take place depending on the values of $od(R_P)$ and $od(R_C)$:

Case 1: None of the main changes will take place.

This case occurs when $od(R_P) > 1$ and $od(R_C) = 1$. This case is handled as the case discussed before and shown in Figure 5.11. The only difference between this case and the previous one is that $od(R) = 1$.

Case 2: Domain Division.

This case occurs when $od(R_P) = 1$ and $od(R_C) = 1$. This case is depicted in Figure 5.12(a). R_{BR} is the first BR over the path from R_C to R_{BR}. R_{BR} could be the child router of R_C. Indeed, R_{BR} is the only BDR in domain d1 that will be affected from the Join-Request message. Before receiving the Join-Request message by R, the three routers R_P, R_C and R are all in the same domain d1. After E_J joins the multicast session (S, G) at R as shown in Figure 5.12(b), R becomes a BR and because R_P is an IR, domain d1 is divided into two sub-domains: d1-a and d1-b. R becomes a BDR in domain d1-a and R_P becomes the root for domain d1-b as shown in Figure 5.12(b). Note that domain d1-b will have only two BDRs: R_{BR} and E_J.

The following steps are carried out to divide domain d1 into two sub-domains:

1. R receives the path segment $P_{R \to E_J}$ and then distributes that information to all the routers in the domain which it belongs to.
2. R forwards the domain's topology to every router over the path from R to E_J.

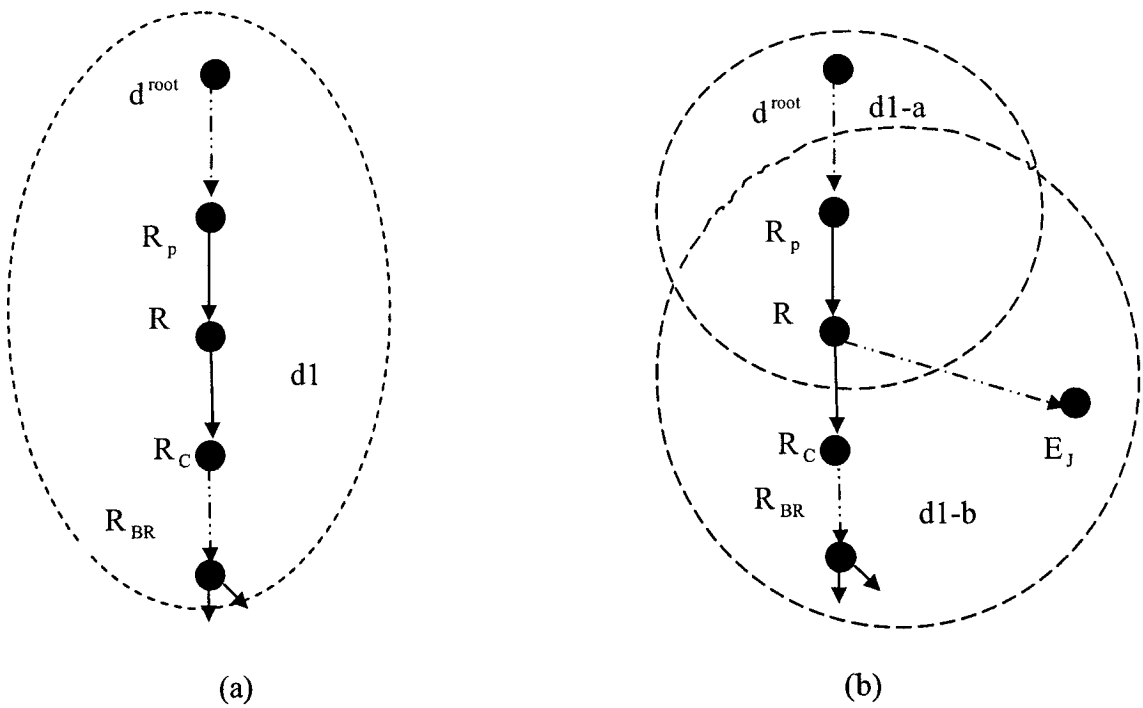


Figure 5.12 A domain division when R receives a Join-Request message.

3. If method 1 is used, d^{root} will release the backup path that was set up with R_{BR} and will set up a new path with R. On the other hand, R_P takes its role as a new domain's root for d1-b and sets up a backup path with R_{BR} and another backup path with E_J.

4. If method 2 is used, R_{BR} will release the backup path that was set up with its PSL. On the other side, R will set up a backup path with its PSL after running the algorithm given in Figure 5.9. Individually, R_{BR} and E_J will run the algorithm given in Figure 5.9 to find their PSL routers and then set up the backup paths with them. R_P will be selected as PSL router for both R_{BR} and E_J in domain d1-b

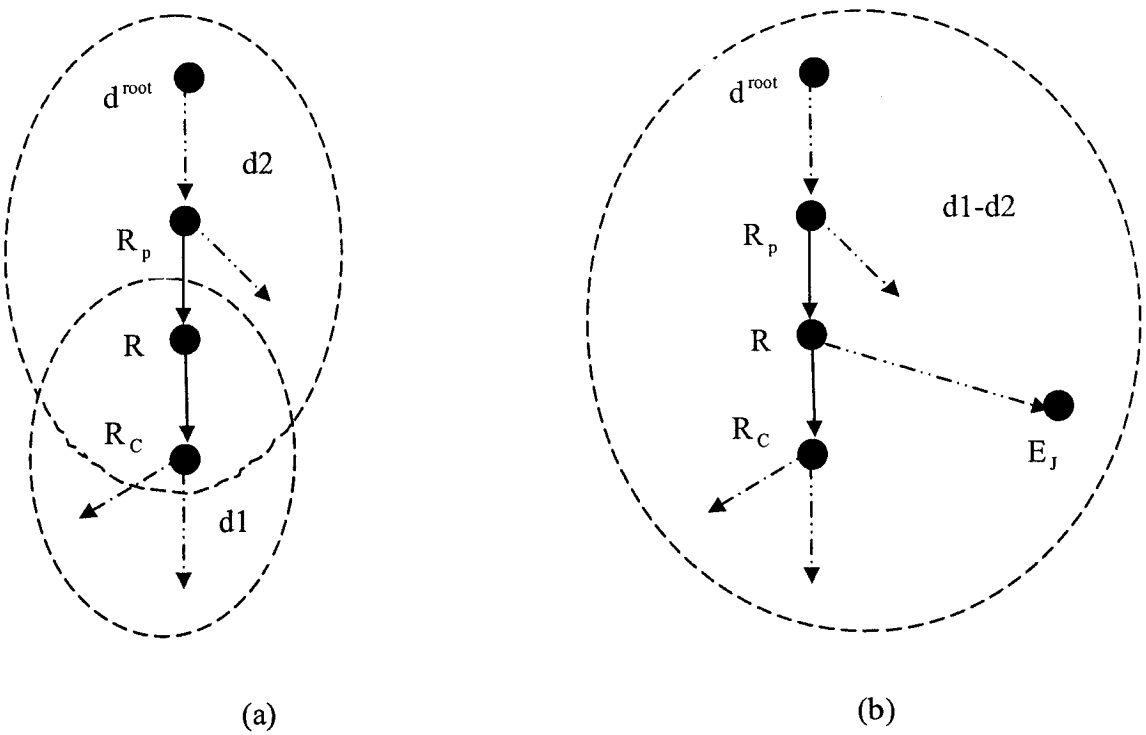


Figure 5.13 A merging of two domains when R receives a Join-Request message.

Case 3: Domain Merging.

This case takes place when $od(R_P) > 1$ and $od(R_C) > 1$. Figure 5.13(a) illustrates the situation before receiving the Join-Request message by R. In that Figure, R is a root for domain d1. R_P and R_C are in domain d2 where R_C is a BDR in that domain. After E_J joins the multicast session (S, G) at R, R becomes a BR. This results in the merging of the two domains d1 and d2 to form a large domain d1-d2 as shown in Figure 5.13(b). The root of the new domain is the root of d2, d^{root}. R will no longer be a domain's root.

The following steps are carried out to merge d1 and d2 into one domain:

1. R receives the path segment $P_{R \to E_J}$ and then builds the topology of the new domain. R then distributes the new topology to the routers in the two domains.

2. R forwards the topology of the new domain to every router over the path from R to E_J.
3. If method 1 is used, R will release the backup paths that were set up with every BDR in domain d1. On the other hand, d^{root} sets up backup paths with E_J and with the BDRs that were in d1.
4. If method 2 is used, each BDR in domain d1 will release the backup path that was set up with its PSL. Then, separately, those BDRs and E_J will run the algorithm given in Figure 5.9 to find their best PSL candidates and then set up the backup paths with them.

Case 4: Domain Shift.

This case occurs when $od(R_P)=1$ and $od(R_C)>1$. Figure 5.14(a) shows the situation before receiving the Join-Request message by R. In that Figure, R is a root for domain d1. R_P and R_C are in domain d2 where R_C is a BDR in that domain. After E_J joins the multicast session (S, G) at R, R becomes a BR. Because $od(R_P)=1$, R_P becomes the root of domain d1 instead of R. Moreover, R becomes a BDR in domain d2 instead of R_C which means that the two domains are shifted up as shown in Figure 5.14(b).

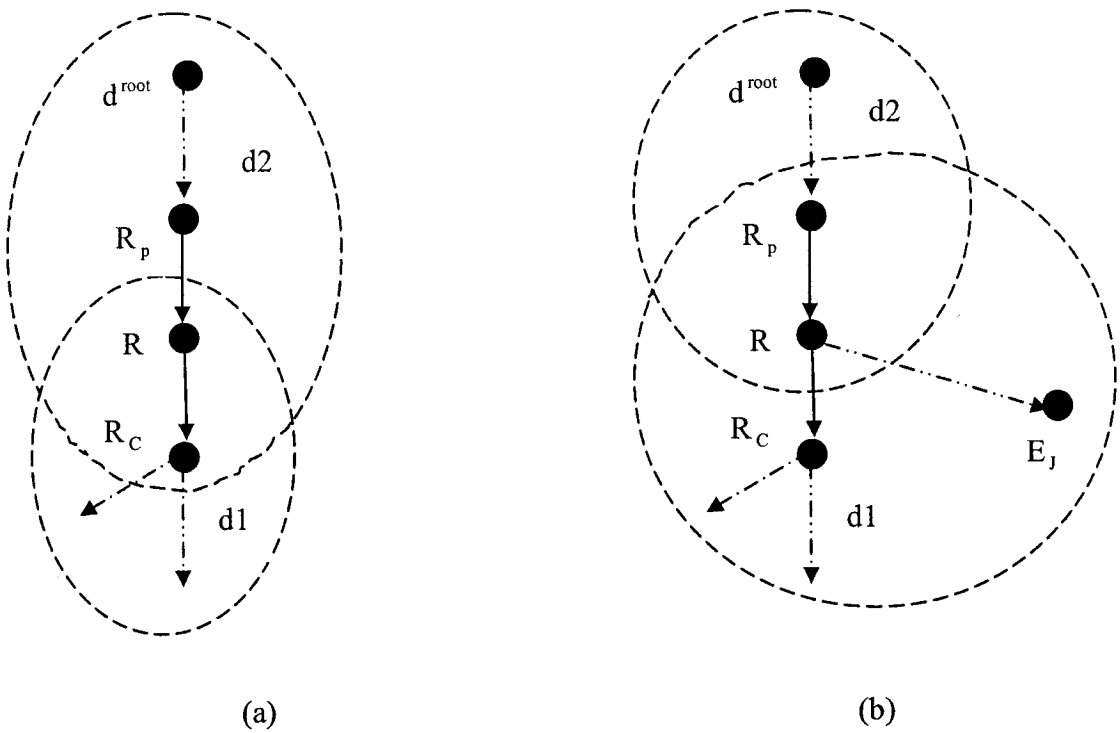


Figure 5.14 A shift up in the borders of d1 and d2 when R receives a Join-Request message.

The following steps are carried out in order to shift the borders of d1 and d2 up:

1. R receives the path segment $P_{R \to E_J}$ and then updates the topologies of the two domains d1 and d2. R then distributes the updated topologies to the routers in d1 and d2.
2. Because E_J belongs to domain d1, R forwards the topology of domain d1 to every router over the path from R to E_J.
3. If method 1 is used, R will release the backup paths that were set up with every BDR in domain d1 and d^{root} will release the backup path was set up with R_C and then set up a backup path with R. On the other side, R_P will set up backup up paths with the BDRs in domain d1 which includes E_J.

4. If method 2 is used, R_C will release the backup path was set up with its PSL. In domain d1, E_J will run the algorithm given in Figure 5.9. The result will be that R_P will the only candidate to set up that backup path with. The other BDRs in domain d1 will run the algorithm in Figure 5.9. If any BDR finds a better PSL candidate, a backup path will be set up with that candidate.

5.4.2 Prune-Request

When an on-tree egress LER E_P sends a Prune-Request message to leave a multicast session (S, G), that message will be forwarded upstream until it is received by router R that has an outgoing degree greater than 1. Any router between R and E_P which has an entry for (S, G) and its outgoing degree is equal to 1 will remove that entry from the MRT. The main changes that were mentioned in the beginning of Section 5.4 can take place on the topologies of some domains in the case of Prune-Request message and these changes depend also on the values of $od(R)$, $od(R_P)$ and $od(R_C)$. As in Join-Request, a set of messages will be distributed to the routers in the domains where their topologies will change. The purpose of these messages is also to keep the multicast tree divided as proposed by tree division algorithm. Let $P_{R \to E_P}$ denote to the path segment from R to E_P.

If $od(R) > 2$ and irrespective of the values of $od(R_P)$ and $od(R_C)$, the following steps are carried out to release the backup path for E_P:

1. R removes the path segment $P_{R \to E_P}$ from the domain's topology and then distributes that information to all the routers in the domain which it belong to.

2. If method 1 is used, the domain's root releases the backup path that was set up with E_P.
3. If method 2 is used, E_P releases the backup path that was set up with its PSL. Moreover, each BDR that considers E_P as its PSL router will release the backup path that was set up with E_P and then run the algorithm given in Figure 5.9 to find a new PSL candidate to set up a backup path with.

If $od(R) = 2$, one of the following four cases can take place depending on the values of $od(R_P)$ and $od(R_C)$:

Case 1: None of the main changes will take place.

This case occurs when $od(R_P) > 1$ and $od(R_C) = 1$. This case is handled as the case discussed before. The only difference between this case and the previous one is that $od(R) = 2$.

Case 2: Domain Division.

This case occurs when $od(R_P) > 1$ and $od(R_C) > 1$. This case is depicted in Figure 5.15(a). Before receiving the Prune-Request message by R, the three routers R_P, R_C and R are all in the same domain d1. After E_P leaves the session (S, G), the path segment $P_{R \to E_P}$ is removed from the topology of domain d1. This results in dividing d1 into two sub-domains: d1-a and d1-b. R becomes a domain's root for d1-b and R_C becomes a BDR in d1-a as shown in Figure 5.15(b).

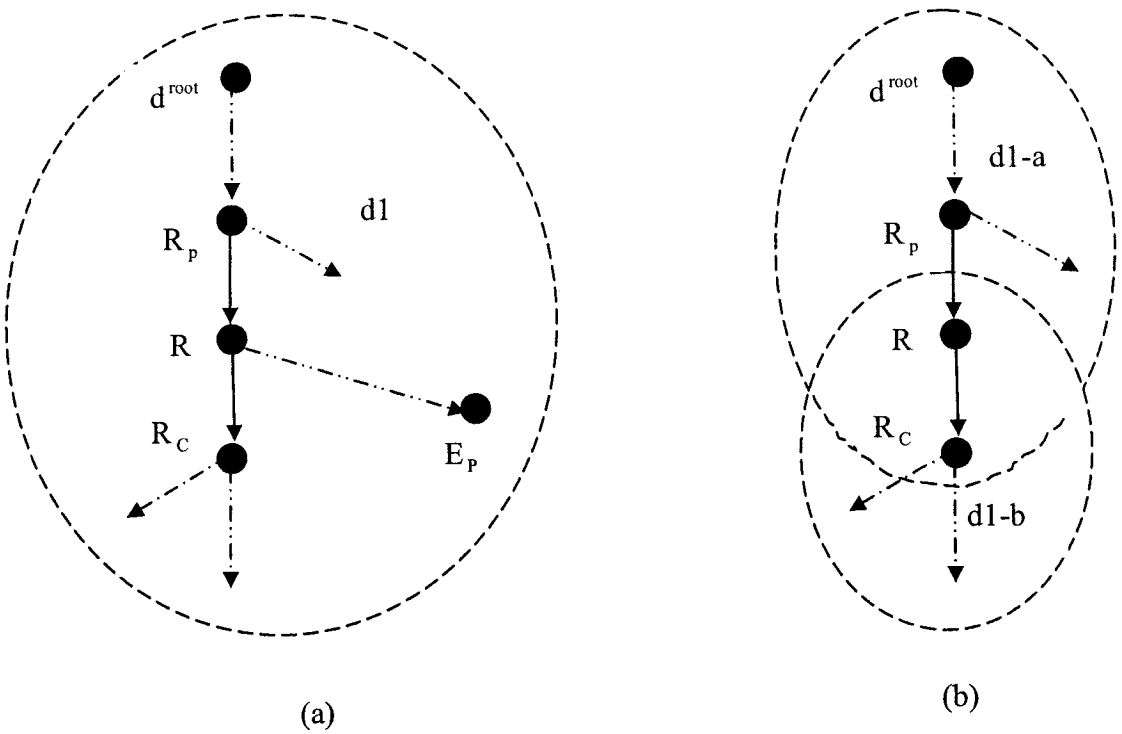


Figure 5.15 A domain division when R receives a Prune-Request message.

