Dynamic Tree Switching for IP Networks

Constantinos Neophytou

A thesis submitted in partial fulfilment of the
requirements for the degree of

Doctorate of Philosophy

Department of Electronic Engineering
Queen Mary, University of London

2007

UNIVERSITY OF LONDON

ABSTRACT

Dynamic Tree Switching for IP Networks

Constantinos Neophytou

Multi-Protocol Label Switching (MPLS) provides a flexible forwarding mechanism for traffic engineering in data networks. A valuable way of further simplifying the forwarding process, particularly within IP networks is through the use of multipoint-to-point label switched paths (LSPs). The use of multipoint-to-point (MP2P) LSPs in this way was identified in even the earliest IETF drafts of MPLS. Whilst some previous work addressed the formulation of such multipoint-to-point (MP2P) LSPs, only a few schemes have been proposed and it is recognised that the formation of such LSPs is non trivial. Furthermore, resilience in such configurations has only been partially addressed. In this thesis, the notion of Dynamically Adaptive Multipoint to Point LSPs is presented whereby the MP2P LSPs are created and maintained dynamically. A new scheme (for the online creation and maintenance of MP2P LSPs) called Dynamically Adaptive Multipoint to Point (DAM) is presented. In DAM, MP2P LSPs are maintained by the exploitation of ad-hoc algorithms that can lead to fast localised healing. DAM, as part of its restoration strategy, supports a Directed Acyclic Graph (DAG). The multipath nature of this DAG has been used to formulate a load balancing strategy for use in supporting multiple overlaid MP2P LSPs.

Acknowledgements

I would like to especially thank my supervisor Dr Chris Phillips for his support and guidance throughout this work and express my gratitude for his patience. I would also like to thank my second supervisor Professor Laurie Cuthbert for his help and support, and my monitors Dr Eliane Bodanese and Dr John Bigham for their guidance and comments. I would also like to express my gratitude and thank the EPSRC for providing the funding for this work and to the technical support team at OPNET.

I take this opportunity to thank Dr R. N. Adams for giving me the opportunity to study at Queen Mary and Professor Jonathan Pitts for his helpful advice and support over the years.

I would also like to thank all my friends at Queen Mary especially Janny Huang, Jim Tan, Karen Shoop, Landong Zuo, Leonid Titkov, Ivan Djordjevic, Huifang Kong, Linda Rolfe, Phil Willson, Ho Huen and Konstantinos Smparounis.

Finally I would like to thank my family and friends for their patience and support throughout my studies especially my mother.

This thesis is dedicated to my family, Stella, Irene, Anthony, John and my Mother Panayiota

in Loving Memory of my Father Neophytos Neophytou

(17/7/1937 - 14/10/