The following steps are carried out to divide domain d1 into two sub-domains:

1. R removes the path segment $P_{R \to E_P}$ from the topology of domain d1 and then distributes that information to each router in the domain.
2. If method 1 is used, d^{root} releases the backup paths that were set up with the BDRs in d1-b and sets up a backup path with R_C. As a root for d1-b, R starts setting up backup paths with the BDRs in domain d1-b.
3. If method 2 is used, each BDR in domain d1-b releases the backup that was set up with its PSL and runs the algorithm in Figure 5.9 to find a new PSL router. Moreover, R_C in d1-a runs the algorithm in Figure 5.9 to find its PSL.

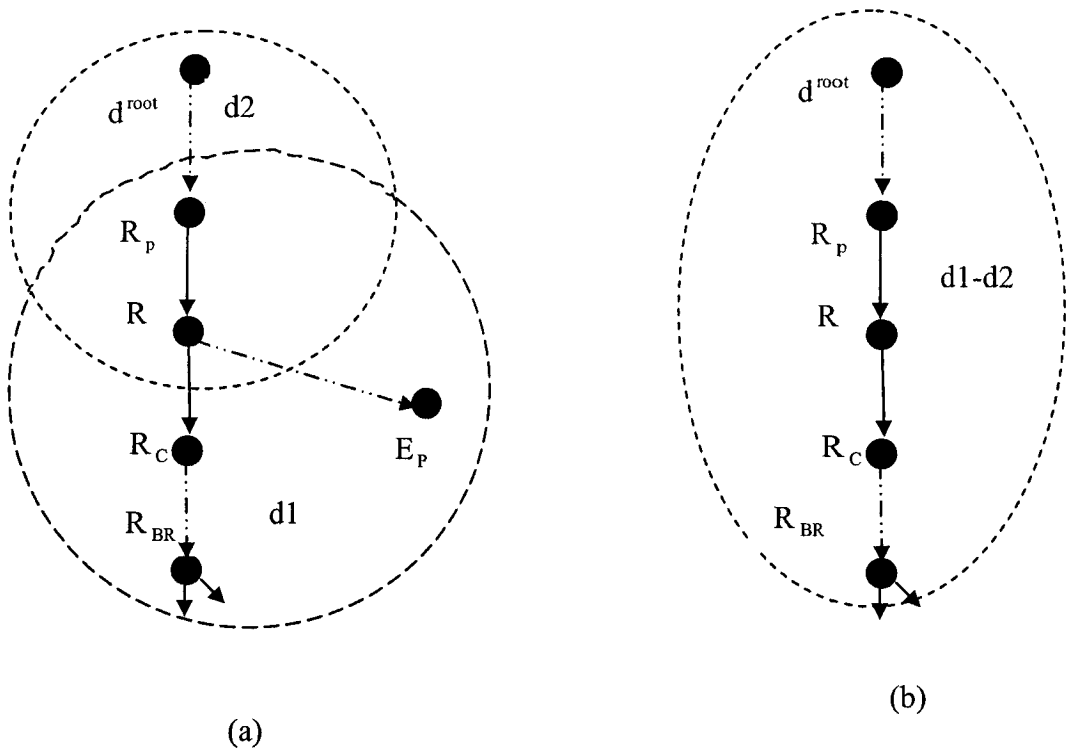


Figure 5.16 A merging of two domains when R receives a Prune-Request message.

Case 3: Domain Merging.

This case takes place when $od(R_P)=1$ and $od(R_C)=1$. Figure 5.16(a) shows the situation before receiving the Prune-Request message by R. In that Figure, R_C and R_P are in domain d1, where R_P is a root for d1. R is a BDR in d2. R_{BR} is the first BR over the path from R_C to R_{BR}. R_{BR} could be the child router of R_C. E_P and R_{BR} are the only BDRs in d1. After removing $P_{R\rightarrow E_P}$ from the topology of domain d1, the two domains d1 and d2 are merged together to form a large domain d1-d2 as illustrated in Figure 5.16(b). The root of the new domain is the root of d2, d^{root}. R will no longer be a domain's root.

The following steps are carried out to merge d1 and d2 in one domain:

1. R removes the path segment $P_{R \rightarrow E_P}$ from the topology of domain d1 and then distributes that information to each router in the two domains.
2. If method 1 is used, d^{root} releases the backup path that was set up with R. R_P releases the backup paths that were set up with E_P and R_{BR}. A new backup path will be set up between d^{root} and R_{BR}.
3. If method 2 is used, R releases the backup that was set up with its PSL. On the other hand, each of R_{BR} and E_P releases the backup path that was set up with its PSL. R_{BR} runs the algorithm in Figure 5.9 to find a new PSL in order to set up a backup path with it.

Case 4: Domain Shift.

This case takes place when $od(R_P) = 1$ and $od(R_C) > 1$. Figure 5.17(a) shows the situation before receiving the Prune-Request message by R. In that Figure, R_P is a root for domain d1, which also includes R_C and E_P. R is a BDR in domain d2. After removing $P_{R \rightarrow E_P}$ from the topology of domain d1, a shift down in the two domains takes place. This results in making R a root for domain d1 and R_C a BDR in domain d2 as illustrated in Figure 5.17(b).

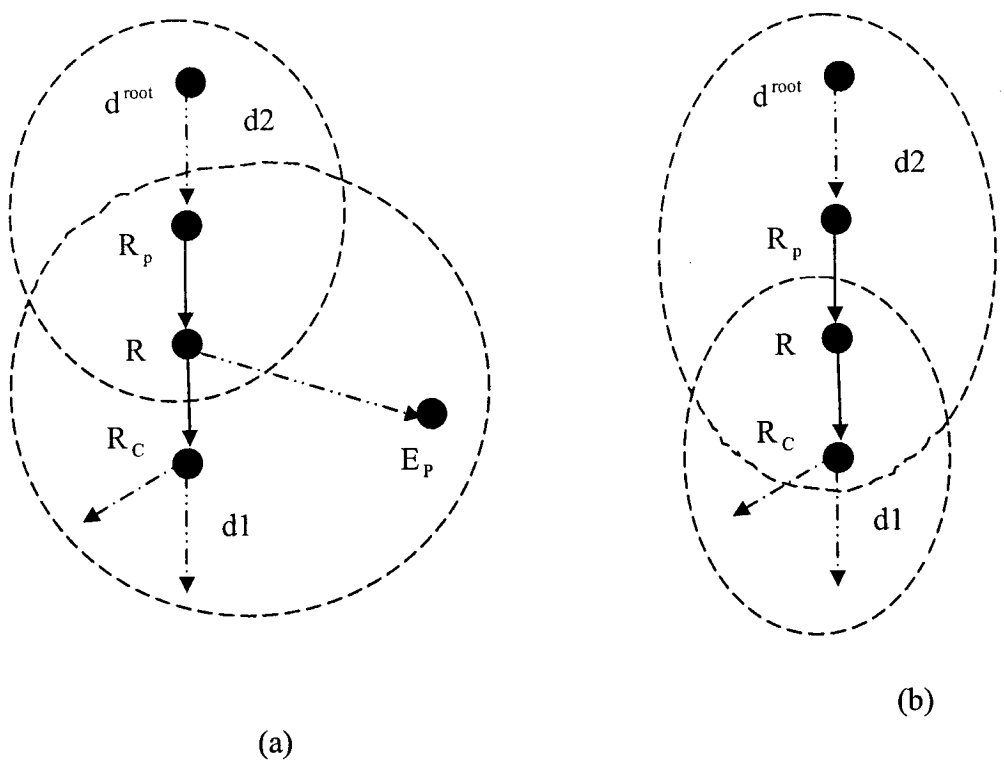


Figure 5.17 A shift down in the borders of d1 and d2 when R receives a Prune-Request message.

The following steps are carried out in order to shift the borders of d1 and d2 down:

1. R removes the path segment $P_{R \to E_P}$ from the topology of domain d1 and then distributes that information to all the routers in the two domains.
2. If method 1 is used, d^{root} releases the backup path that was set up with R and sets up a new path with R_C. Because R_P is no longer a root for domain d1, it releases the backup paths that were set up with E_P and the other DBRs in d1. On the other side, R starts setting up backup paths with the BDRs in domain d1.

3. If method 2 is used, R releases the backup path that was set up with its PSL. E_P also releases the backup path that was set up with its PSL. Moreover, each BDR in domain d1 that considers E_P as its PSL router will release the backup that was set up with E_P and then run the algorithm in Figure 5.9 to find a new PSL router. In domain d1, R_C runs the algorithm in Figure 5.9 to find its PSL.

5.4.3 Probability of Change in the Domain Topology

We have seen from the previous discussion that any new Join-Request message or Prune-Request message will influence the topology of certain domains. Depending on the values of $od(R)$, $od(R_P)$ and $od(R_C)$, we have eight possible cases. By adding the message type received by R whether it is a Join-Request message or Prune-Request message, the number of cases will be sixteen. Only six of them have their influence on the topologies of two domains. Those cases produce what we called the main changes. Table 5.2 shows the conditions of the main changes in both Join-Request and Prune-Request. The other cases change only the topology of one domain. As discussed above, a set of messages are generated to update the topologies of the affected domains. However, recall that we have assumed at the beginning of this thesis that the MPLS network is a backbone one that either connected to direct local networks or connected to access networks. This means that it has low probability to receive Join-Request and Prune-Request messages frequently.

Table 5.2 The conditions of the main changes in both Join-Request and Prune-Request.

Event	Domain Division	Domain Merging	Domain Shift
Conditions in Join-Request	$od(\mathrm{R}) = 1$ and $od(\mathrm{R_P}) = 1$ and $od(\mathrm{R_C}) = 1$.	$od(\mathrm{R}) = 1$ and $od(\mathrm{R_P}) > 1$ and $od(\mathrm{R_C}) > 1$.	$od(\mathrm{R}) = 1$ and $od(\mathrm{R_P}) = 1$ and $od(\mathrm{R_C}) > 1$.
Conditions in Prune-Request	$od(\mathrm{R}) = 2$ and $od(\mathrm{R_P}) > 1$ and $od(\mathrm{R_C}) > 1$.	$od(\mathrm{R}) = 2$ and $od(\mathrm{R_P}) = 1$ and $od(\mathrm{R_C}) = 1$.	$od(\mathrm{R}) = 2$ and $od(\mathrm{R_P}) = 1$ and $od(\mathrm{R_C}) > 1$.

To see the effect of Join-Request and Prune-Request messages on the domains topologies, we use the probability-based analysis. Let N_{IR} be the number of IRs in tree T and N_{BR} be the number of BRs in that tree. Let $N_{BR}^{od=2}$ be the number of BRs which have an outgoing degree equal to 2. The total number of nodes in the tree excluding the egress LERs and the ingress LER is $N_{IR} + N_{BR}$. Define the following notations:

$P_{division}$: The probability that a domain is divided.

$P_{merging}$: The probability that two domains are merged.

P_{shift} : The probability that a shift change takes place.

P_{simple} : The probability that none of the main changes occurs.

P_{Join} : The probability that the message received by R is a Join-Request message.

P_{Prune} : The probability that the message received by R is a Prune-Request message. Note that $P_{Prune} + P_{Join} = 1$.

Let $a = N_{IR}$, $b = N_{BR}$, $c = N_{IR} + N_{BR}$ and $d = N_{BR}^{od=2}$. Before we go further with the probability-based analysis and the derivations of $P_{division}$, $P_{merging}$, P_{shift} and P_{simple} we clarify the following points:

1. The probability that R is an IR is $(\frac{a}{c})$. This is equivalent to the probability that $od(\mathrm{R}) = 1$.
2. The probability that $\mathrm{R_C}$ or $\mathrm{R_P}$ is a BR is $(\frac{b}{c})$. This is equivalent to the probability that $od(\mathrm{R_P}) > 1$ or $od(\mathrm{R_C}) > 1$.
3. The probability of the compound condition $od(\mathrm{R}) = 1$ and $od(\mathrm{R_P}) = 1$ and $od(\mathrm{R_C}) = 1$ is equal to $\left(\frac{a}{c}\right).\left(\frac{a-1}{c-1}\right).\left(\frac{a-2}{c-2}\right)$.
4. The probability that $od(\mathrm{R}) = 2$ is equal to $(\frac{d}{b}).(\frac{b}{c}) = \frac{d}{c}$

The value of $P_{division}$ is calculated as:

$$P_{division} = P_{Join}.C_{Join}^{division} + P_{\mathrm{Prune}}.C_{\mathrm{Prune}}^{division} \tag{5.5}$$

where $C_{Join}^{division}$ is the probability that the conditions of domain division are satisfied given that the received message is a Join-Request message and $C_{\mathrm{Prune}}^{division}$ is the probability that the conditions of domain division are satisfied given that the received message is a Prune-Request message. From the conditions given in Table 5.2, $C_{Join}^{division}$ is given as:

$$C_{Join}^{division} = \left(\frac{a}{c}\right).\left(\frac{a-1}{c-1}\right).\left(\frac{a-2}{c-2}\right) \tag{5.6}$$

and $C_{Prune}^{division}$ is given as:

$$C_{Prune}^{division} = \left(\frac{d}{c}\right).\left(\frac{b-1}{c-1}\right).\left(\frac{b-2}{c-2}\right) \tag{5.7}$$

assuming $P_{Prune} = P_{Join}$ and by substituting Equations (5.6) and (5.7) in Equation (5.5), we get:

$$P_{division} = 0.5.\frac{a.(a-1).(a-2) + d.(b-1).(b-2)}{c.(c-1).(c-2)} \tag{5.8}$$

The value of $P_{merging}$ is calculated as:

$$P_{merging} = P_{Join}.C_{Join}^{merging} + P_{Prune}.C_{Prune}^{merging} \tag{5.9}$$

where $C_{Join}^{merging}$ is the probability that the conditions of domains merging are satisfied given that the received message is a Join-Request message and $C_{Prune}^{merging}$ is the probability that the conditions of domains merging are satisfied given that the received message is a Prune-Request message. From Table 5.2, $C_{Join}^{merging}$ and $C_{Prune}^{merging}$ are calculated as:

$$C_{Join}^{merging} = \left(\frac{a}{c}\right).\left(\frac{b}{c-1}\right).\left(\frac{b-1}{c-2}\right) \tag{5.10}$$

$$C_{Prune}^{merging} = \left(\frac{d}{c}\right).\left(\frac{a}{c-1}\right).\left(\frac{a-1}{c-2}\right) \tag{5.11}$$

again assuming $P_{Prune} = P_{Join}$ and by substituting Equations (5.10) and (5.11) in Equation (5.9), we get:

$$P_{merging} = 0.5.\frac{a.[b.(b-1)+d.(a-1)]}{c.(c-1).(c-2)} \tag{5.12}$$

The value of P_{shift} is calculated as:

$$P_{shift} = P_{Join}.C_{Join}^{shift} + P_{Prune}.C_{Prune}^{shift} \tag{5.13}$$

where C_{Join}^{shift} is the probability that the conditions of domain shifts are satisfied given that the received message is a Join-Request message and C_{Prune}^{shift} is the probability that the conditions of domain shifts are satisfied given that the received message is a Prune-Request message. From Table 5.2, C_{Join}^{shift} and C_{Prune}^{shift} are calculated as:

$$C_{Join}^{shift} = \left(\frac{a}{c}\right).\left(\frac{a-1}{c-1}\right).\left(\frac{b}{c-2}\right) \tag{5.14}$$

$$C_{Prune}^{shift} = \left(\frac{d}{c}\right).\left(\frac{a}{c-1}\right).\left(\frac{b-1}{c-2}\right) \tag{5.15}$$

again assuming $P_{Prune} = P_{Join}$ and by substituting Equations (5.14) and (5.15) in Equation (5.13), we get:

$$P_{shift} = 0.5.\frac{a.[b.(a-1)+d.(b-1)]}{c.(c-1).(c-2)} \tag{5.16}$$

Let $\lambda = \frac{d}{b}$. Then $d = \lambda.b = \lambda.(c-a)$. By substituting the values of d and $b = c-a$ in Equations (5.8), (5.12) and (5.16), we get:

$$P_{division} = 0.5.\frac{a.(a-1).(a-2) + \lambda.(c-a).(c-a-1).(c-a-2)}{c.(c-1).(c-2)} \tag{5.17}$$

$$P_{merging} = 0.5.\frac{a.[(c-a).(c-a-1) + \lambda.(c-a).(a-1)]}{c.(c-1).(c-2)} \tag{5.18}$$

$$P_{shift} = 0.5.\frac{a.[(c-a).(a-1) + \lambda.(c-a).(c-a-1)]}{c.(c-1).(c-2)} \tag{5.19}$$

and $$P_{simple} = 1 - (P_{division} + P_{merging} + P_{shift}) \tag{5.20}$$

Figure 5.18 through Figure 5.21 illustrate, respectively, the values of $P_{division}$, $P_{merging}$, P_{shift} and P_{simple} versus the number of IRs when $N_{BR} = 100$. As shown in Figure 5.21, P_{simple} records the maximum value, and it is always greater than or equal to 0.5. A comparison among the three main changes shows that a domain division has the maximum probability as shown in Figure 5.18. The values of $P_{merging}$ and P_{shift} are always less than 0.14 when $N_{BR} = 100$ as shown in Figures 5.19 and 5.20.

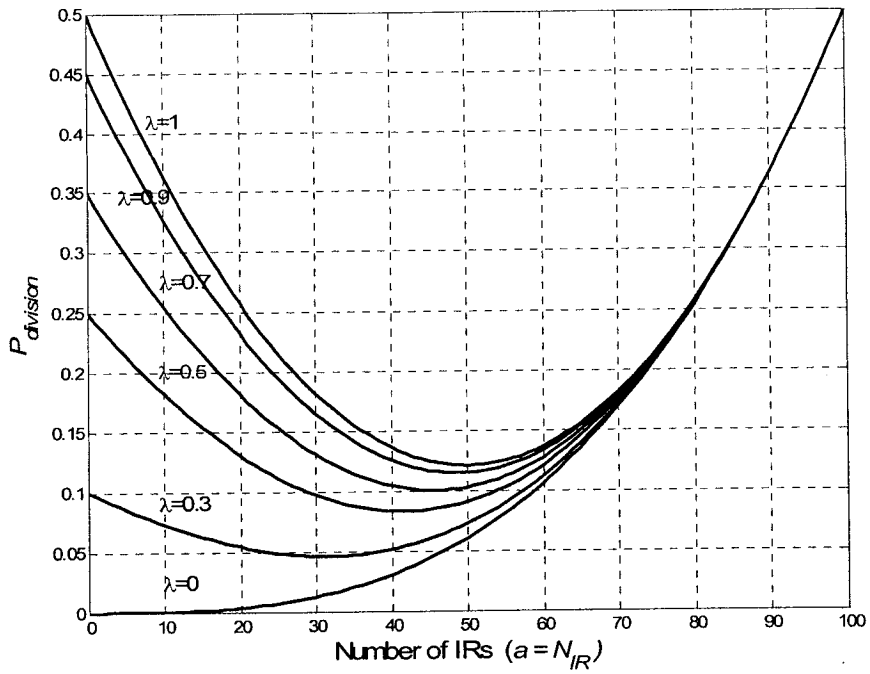


Figure 5.18: $P_{division}$ versus the number of IRs when $N_{BR} = 100$.

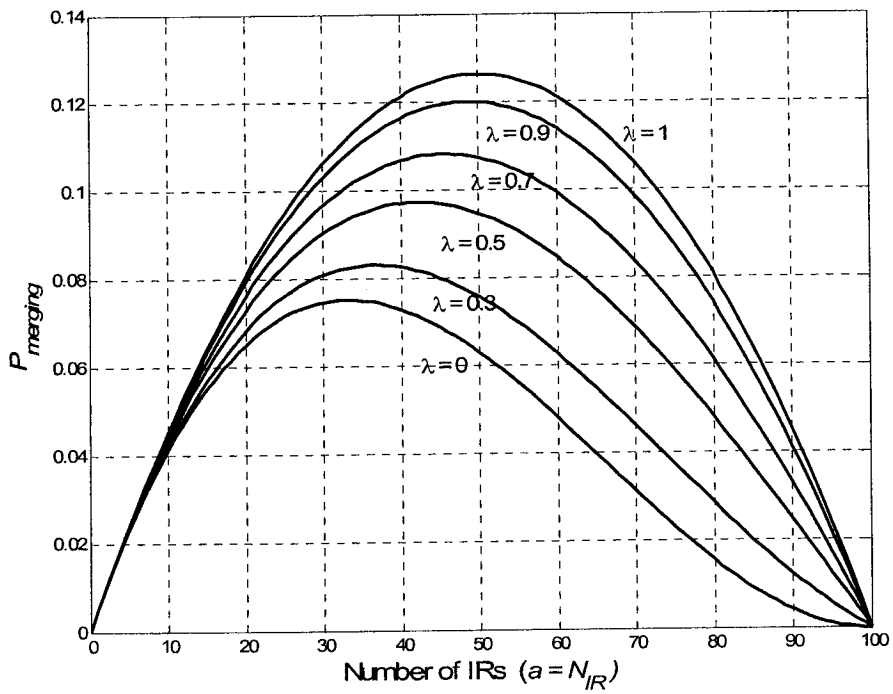


Figure 5.19: $P_{merging}$ versus the number of IRs when $N_{BR} = 100$.

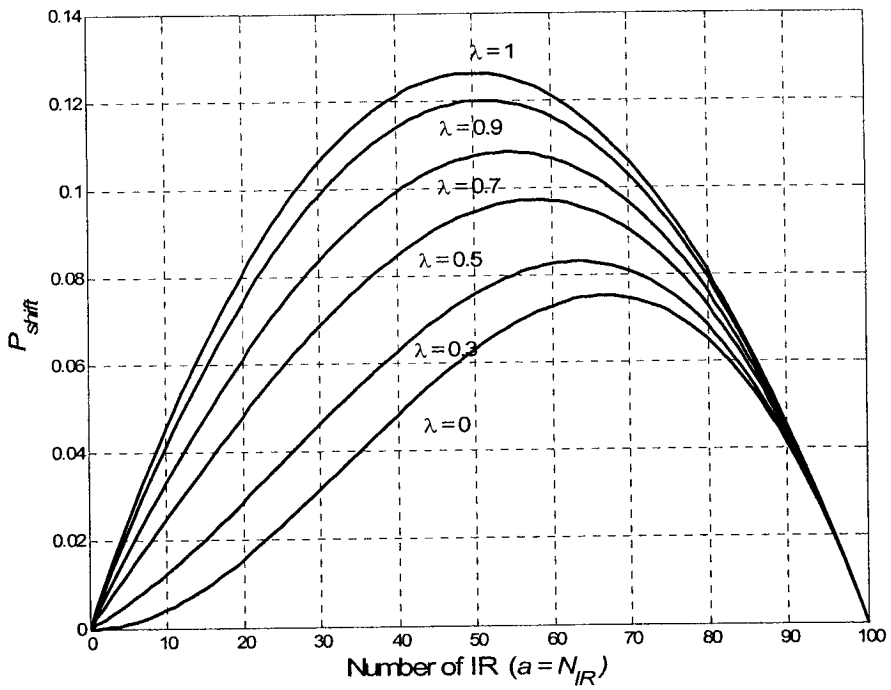


Figure 5.20: P_{shift} versus the number of IRs when $N_{BR} = 100$.

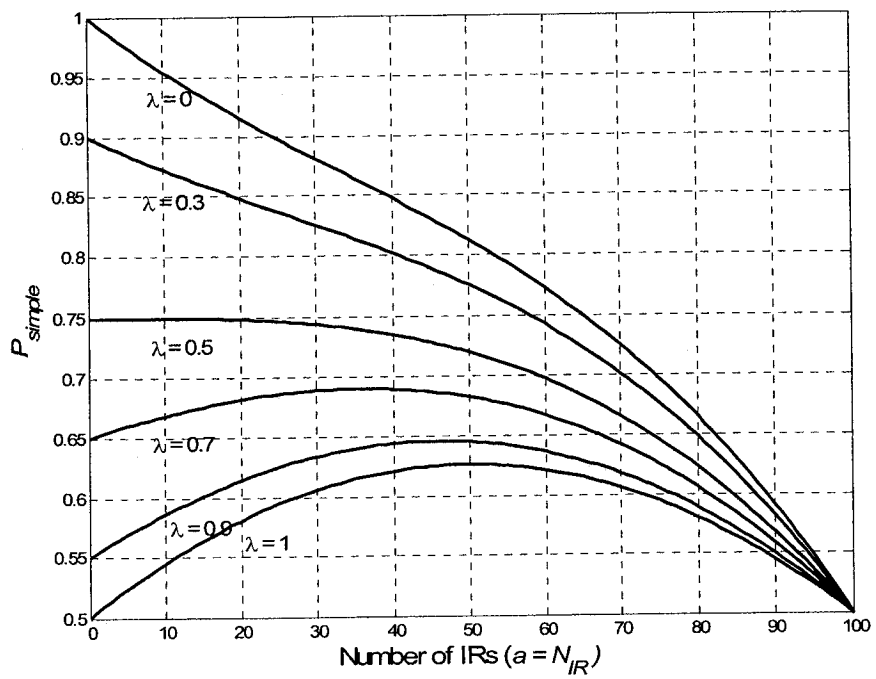


Figure 5.21: P_{simple} versus the number of IRs when $N_{BR} = 100$.

5.5 Justification of the Assumptions

At the beginning of this chapter, we have made the following two assumptions:

1. Only a single failure can take place at any time in the network [44]. In other words, a single failure model is assumed.
2. We assume that no failure will take place until the backup paths are set up in all the domains.

Assumption 1 implies a single failure model. However, a multiple failure model can be assumed in the network. In this case, multiple backup paths should be reserved for each working path. For instance, if a maximum of two simultaneous failures is assumed in the network, then two backup paths for each primary path should be reserved in order to guarantee uninterrupted connectivity in the worst case when two out of the three paths fail. The disadvantage of considering a two-failure model is that there is wastage of the capacity over the second backup path especially when the two failures occur most of the time over the same path.

If a failure occurs during the time when the topology is collected or when the tree division algorithm is run at the ingress LER, the failure recovery architecture will probably not function. Assumption 2 is necessary in this case to guarantee that the tree topology will be divided correctly into sub-domains at the ingress LER. Assumption 2 can lead to a significant impact if the time it takes to collect the tree topology, run the tree division algorithm at the ingress, and then distribute the domain information to the corresponding routers is too long. The impact that may result from considering assumption 2 can be relieved if we let each router over a certain tree decide its type (IR, BR, or BDR). This allows the domain topology to be determined locally with no need to

send all the tree topology to the ingress LER. The pseudo code that divides the tree locally is shown in table 5.3. In that table, D_{top} is the topology of the domain where its boundaries are not yet determined. When router R is an IR and its child router (Rd) is a BR, then R can decided by itself that it is a root for the domain whose topology, D_{top}, just received from Rd. In this case, R declares that information to every router in D_{top} and forms a new domain topology where router Rd is a BDR in that domain. R then adds its IP address to the new D_{top} and sends this topology upstream to its parent router.

If R is a BR, then it should collect all the topologies from all its child routers and form a new D_{top} from these topologies. Then, it adds IP address to the new D_{top} and sends this topology upstream to its parent router.

Dividing the tree locally will speed up the process of determining the domains and their BDRs. In this case, even if a failure occurs during the tree division process, the impact of that will be reduced significantly because the effect of producing wrong division of the tree is limited to one domain. The other domains will not be affected.

Table 5.3: The pseudo code that divides the tree locally.

```
1   If R is an IR then                          // od(R) =1
2       begin
3       receive D_top from the downstream router Rd;
4       If Rd is a BR then
5           begin
6           R is a domain's root for the domain whose topology is D_top ;
7           R sends a message to each router in T_top to let it know about its domain;
8           R resets the D_top ;
9           R establishes a new D_top with Rd as a BDR in that domain;
10          end
11      R adds its IP to D_top ;
12      R sends D_top to the upstream router   // i.e. to its parent router
13      End
14  If R is a BR then                          // od(R) >1
        Begin
15      m = od(R)                   // m = number of the outgoing interfaces of R
16      for i = 1 to m do   receive D_top[i] from the downstream router Rd[i];
17      Form a new domain D_top that is composed of D_top[i] for 1 ≤ i ≤ m;
18      R adds its IP to D_top ;
19      R sends D_top to the upstream router   // i.e. to its parent router
20      End
```

5.6 Summary

We have proposed an MPLS-based architecture that protects a multicast tree from a single link/node failure. The architecture is composed of a set of steps. The first step in the architecture is the collection of the tree topology of the multicast tree. That topology should be sent to the ingress LER of the corresponding tree. After that step, a tree division algorithm is triggered at the ingress LER. The algorithm divides the multicast tree into several domains where each domain represents a stand-alone tree. The next step is the distribution of the topologies of the domains to the corresponding routers. The last step in the architecture is the setup of the backup paths inside the domains. Two methods are proposed for this purpose: the common PSL method and the PSL at high level method.

We have given a discussion of the necessary steps that should be taken in the case when a new member joins the multicast session or in the case when a current member leaves the multicast session. Furthermore, we have used the probability-based analysis to study the effect of Join-Request and Prune-Request messages on the domains topologies.

In the next chapter, we propose three metrics in order to study the performance of the proposed architecture. Then, we present the results that evaluate the performance of the proposed architecture. The results compare the common PSL method and the PSL at high level method with the single local recovery method, the global recovery method and the method that sets up the paths between the branching points in the multicast tree. The last method is called segmentation-based method [66, 67].

Chapter 6

Simulation Results and Discussion

In this chapter, we define three metrics in order to study the performance of the architecture that was proposed in the previous chapter. Then, we present the results obtained by applying the proposed algorithms on different network topologies. A discussion of the results is presented in the last section of this chapter.

6.1 Performance Analysis

We use the following metric parameters to study the performance of the proposed architecture:

1. The total capacity used to allocate the backup paths.
2. Maximum and average notification times.
3. Average number of notification and activation messages

These metrics are discussed in the following subsections.

6.1.1 The Total Capacity Used to Allocate the Backup Paths

If the bandwidth request for multicast session k is b_k, then a backup capacity of the amount b_k should be reserved on each link of the backup path if that link is not part of the multicast tree built by session k. Mathematically, the cost of allocating a backup capacity on link l is given as

$$\text{Cost}(l) = \begin{cases} \mu_l . b_k, \text{if } l \notin T \\ 0, \text{if } l \in T \end{cases} \tag{6.1}$$

where μ_l is the unit cost of link l. Moreover, if link $l \notin T$ but it is reserved on a backup path in another domain for the same multicast tree T, the backup capacity b_k is not reserved again on link l. This mode of backup capacity reservation is called intra-demand sharing [48]. Assume the multicast tree T = (NT, LT) is divided into ND domains. Then the total backup capacity required for reserving the backup paths for tree T is the sum of all the backup capacities reserved for each domain:

$$C_t = \sum_{i=1}^{ND} C_i \tag{6.2}$$

In the simulation results in this chapter, C_t is represented by the total number of links reserved in the backup paths.

Let C_{local} be the reserved capacity when the local recovery is used. Then we define the percentage of saving in the capacity as

$$\Delta C = 100 \times \frac{C_{local} - C_t}{C_{local}} \tag{6.3}$$

6.1.2 Maximum and Average Notification Time

The total repair time, T_{repair}, is defined as the time spent by a particular rerouting approach to restore a communication after a link/node has failed. In general, T_{repair} depends on the following three components:

- T_{detect}: The time needed to detect the failure. This time depends on the time interval of the probe message.
- T_{notify}: The time needed to notify the PSL router about the failure. This depends on the distance between the router that detects the failure and the PSL router that switches the traffic over the backup path. Furthermore, T_{notify} depends on the notification mechanism used in a particular recovery approach.
- T_{switch_ov}: The time needed to switchover the data to the backup path by the PSL. This time depends on the processing time at the PSL router.

T_{repair} is the sum of the three components:

$$T_{repair} = T_{detect} + T_{notify} + T_{switch\_ov} \tag{6.4}$$

T_{notify} has the major effect on the repair time. Assume NE is the number of elements in tree T where e could be a link or a node. Define $T_{BDRi}^{notify}(e)$ as the time it takes until BDRi receives the notification message after the detection of the failure of element e. Define T_{BDRi}^{active} as the time it takes until the backup path between BDRi and its PSL router is active. If the processing times for the notification and activation messages at the routers are ignored, then $T_{BDRi}^{notify}(e)$ and T_{BDRi}^{active} are given as:

$$T_{BDRi}^{notify}(e) = delay(BDR_i, R_{det}) \tag{6.5}$$

and

$$T_{BDRi}^{active} = delay(BDR_i, PSL_i) \tag{6.6}$$

where $delay(BDR_i, R_{det})$ is the time delay between BDRi and the router that detects the failure of element e and $delay(BDR_i, PSL_i)$ is the time delay between BDRi and its PSL router. Then, the time it takes to activate the backup path of BDRi after the failure of e is given as:

$$T_{BDRi}^{notify/active}(e) = T_{BDRi}^{notify}(e) + T_{BDRi}^{active} \tag{6.7}$$

Note that while $T_{BDRi}^{notify}(e)$ depends on the location of the failure, T_{BDRi}^{active} is constant. Consider a certain domain in the multicast tree T. Let S_f represents the set of BDRs that are disconnected when element e fails. We define $T^{notify/active}(e)$ as the time required until all the backup paths of the BDRs in S_f become active. Mathematically, $T^{notify/active}(e)$ is given as

$$T^{notify/active}(e) = \arg \max_{BDRi \in S_f} (T_{BDRi}^{notify/active}(e)) \tag{6.8}$$

The maximum notification time to notify about a failure in tree T then is calculated as:

$$T_T^{max} = \arg \max_i (T^{notify/actice}(e_i)) \quad \text{for } 1 \le i \le NE \tag{6.9}$$

The average notification time in tree T is defined as

$$T_T^{ave} = \frac{\sum_{i=1}^{NE} T^{notify/active}(e_i)}{NE} \tag{6.10}$$

6.1.3 Average Number of Notification and Activation Messages

Define $N_{mess}(e)$ as the number of routers that receive the notification or activation messages after the detection of the failure of element e. The average number of routers that can receive the notification or activation messages is defined as

$$N_{mess}^{ave} = \frac{\sum_{i=1}^{NE} N_{mess}(e_i)}{NE} \tag{6.11}$$

$N_{mess}(e)$ and N_{mess}^{ave} give an indication about the control messages that flood the network after the failure of element e. This time spent at each router to process the notification and activation messages has an impact on the total repair time.

6.2 Network Topologies

For the purpose of evaluating the proposed recovery approaches, four graphs are generated using the Waxman model [43], where each graph represents a network. Table 6.1 shows the information for each network in terms of the number of nodes, the number of links and the average node degree. Seven multicast trees were generated in each network, where the ingress LER and the egress LERs are selected randomly. Appendix A presents a description of the simulated environment used to get the results discussed in this chapter. Tables 6.2 through 6.5 show the results of applying the tree division algorithm on each tree. T_{diam} represents the longest path in a tree measured in number of links. $D_{\max}$, $D_{\min}$, and D_{ave} are, respectively, the maximum, the minimum and the average domain size measured in terms of the number of BDRs.

Table 6.1: Information on the four networks used to test the recovery methods.

Network	*Number of nodes*	*Number of links*	*Average degree*
Network 1	42	75	3.60
Network 2	63	142	4.51
Network 3	87	175	4.02
Network 4	122	336	5.51

Table 6.2: Results of applying the tree division algorithm on Network 1.

Tree	N_{LER}	T_{diam}	*ND*	$D_{\max}$	$D_{\min}$	D_{ave}
T1	6	9	3	3	2	2.7
T2	10	7	4	7	2	3.3
T3	8	6	4	5	2	2.8
T4	9	8	3	5	3	3.7
T5	10	9	3	7	2	4.0
T6	14	7	5	8	2	3.6
T7	11	8	4	6	2	3.5

Table 6.3: Results of applying the tree division algorithm on Network 2.

Tree	N_{LER}	T_{diam}	*ND*	$D_{\max}$	$D_{\min}$	D_{ave}
T1	6	12	4	3	2	2.3
T2	10	6	2	9	2	5.5
T3	15	10	6	6	2	3.3
T4	18	8	5	10	2	4.4
T5	19	8	5	13	2	4.6
T6	23	8	5	17	2	5.4
T7	29	8	5	18	2	6.6

Table 6.4: Results of applying the tree division algorithm on Network 3.

Tree	N_{LER}	T_{diam}	*ND*	$D_{\max}$	$D_{\min}$	D_{ave}
T1	4	8	2	3	2	2.5
T2	10	11	3	7	2	4.0
T3	15	10	4	8	2	4.5
T4	16	10	7	5	2	3.1
T5	18	12	9	7	2	2.9
T6	20	12	9	5	2	3.1
T7	23	13	10	7	2	3.2

Table 6.5: Results of applying the tree division algorithm on Network 4.

Tree	N_{LER}	T_{diam}	*ND*	$D_{\max}$	$D_{\min}$	D_{ave}
T1	10	13	6	4	2	2.5
T2	14	12	4	11	2	4.3
T3	18	12	9	7	2	2.9
T4	22	11	9	11	2	3.3
T5	23	14	8	9	2	3.8
T6	23	11	6	16	2	4.7
T7	28	10	12	7	2	3.3

where

N_{LER} : Number of egress LERs in tree T.
T_{diam} : diameter of the tree T.
ND : Number of domains.
$D_{\max}$: Maximum domain size.
$D_{\min}$: Minimum domain size.
D_{ave} : Average domain size.

6.3 Simulation Results

For each network, five types of results are produced that represent the total capacity reserved for each tree, the percentage of the saved capacity (ΔC), the maximum recovery time, the average recovery time and the average number of notification and activation messages. These results are shown in Figure 6.1 through Figure 6.20. In Figures 6.1, 6.6, 6.11 and 6.16, the reserved capacity for each tree is represented by the total number of links used to set up the backup paths. The local recovery approach which is used as a reference for the reserved capacity is the one that is depicted in Figure 4.1. The values of the maximum recovery time (Figures 6.3, 6.8, 6.13, 6.18 and 6.23) and the values of the average recovery time (Figures 6.4, 6.9, 6.14, 6.19 and 6.24) represent time units. These units are assigned randomly to the edges in each network during the simulation.

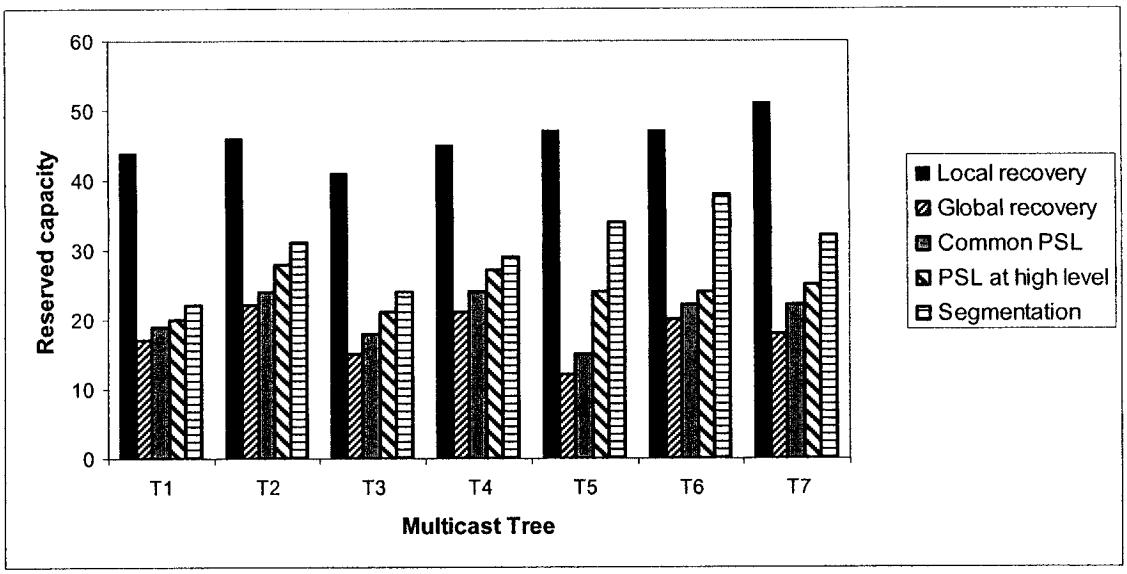


Figure 6.1: The reserved capacity in Network 1.

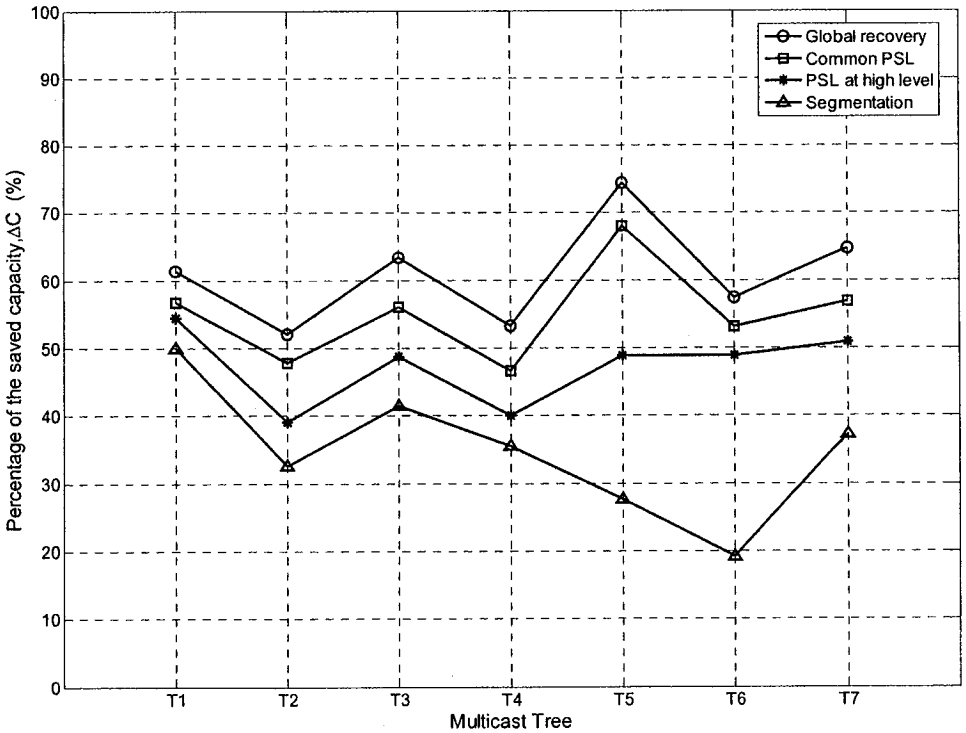


Figure 6.2: The percentage of the saved capacity (ΔC) in Network 1.

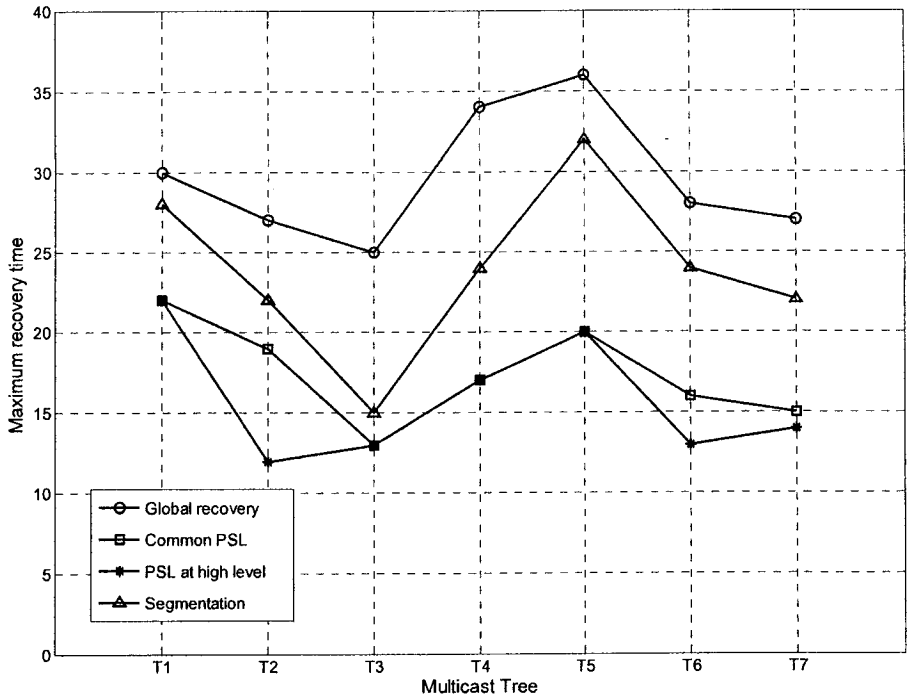


Figure 6.3: Maximum recovery time in Network 1.

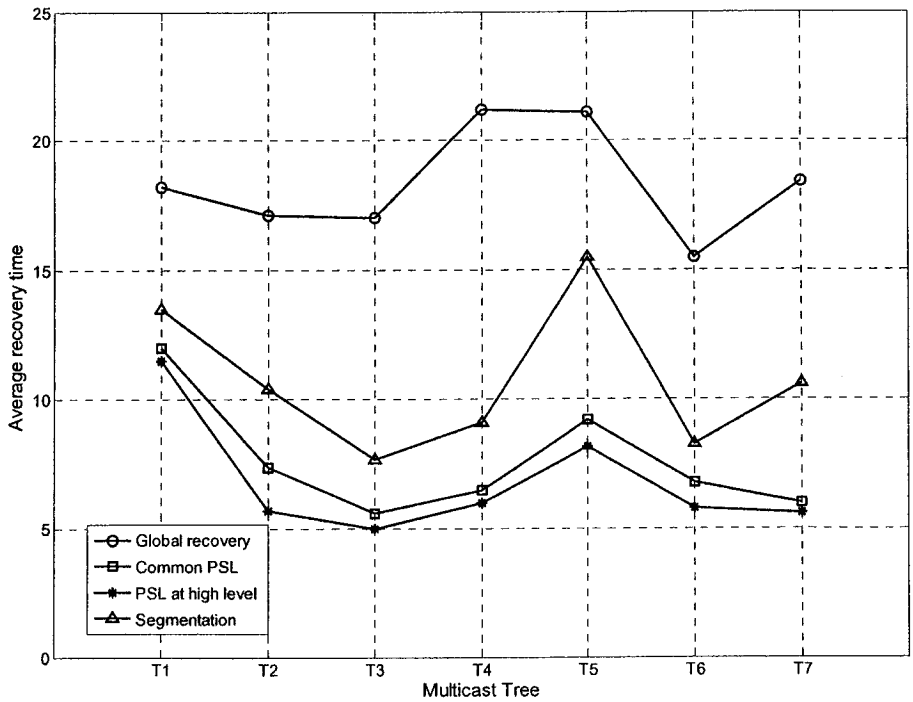


Figure 6.4: Average recovery time in Network 1.

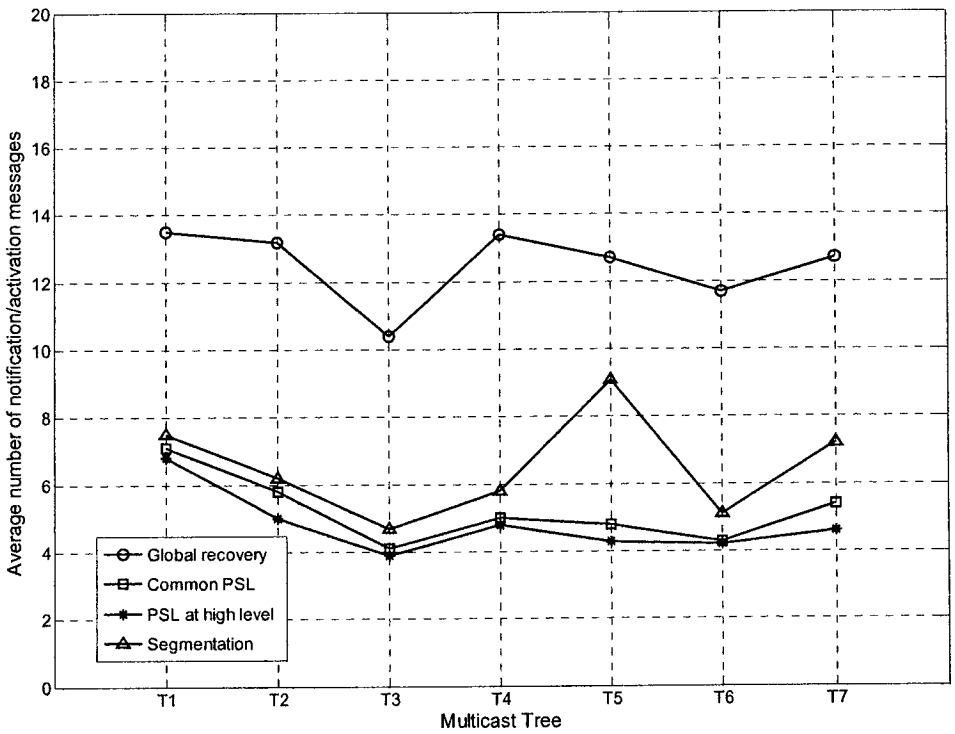


Figure 6.5: Average number of notification and activation messages in Network 1.

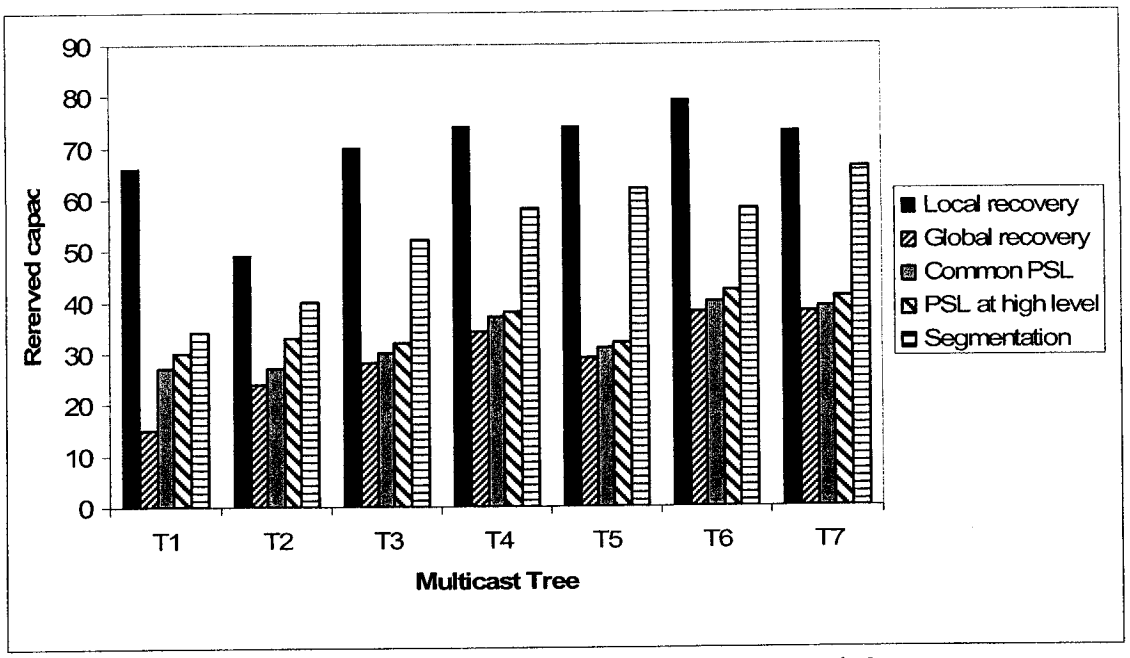


Figure 6.6: The reserved capacity in Network 2.

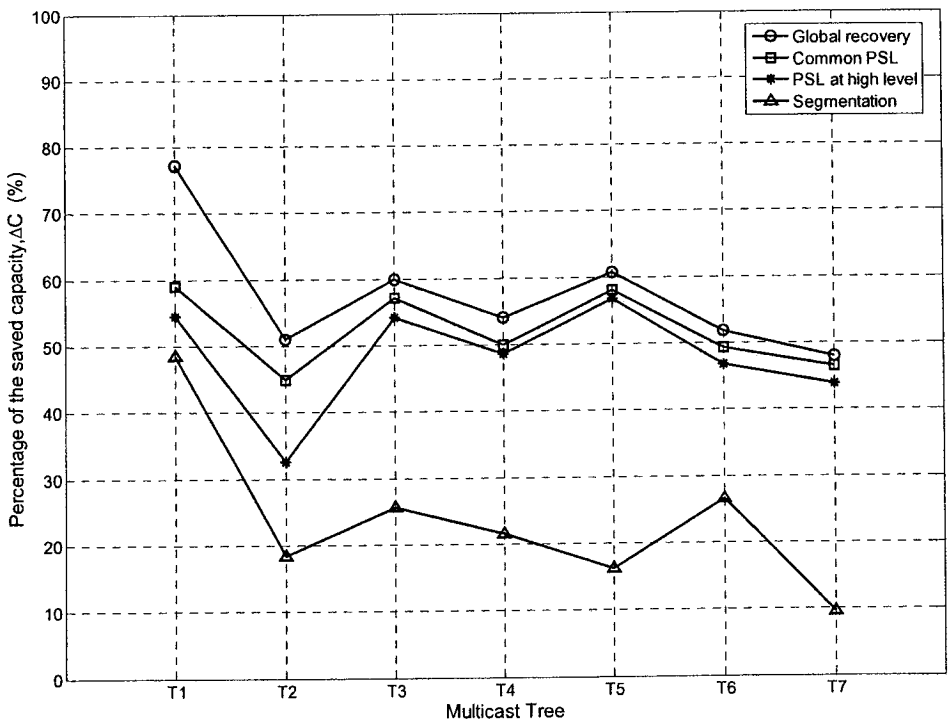


Figure 6.7: The percentage of the saved capacity (ΔC) in Network 2.

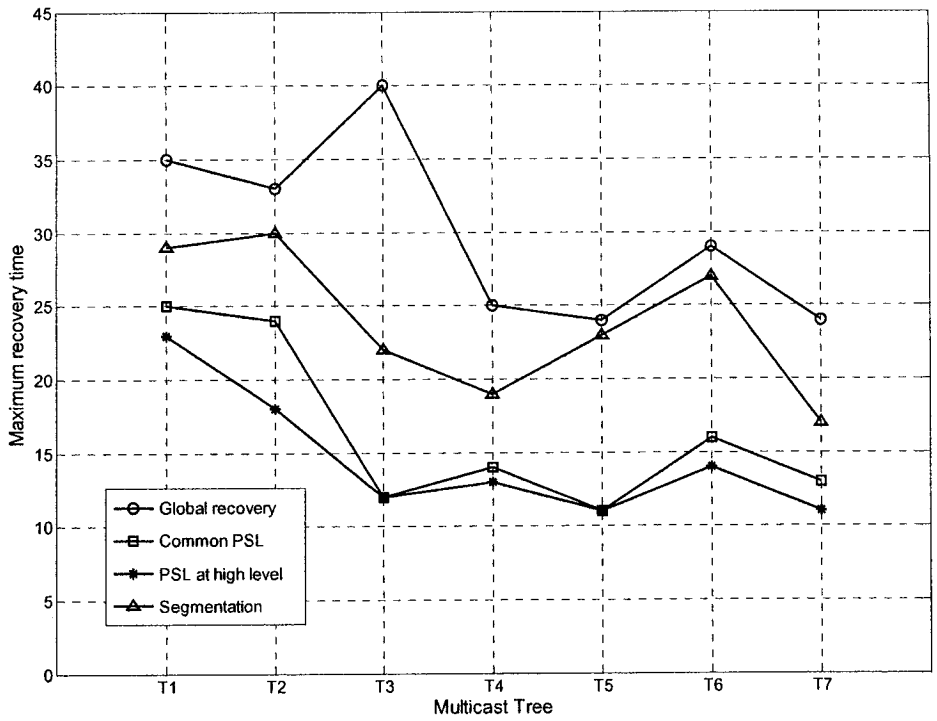


Figure 6.8: Maximum recovery time in Network 2.

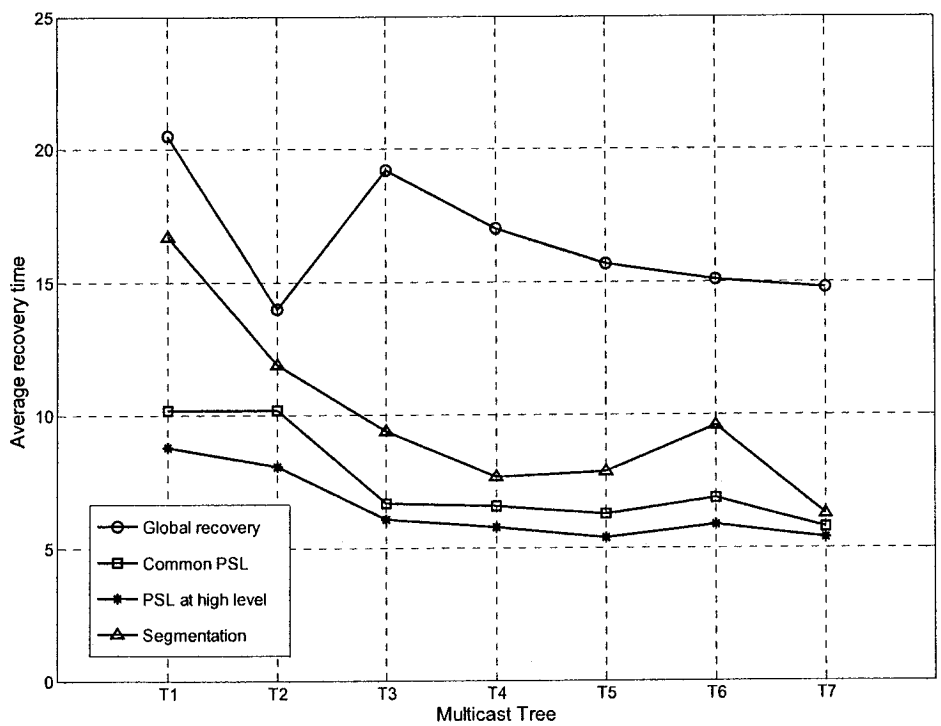


Figure 6.9: Average recovery time in Network 2.

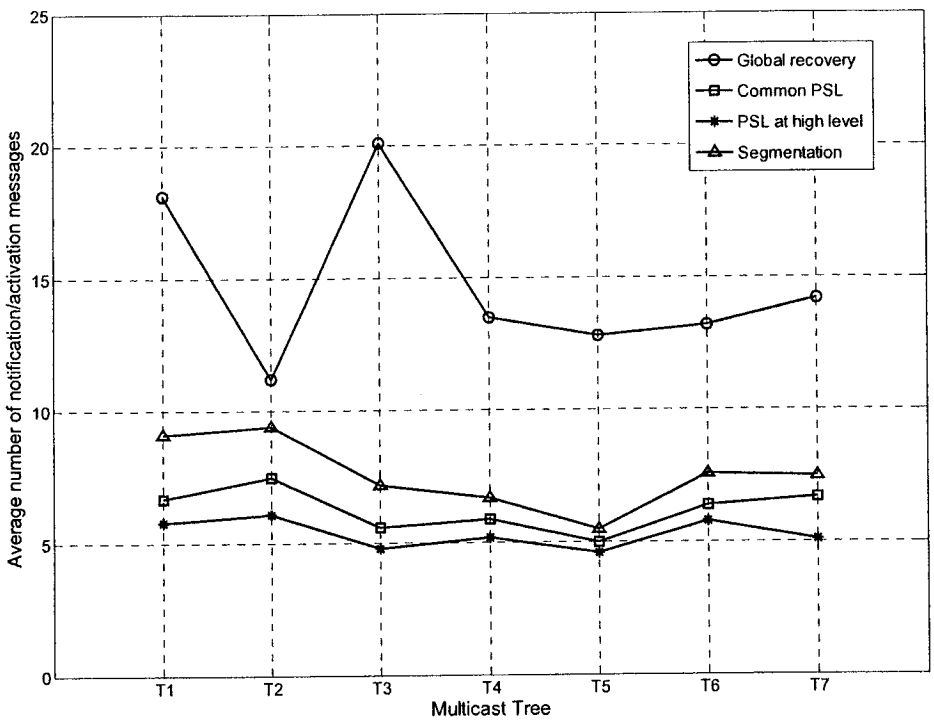


Figure 6.10: Average number of notification and activation messages in Network 2.

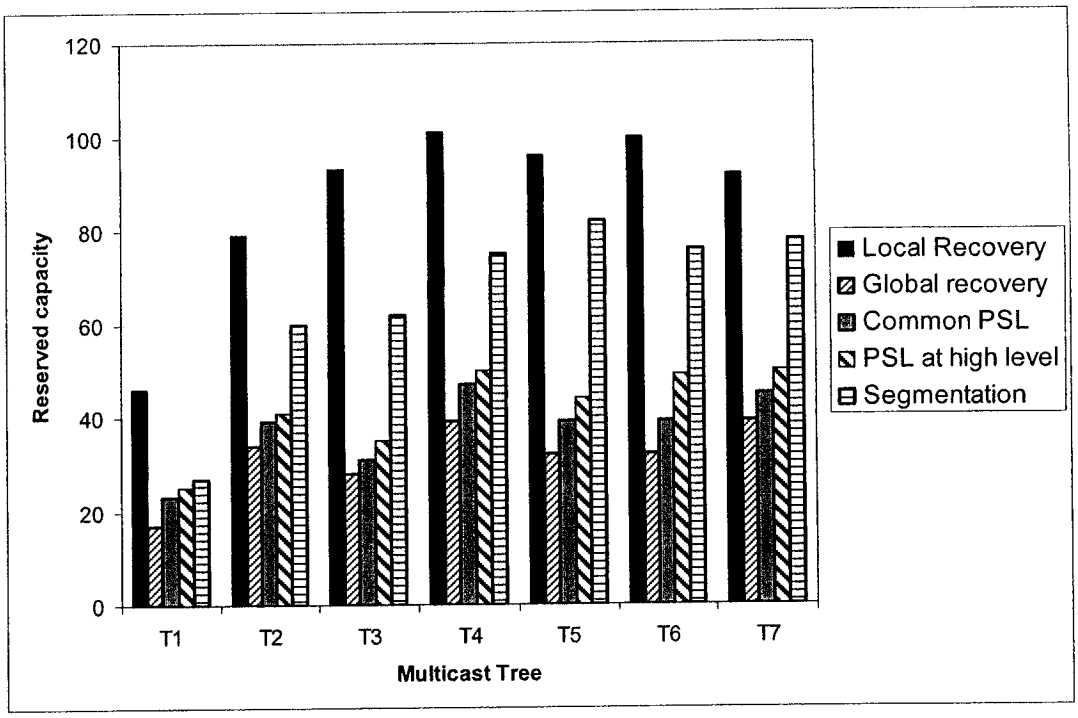


Figure 6.11: The reserved capacity in Network 3.

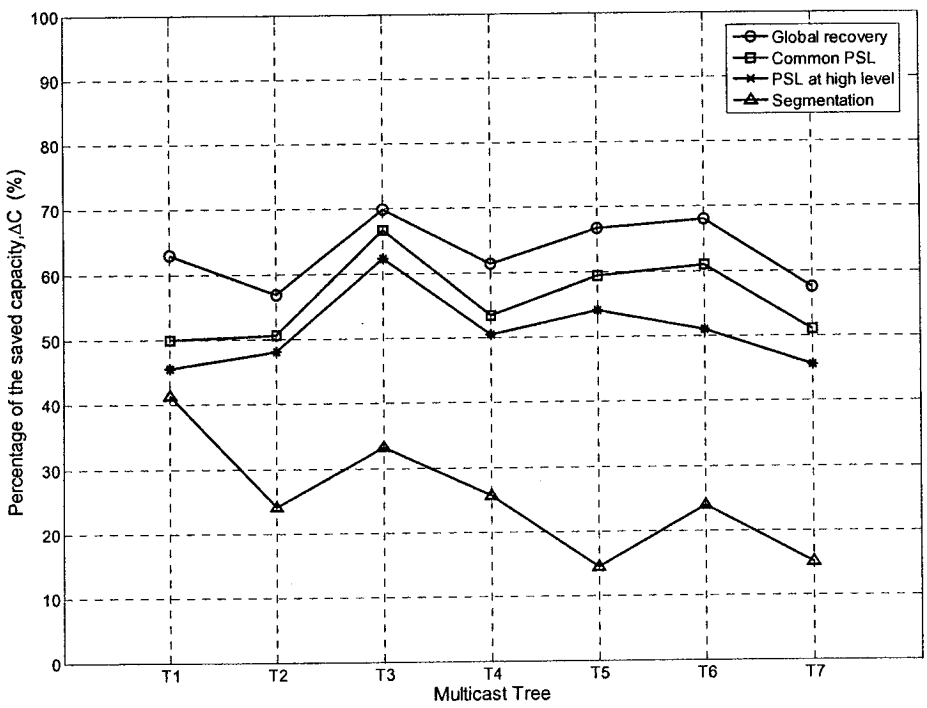


Figure 6.12: The percentage of the saved capacity (ΔC) in Network 3.

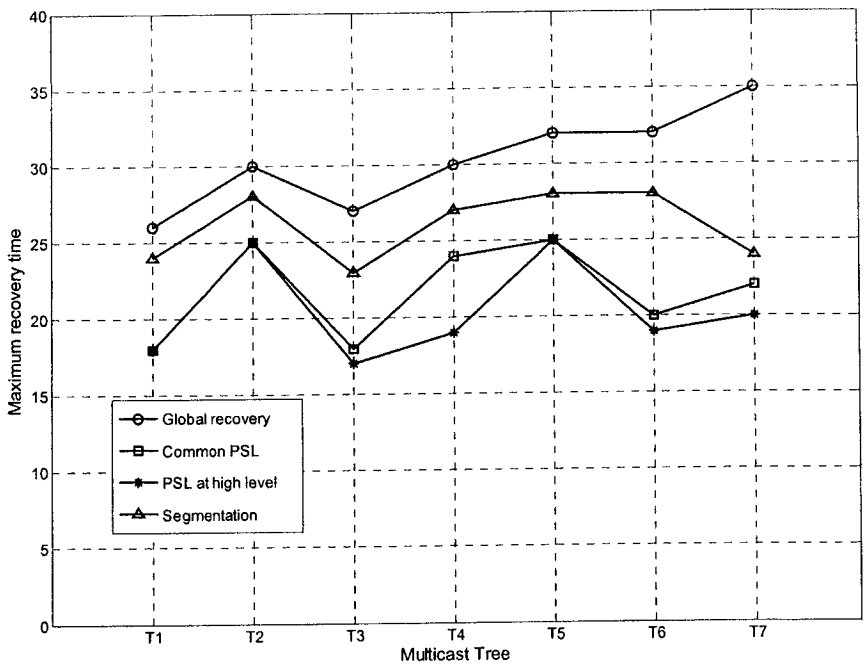


Figure 6.13: Maximum recovery time in Network 3.

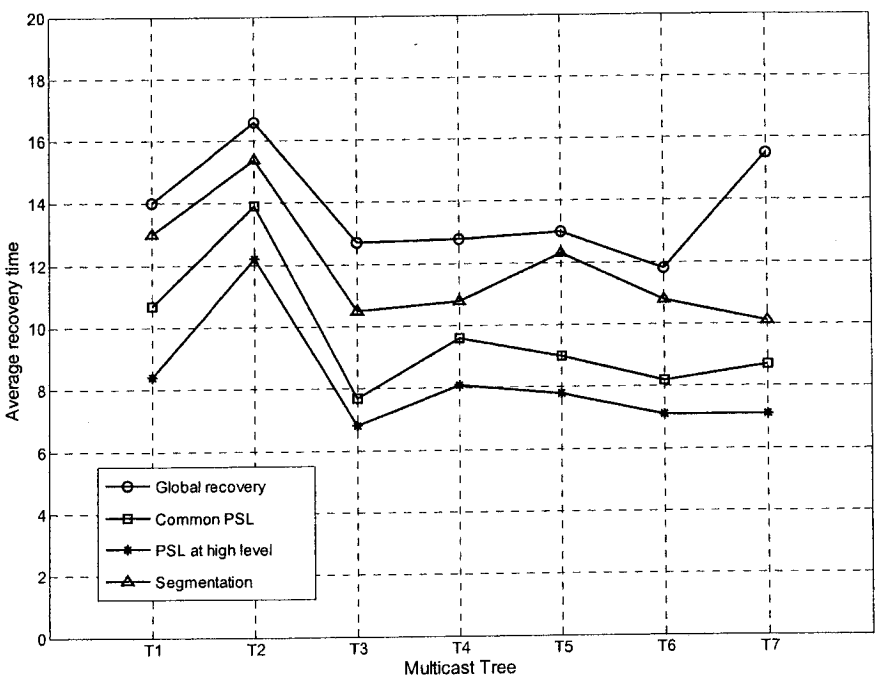


Figure 6.14: Average recovery time in Network 3.

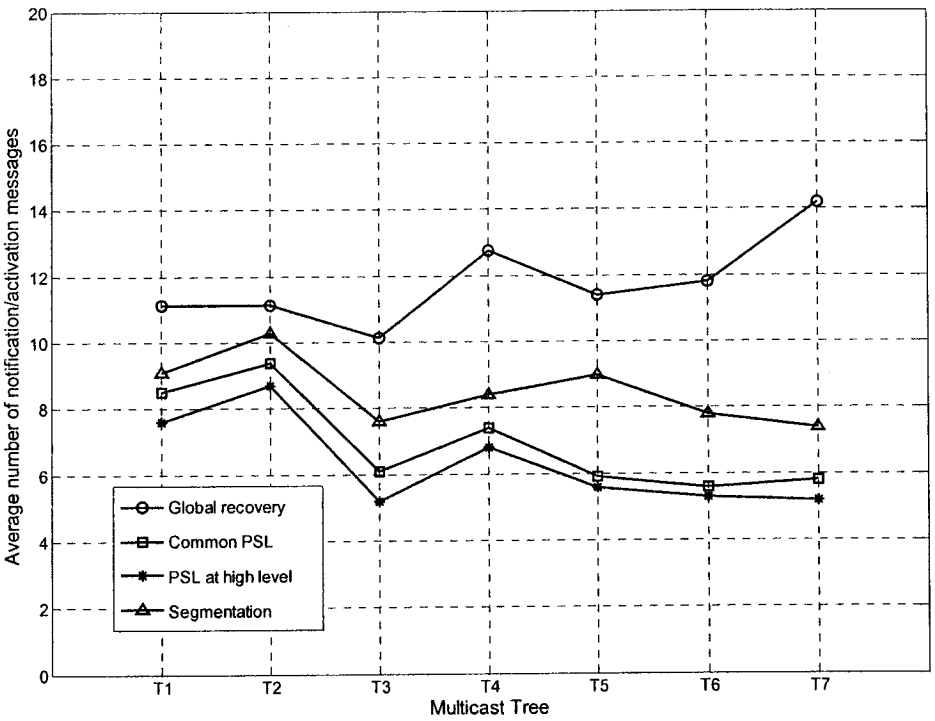


Figure 6.15: Average number of notification and activation messages in Network 3.

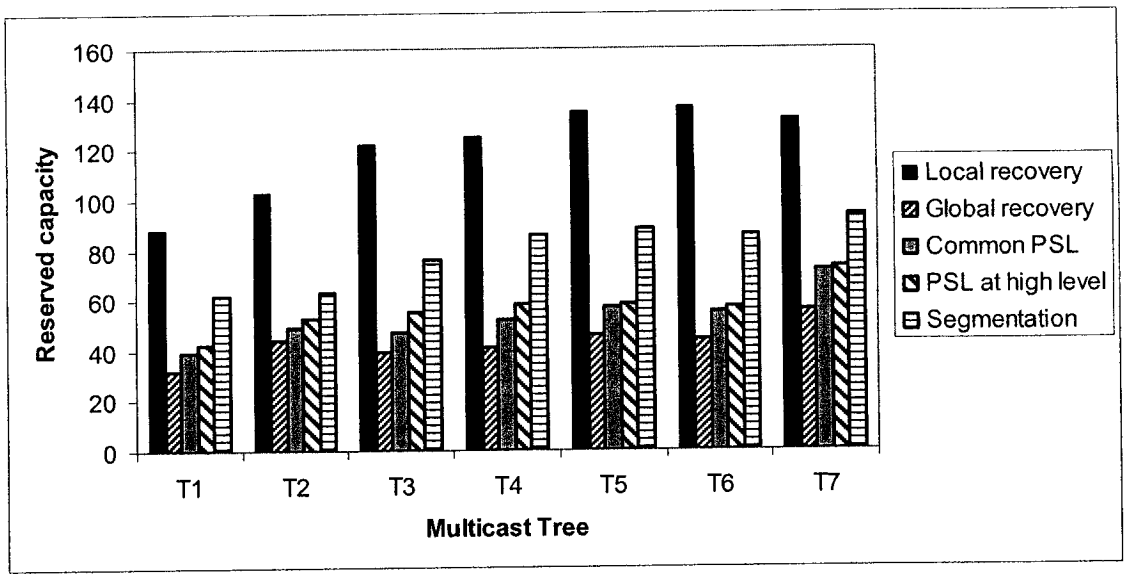


Figure 6.16: The reserved capacity in Network 4.

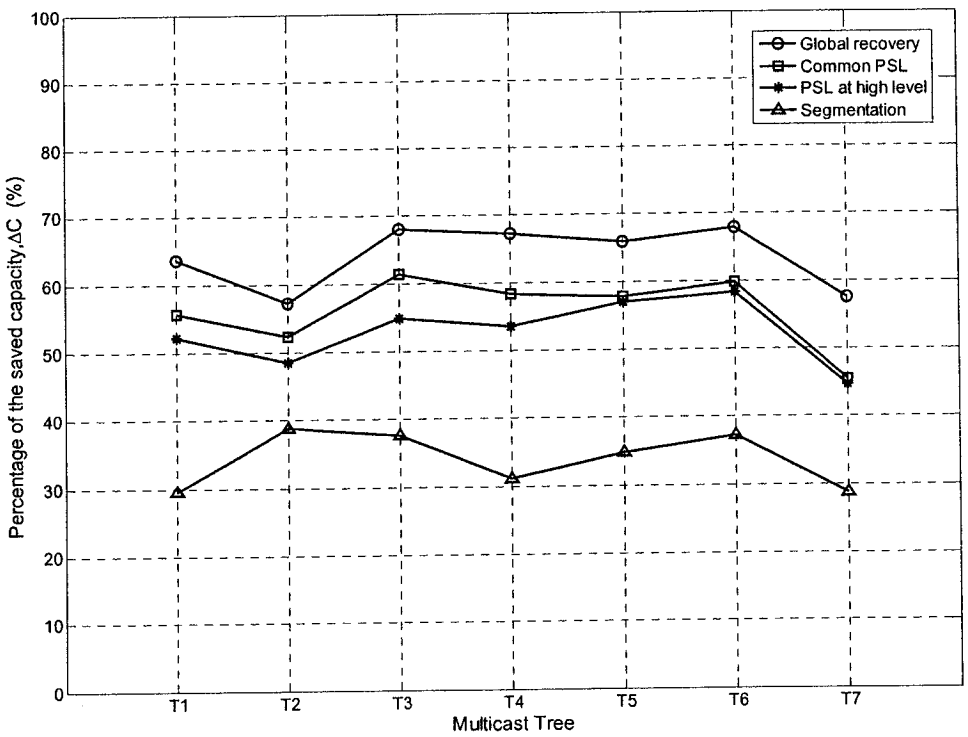


Figure 6.17: The percentage of the saved capacity (ΔC) in Network 4.

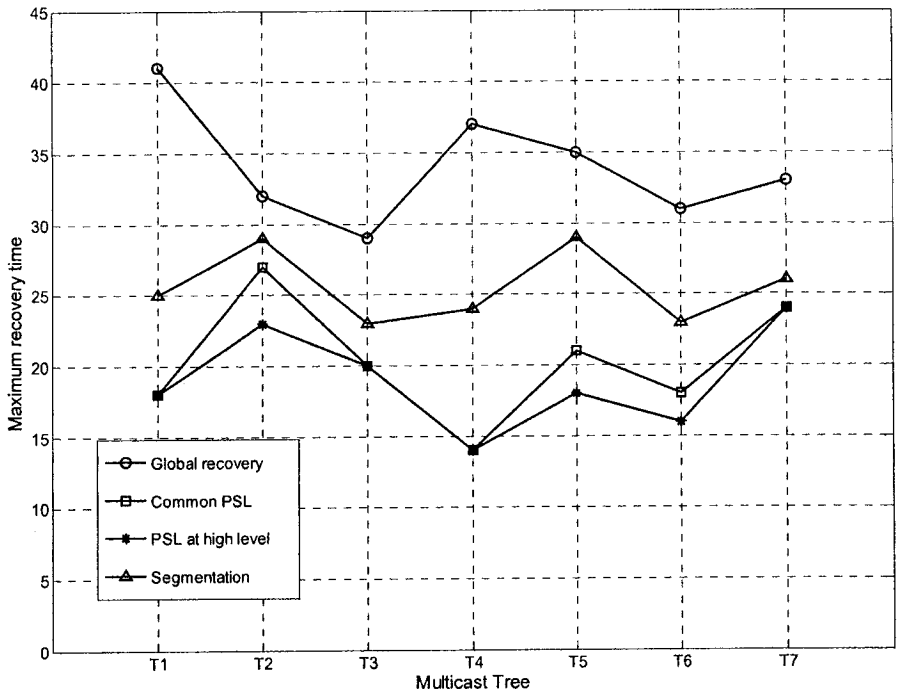


Figure 6.18: Maximum recovery time in Network 4.

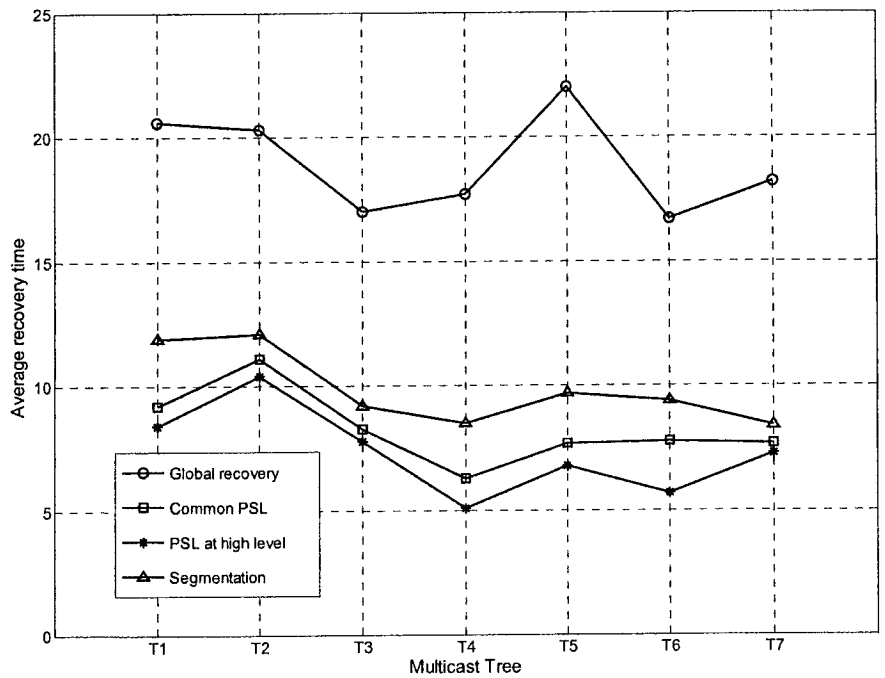


Figure 6.19: Average recovery time in Network 4.

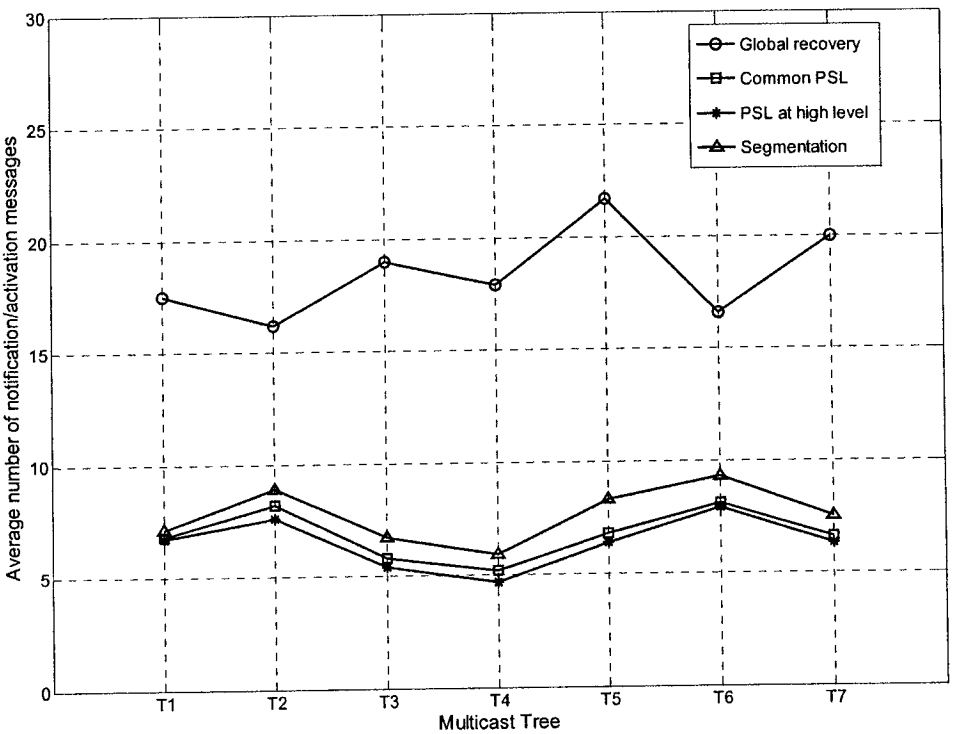


Figure 6.20: Average number of notification and activation messages in Network 4.

6.4 Discussion of the Results

As is shown from the previous figures, the proposed tree division architecture shows a tradeoff between the large amount of bandwidth required for reserving backup paths in local recovery and the large recovery time taken in global recovery. Figures 6.1, 6.6, 6.11 and 6.16 show that the common PSL and PSL at high level methods consume less backup capacity compared with local recovery. In some cases, these methods record a performance that is close to the one produced by the global recovery method. These cases are shown in Figure 6.6 (tree T3 through tree T7) and in Figure 6.11 (tree T3).

In addition, the proposed two methods outperform the segmentation-based approach where the backup paths are set up between the branching routers in the multicast tree [66, 67]. As mentioned in Chapter 4, segmentation-based recovery is considered as a method that supports local recovery for multicasting. In that approach, branching routers are assumed to be non-prone to a failure. However, that assumption is not reasonable. Furthermore, in some cases the saving in the reserved capacity does not increase more than 10% (see Figures 6.6 and 6.7 in tree T7).

In terms of the maximum recovery time, Figures 6.3, 6.8, 613 and 6.18 show that common PSL or PSL at high level methods as well as segmentation-based method outperform the global recovery method. However, the common PSL and PSL at high level methods have a better performance than segmentation-based recovery. For example, segmentation-recovery in Figure 6.3 (tree T1) and in Figure 6.8 (tree T5) has recorded maximum times close to the one recorded by global recovery. Recall that local recovery is the fastest one. Local recovery is not shown in the figures of maximum recovery time

(Figure 6.3, 6.8, 6.13 and 6.18) or in the figures that represent the average recovery time (Figure 6.4, 6.9, 6.14 and 6.19) because it is assumed to be zero.

An interesting point that should be mentioned is the comparison between the common PSL method and the PSL at high level method. From the previous discussion, both methods outperform the global-based recovery and segmentation-based recovery in terms of the maximum recovery time, the average recovery time and the number of notification/activation messages. However, the common PSL method has better performance in terms of the reserved capacity while the PSL at high level method outperforms the common PSL method in terms of the time and notification/activation messages metrics. In some cases, these two methods records the same maximum time. These cases are given in Figure 6.3 (trees T3, T4 and T5), Figure 6.8 (trees T3 and T5), Figure 6.13 (trees T1, T2 and T5) and Figure 6.18 (trees T1, T3, T4 and T7). The justification of that point is that the domain where the maximum recovery time has produced is the same in the two methods. This can happen when a domain has only one BR and all the BDRs in that domain are connected to the root through that BR. Figure 6.21 illustrates an example of a domain where the backup paths are the same whether the common PSL method or the PSL at high level method is used. Indeed, the common PSL method is considered as a special case of the PSL at high level method. In order to ensure the performance of the PSL at high level method over the common PSL method, eight domains D1 through D8 are selected from the multicast trees generated in Network 3. For each domain *i*, we define ζ_i as ratio of the number of BRs to the number of BDRs in that domain. Mathematically, ζ_i is written as:

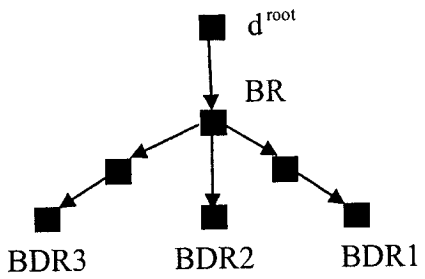


Figure 6.21: An example of a domain where the backup paths will be the same whether the common PSL method or the PSL at high level method is used.

Table 6.6: The eight domains generated in Network 3.

Domain	*Number of BDRs*	ζ
D1	7	0.43
D2	8	0.63
D3	5	0.80
D4	7	0.43
D5	5	0.60
D6	13	0.54
D7	5	0.60
D8	4	0.75

$$\zeta_i = \frac{N_{BR}^i}{N_{BDR}^i}$$

where N_{BR}^i and N_{BDR}^i are the number of BRs and the number of BDRs, respectively, in domain i. Table 6.6 gives the description of the eight domains in terms of the number of BDRs in each domain and the value of ζ_i.

Each domain is treated as a stand-alone tree. The common PSL and the PSL at high level methods are tested in each domain. In terms of the reserved capacity, Figure 6.22

shows that the common PSL method has better performance. Figures 6.23 through 6.25 show that the PSL at high level method has better performance in terms of the average and maximum times and in terms of the average number of notification and activation messages in the domains.

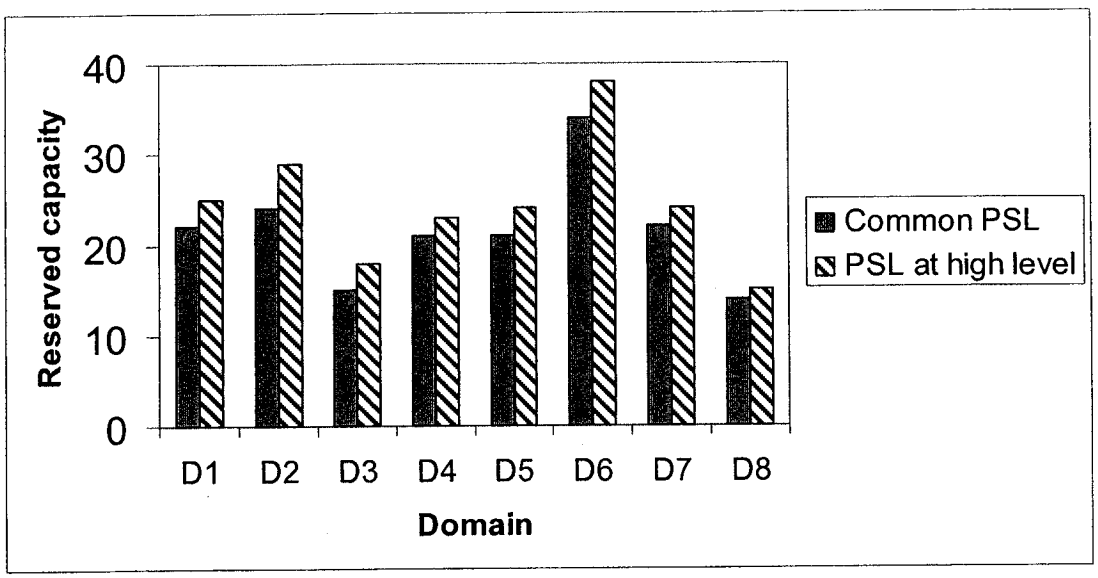


Figure 6.22: The reserved capacity in the domains.

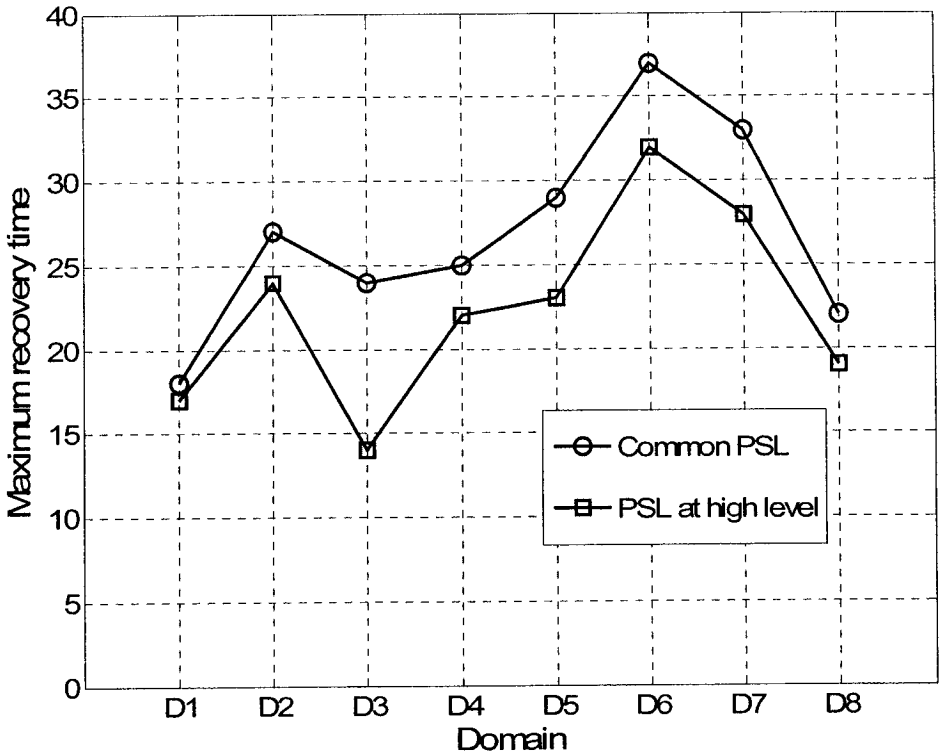


Figure 6.23: Maximum recovery time in the domains.

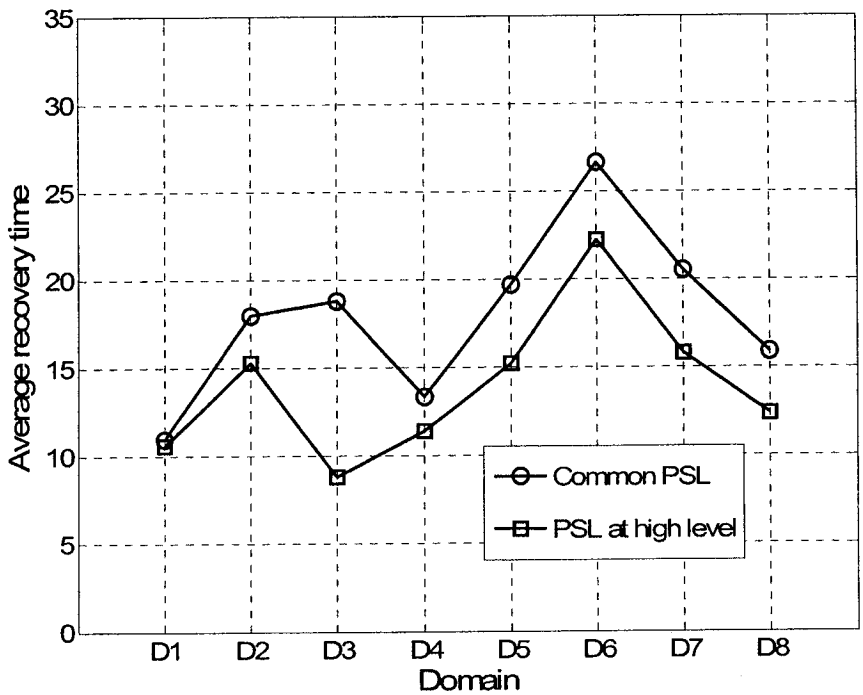


Figure 6.24: Average recovery time in the domains.

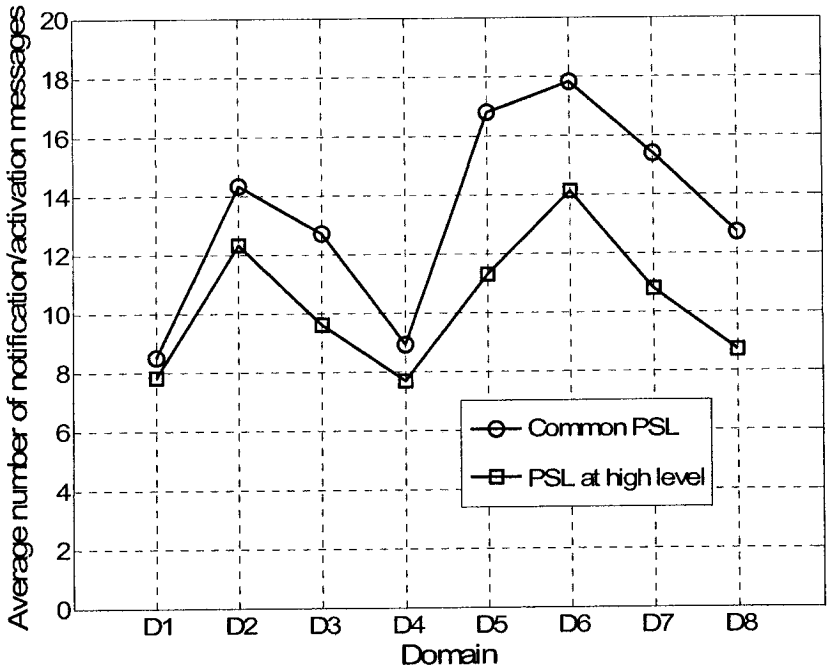


Figure 6.25: Average number of notification and activation messages in the domains.

As we have seen from the results discussed above, the common PSL method and the PSL at high level method have close performance. The difference between these two methods is related to the way the backup paths are built inside the domains. In the common PSL method, a domain root initiates the process of building all the backup paths in that domain where in PSL at high level method, each BDR has the choice to select its PSL router.

Chapter 7

Conclusions and Future Work

Multicast communication has proved to be an efficient solution for most applications that consume a large amount of network bandwidth such as video-on-demand and teleconferencing applications. On the other hand, MPLS has emerged as an elegant solution to overcome most of the problems that current IP-based networks suffer. However, integrating MPLS with multicasting still suffers from some problems. This thesis has introduced solutions for two of these problems: scalability and MPLS-based recovery. In the next section, we point out the main conclusions of this thesis. Then, in Section 7.2, we introduce some directions for future work.

7.1 Conclusions

Multicasting still suffers from the scalability problem. In current IP networks, a multicast router should keep a forwarding state entry for each multicast session passing through it. As the number of multicast sessions grows, the number of multicast entries

will grow in linear relationship. In backbone networks, the same multicast tree might be used to deliver the IP packets of different multicast sessions. One feature of MPLS is its ability to aggregate different flows that use the same path inside the backbone network. In MPLS, aggregation means the classification of different flows to the same FEC and then assigning them the same MPLS label. In unicast communication, MPLS relies on the IP address to classify the IP packets of different flows into the same FEC. In multicasting, it is impossible to rely on the IP address to aggregate different multicast flows that follow the same tree. Some mechanisms have to be proposed in order to distinguish the structure of each multicast tree that is built over the MPLS network. Once the tree structure is distinguished, it will be easy to aggregate different multicast flows to the same FEC. Then, only one MPLS label will be assigned to each IP packet that belongs to that FEC.

This thesis has introduced two mechanisms to distinguish the structure of the multicast trees: state Encoding (SE) and Tree Numbering (TN). Chapter 3 of this thesis has discussed in detail the concepts and the mechanisms of implementing SE and TN in the IP networks. It was clearly shown that label aggregation approaches (TNT, SE, TN) provide a significant saving in the usage of the labels. However, these approaches maintain different memory sizes at the ingress LER. In terms of the memory size required to maintain the tree codes or the tree numbers, SE and TN approaches outperform the TNT approach. A typical backbone MPLS network would have large size and the average number of LSRs and egress LERs for each tree could be in the range of hundreds. That makes TN a preferable approach to support label aggregation, not only because the TN

approach consumes less memory size compared with SE and TNT approaches, but also because it uses a fixed size of bits to represent tree numbers.

One of the objectives behind introducing MPLS is to speed up the forwarding process of the IP packets inside the MPLS networks. That objective is achieved by relieving the LSRs from the heavy processing of the IP packets, which is left for the LERs. Unicast communication is fully compatible with that objective. Once a LSP is set up, all the IP unicast packets that use that path are switched based on the labels they carry. In multicasting, most of the delivery trees have branching points. At each branching point, a multicast packet is duplicated and sent over each link of that point. Some approaches that have been proposed to support multicasting in MPLS networks require the recourse to the traditional IP forwarding. As the number of branching points in a certain tree increases, the traditional IP forwarding increases too. For the purpose of measuring the compatibility of any proposed approach with the MPLS protocol, we have proposed $MPLS_{comp}$ as metric in order to give an indication to that compatibility. While SE, TN ant TNT approaches are fully compatible with the MPLS protocol, some approaches such as LBP and ERM mostly record less than that.

Network survivability when faults occur is important, especially when continuous media are carried. Despite the notable attention that has been given to MPLS-based recovery for unicast communication, little research has been done that targets MPLS-based recovery for multicast communication. MPLS-based recovery can be implemented using local recovery, in which a backup path is set up for each element in the multicast tree. The drawback of local recovery is that a large amount of capacity is reserved over the backup paths. The amount of reserved capacity can be reduced if global (end-to-end)

recovery is used. However, the disadvantage of global recovery is that it takes a longer time to recover from a failure.

In Chapter 5 of this thesis, we have proposed an MPLS-based architecture that trades off between the large amount of bandwidth required for reserving backup paths in local recovery and the large recovery time taken in global recovery. A new tree division approach is proposed. In this approach, a multicast tree is divided into several domains, where each domain represents a local stand-alone sub-tree of the original one. Two MPLS-based methods are then proposed to set up the backup paths inside the domains. These methods are called: the common PSL method and the PSL at high level method.

In order to study the performance of the proposed architecture, three metrics have been proposed: the total backup capacity, the maximum and the average notification times, and the average number of the notification messages that are produced as a result of a failure. A comparison is made among the local recovery approach, the global recovery approach, a method that sets up the backup paths between the branching points, and the two methods used in the proposed architecture. Chapter 6 is dedicated to presenting and discussing the results. In terms of the reserved capacity, the results have shown that our architecture consumes backup capacity close to that consumed by the global recovery. However, in terms of the average notification time and the average number of notification and activation messages our approach outperforms the global recovery approach and the one that sets up the backup paths between the branching points.

Moreover, dynamic behavior is a basic feature of multicasting. In the last section of Chapter 5, we have used the probability-based analysis to study the effect of Join-Request

and Prune-Request messages on the domain topologies. Three types of changes can take place on the topologies of some domains as a result of receiving a Join-Request or a Prune-Request message. These changes are:

1. A domain is divided into two sub-domains.
2. A merging between two domains.
3. A shift in a domain.

If none of these changes occurs, we call the change that occurs on the topology of the domain as a result of receiving a Join/Prune message the simple change. The probability-based analysis has shown that the probability of the occurrence of the simple change is the maximum, and it is always greater than 0.5. This leads us to the conclusion that the cost of the proposed MPLS-based recovery architecture when the maintenance of the multicast tree should be done is not high. Only the necessary action should be taken, which is the release of the backup path when a Prune-Request message is received, or the setup of a new backup path when a Join-Request message is received.

7.2 Future Research Directions

The following points are directions for our future work:

- In both parts of this dissertation, we have discussed the actions that should be taken when a new member joins or a current member leaves the multicast session. However, a comparative study of this dynamic behavior for the different approaches that support multicast communication in MPLS networks as discussed in the first part, and for the recovery approaches in the second part as well would be a significant direction for future work.

- The proposed Tree Numbering approach is discussed based on multicast protocols that build a source tree. It is interesting to study the Tree Numbering approach in the case of the use of multicast protocols based on a Rendez-Vous point such as CBT.
- The recovery algorithm called PSL at the high level does not take all the BDRs in all the higher levels into consideration. The algorithm stops searching when the S_{PSL} set is not empty. A modified version of the proposed algorithm that takes all the possible PSL routers into consideration could be a good direction for future work in order to enhance the results obtained by using the PSL at high level method.
- An interesting idea would be the reconsideration of the proposed tree division scheme and the recovery algorithms in networks where inter-demand sharing is assumed. In other words, the setup of the backup paths would target the minimization of the shared capacity for several trees as an objective function.
- It is an interesting idea if some results can be obtained by testing the proposed algorithms in the two parts on a real world network or using commercial simulators such as Opnet whether they are tested separately or integrated together. Furthermore, a good idea would be to carry out a validation of the proposed algorithms using any tool that employs any modeling language such as Promela language.

REFERENCES

[1] S. Deering and D. Cheriton, "A Multicast Extension to the Internet Protocol," RFC 966, December 1985.

[2] S. Deering, "Multicast routing in a datagram internetwork," PhD thesis, Stanford University, December 1991.

[3] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, January 2001.

[4] White paper, "Multiprotocol Label Switching (MPLS)," http://www.iec.org/online/tutorials/mpls.

[5] C. K. Miller, "Multicast Networking and Applications," Addison Wesley, 2000.

[6] R. Wittmann and M. Zitterbart, "Multicast Communication: Protocols and Applications," Morgan Kaufmann Publishers, 2001.

[7] B. Quinn et al., "IP Multicast Applications: Challenges and Solutions," RFC 3170, September 2001.

[8] S. Paul, "Multicasting on the Internet and its Applications," Kluwer Academic Publisher, 1998.

[9] W. Fenner, "Internet Group Management Protocol, Version 2," RFC 2236, November 1997.

[10] B. Wang and J. Hou, "Multicast Routing and Its QoS Extension: Problems, Algorithms, and Protocols," *IEEE Network*, Jan./Feb. 2000.

[11] L. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network*, Jan./Feb. 2000.

[12] D. Waitzman and C. Partridge, "Distance Vector Multicast Routing Protocol," RFC 1075, November 1988.

[13] J. Moy, "MOSPF: Analysis and Experience,", RFC 1585, March 1994.

[14] J. Moy, "Multicast Routing Extensions for OSPF," *Communications of the ACM*, Vol. 37, Aug. 1994, pp. 61-66.

[15] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[16] A. Adams, J. Nicholas W. Siadak, "Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification," RFC 3973, January 2005.

[17] D. Estrin et al., "Protocol Independent Multicast-Spare Mode (PIM-SM): Protocol Specification," RFC 2362, June 1998.

[18] T. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," RFC 2201, September 1997.

[19] T. Ballardie et al., "Core Based Trees (CBT): An Architecture for scalable inter-Domain Multicast Routing," *Comp. Commun.* Rev. Vol. 23, Oct. 1993, pp. 85-95.

[20] B. Jamoussi et al., "Constraint-Based LSP Setup using LDP", RFC 3212, January 2002.

[21] D. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[22] B. Davie and Y. Rekhter, "MPLS: Technology and Applications," Morgan Kaufmann Publisher, 1st edition, 2000.

[23] D. Awduche and B. Jabbari, "Internet Traffic Engineering Using Multi-Protocol Label Switching," *Computer Networks*, Vol. 40, pp. 111-129, 2002.

[24] D. Awduche, "MPLS and Traffic Engineering in IP Networks," *IEEE*

Communication Magazine, December 2001.

[25] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic Engineering with MPLS in the Internet," *IEEE Network Magazine*, March 2000.

[26] D. Ooms et al., "Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment," RFC 3353, August 2002.

[27] D. Ooms and W. Livens, "IP Multicast in MPLS Networks," *Proceedings of the IEEE Conference on High Performance Switching and Routing*," June 2000, pp. 301-305.

[28] U. Black, "MPLS and Switching Networks," Prentice Hall PTR, 2001.

[29] A. Acharya, F. Griffoul and F. Ansari, "IP Multicast Support in MPLS," *IEEE Proceedings of the ATM Workshop*, 1999. pp. 211 - 218.

[30] A. Acharya and F. Griffoul, "Native IP Multicast in MPLS," *Proceedings of The First International COST264 Workshop on Network Group Communication*. 1999, pp. 204 – 215.

[31] O. Banimelhem, J. W. Atwood and A. Agarwal, "An Approach to solving a Multicasting Scalability Issue in MPLS Networks Using State Encoding," *IEEE CCECE*, May 2-5 2004, Niagara Falls, Canada.

[32] R. Braden et al., "Resource ReServation Protocol (RSVP)," RFC 2205, September 1997.

[33] O. Banimelhem, J. W. Atwood and A. Agarwal, "Deploying Multicast Communication over MPLS Networks Using Tree Numbering," *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005),* pp. 704–711.

[34] J. Chung et al., "MPLS Multicasting Through Enhanced LDP and RSVP-TE Control," *The 45th Midwest Symposium on Circuit and Systems, MWSCAS02*, Vol. 3, pp. 93-96, 2002.

[35] Z. Zhang et al., "The New Mechanism for MPLS Supporting IP Multicast," *IEEE APCCAS 2000*, pp. 247-250.

[36] A. Boudani and B. Cousin, "A new Approach to Construct Multicast Trees in MPLS Networks," *Seventh IEEE Symposium on Computers and Communications (ISCC)*, Taormina, Italy, July 2002

[37] A. Boudani et al., "Multicast Routing Simulator over MPLS Networks", *36th Annual Simulation Symposium*, April 2003.

[38] B. Yang and P. Mohapatra, "Edge Router Multicasting with MPLS Traffic Engineering," *10th IEEE International Conference on Networks, ICON02*, August 2002.

[39] B. Yang and P. Mohapatra, "Multicasting in MPLS domains," *Computer Communications*, Vol. 27, pp. 162-170, 2004.

[40] Y. Oh et al., "Scalable MPLS Multicast Using Label Aggregation in Internet Broadcasting Systems," *Proceedings of the 10th International Conference on Telecommunications, ICT'03*, February 23 – March 1, 2003.

[41] W. D. Grover, "The self healing network: a fast distributed restoration technique for networks using digital cross connect machines," *Proceedings of the IEEE GLOBECOM'87*, pp. 1090-1095.

[42] K. Murakami and H. Kim, "Joint optimization of capacity and flow assignments for self-healing ATM networks," *Proceedings of the IEEE ICC'95*, pp. 216-220.

[43] B. Waxman, "Routing of Multipoint Connections", *IEEE Journal of Selected Areas in Communication,* pp. 1617-1622, Vol. 6, No. 9, December 1988.

[44] Dongyun Z. and Suresh S., "Survivability in Optical Networks," *IEEE Network*, pp. 16-23, Vol. 14, No. 6, Dec. 2000.

[45] T.–H. Wu, "Emerging technologies for fiber network survivability," *IEEE Communications Magazine*, vol. 33, No. 2, pp. 58-74, 1995.

[46] T. Frisanco, "Optimal Spare Capacity Design for Various Protection Switching Methods in ATM Networks," *Proceedings of ICC'97*, pp. 293-298.

[47] M. Kodialam and T.V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restorations", *Proceedings of IEEE Infocom 2000*, pp. 902-911.

[48] M. Kodialam and T.V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information", *Proceedings of IEEE Infocom 2001*, Apr. 2001, pp. 376-385.

[49] Bellcore. SR-SR-NWT-001756, Automatic Protection Switching for SONET, Issue 1, October 1990.

[50] T.–H. Wu and R. C. Lau, "A class of self-healing ring architectures for SONET network applications," *IEEE Transactions on Communications*, vol. 40, Issue 11, Nov. 1992, pp.1746 – 1756.

[51] P. Demeester et al., "Resilience in Multilayer Networks," *IEEE Communications Magazine*, Vol. 37, No. 8, pp. 70-76, 1999.

[52] V. Sharma and F. Hellstrand, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery," RFC 3469, February 2003.

[53] A. Farrel and B. Miller, "Surviving failures in MPLS networks," Technical report,

Data Connection, February 2001.

[54] C. Huang, V. Sharma, K. Owens and S. Makam, "Building Reliable MPLS Networks Using a Path Protection Mechanism," *IEEE Communications Magazine*, Vol. 40, March 2002, pp. 156 – 162.

[55] S. Wang, "Survivable Label Switching Networks," Ph.D Thesis, University of California, Los Angeles, 2002.

[56] A. Bremier-Brarr, Y. Afek and H. Kaplan, Edit Cohen and Michael Merritt, "Fast Recovery of MPLS Paths," ACM SIGMETRICS, Jan. 2001.

[57] H. Oh, M. Thomas and L. Jeffery, "Fault Restoration and Spare Capacity Allocation with QoS Constraints for MPLS Networks," *Proceedings of the IEEE GLOBECOM'00*, Nov. 2000, pp. 1731-1735.

[58] R. Bartos and M. Ramn, "A Heuristic Approach to Service Restoration in MPLS Networks," *Proceedings of the IEEE International Conference on Communications*, June 2001, pp. 117-121.

[59] L. Qingsheng, "Fast Failure Recovery in MPLS Networks", Master Thesis, University of British Columbia, September 2002.

[60] S. Dong and C. Phillips, "A New Service Restoration Scheme in MPLS Networks," Proceedings *of the ICT2002*. Beijing, June 2002.

[61] P. Ho and H. T. Mouftah, "Reconfiguration of Spare Capacity for MPLS-based Recovery for the Internet Backbone Networks," *IEEE/ACM Transactions on Networking*, Vol. 12, No. 1, Feb. 2004, pp. 73 - 85.

[62] Y. Pointurier, "Link Failure Recovery for MPLS Networks with Multicasting", Master Thesis, University of Virginia, August 2002.

[63] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed

Multicasts with Failure Backup," *Proceedings of the 10th IEEE International Conference on Network Protocols*, 2002.

[64] J. Cui, M. Faloutsos and M. Gerla, "An Architecture for Scalable, Efficient, and Fast Fault-Tolerant Multicast Provisioning," *IEEE Network*, vol. 18, Issue 2, March-April, 2004, pp. 26-43.

[65] A. Fei, J. Cui, M. Gerla, and M. Faloutsos, "Aggregated Multicast: an Approach to Reduce Multicast State," *Proceedings of the Sixth Global Internet Symposium*, GI'01, November 2001.

[66] R. Deshmukh, A. Agarwal, "Failure Recovery in MPLS Multicast Network using Segmented Backup Approach," *International Conference on Networking*, March 2004, Guadeloupe, French Carribbean, pp. 344-349.

[67] R. Deshmukh, "Failure recovery in MPLS multicast networks using a segmented backup approach," Master Thesis, Concordia University, 2004.

[68] F. Buckley and M. Lewinter, "A Friendly Introduction to Graph Theory," Upper Saddle River, N. J.: Prentice Hall, 2003.

[69] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeman and Company, New York, 1979.

Appendix A: The architecture of the Programs used to produce the results

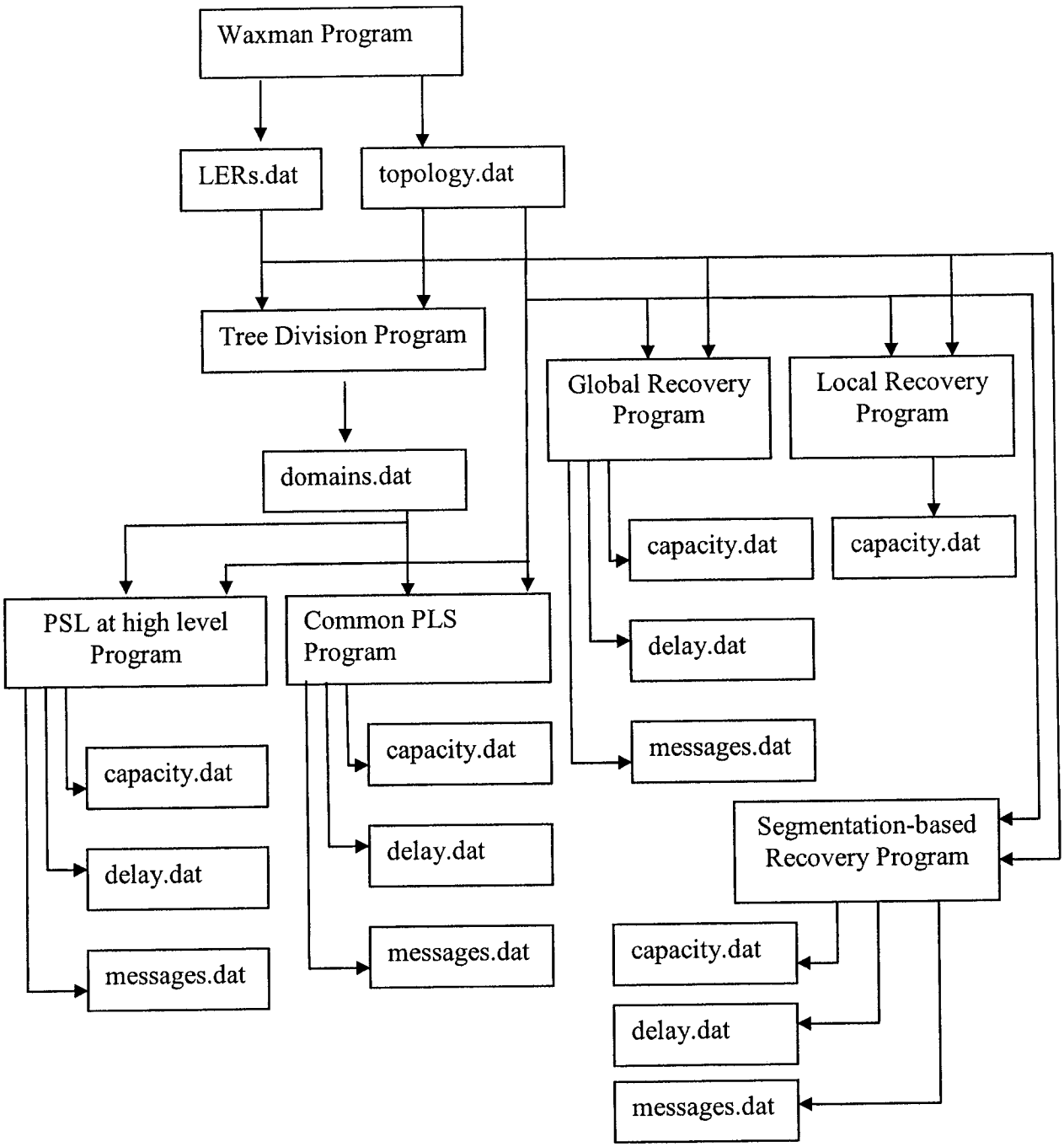


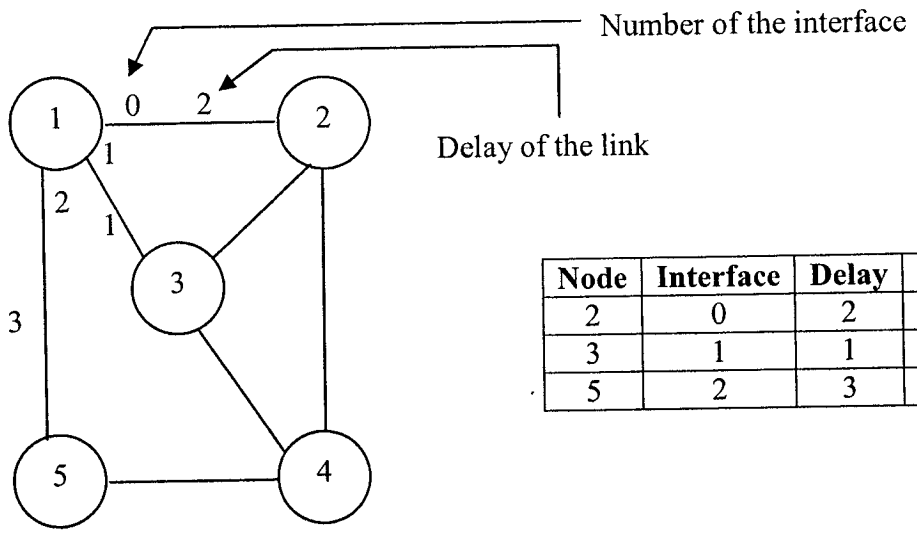
Figure A.1: The architecture of the simulation environment used to evaluate the recovery schemes.

The architecture of the simulation environment used to evaluate the performance of the different MPSL-based recovery schemes is shown in Figure A.1. For local recovery Program, the single element failure model is used (See Figure 4.1, Chapter 4). That Program only generates one output file (capacity.dat). Each Program of the other schemes generates three output files: (capacity.dat), (delay.dat) and (messages.dat). In global recovery, a backup path is built between the ingress LER and each egress LER in each multicast tree. Segmentation-based recovery builds the backup paths between the branching point in a multicast tree (see Figure 4.5, Chapter 4).

A.1 Building the Routing Tables in Each Router.

1. Each one of the above Programs starts by reading the network topology from (tolopogy.dat) file generated by (Waxman Program).
2. Each router knows its neighbors routers. Then, each router starts building the routing table from the basic entries that represent the neighbors of that router.
3. A flag for each entry is set as (unsent entry) to indicate that entry is not sent yet to the neighbors routers.
4. for each router do the following
    - exchange the unsent entries in the routing table with all the neighbor routers.
    - If the received entry is already received before, ignore that entry.

Step 4 is repeated until all the entries of the routing table at each router are flagged as sent.



Node	Interface	Delay	Sent
2	0	2	0
3	1	1	0
5	2	3	0

Figure A.1: A network composed of five nodes.

Figure A.1 (a) shows a network composed of five nodes while Figure A.1 (b) illustrates the basic entries of the routing table at node 1. Initially, all the entries are flagged as not sent (0 in the table indicates that the corresponding entry has not sent yet to the neighbor routers).

A.2 Building the Multicast Tree

1. The root of the tree (the ingress LER in MPLS networks) and the set of egress LERs are read from a file (LERs.dat).
2. A shortest path between the ingress LER and each egress LER is calculated using the routing tables.

3. The result of that will be tree (T) as a shortest path tree.

4. In (Tree division Program), tree T is used as an input to the tree division algorithm (See Figure 5.3, Chapter 5) to produce the domains. The result of (Tree division Program) is a file called (domains.dat). Table A.1 explains the format that file.

Table A.1: Format of (domains.dat) file.

Tree root n LER1 LER2 ..... LERn
Domain1's root m1 BDR11 BDR12 .....BDR1m1
Domain2's root m2 BDR21 BDR22 .....BDR2m2

where,
n = number of egress LERs in tree T.
m1 = number of BDRs in domain 1.
m2 = number of BDRs in domain 2.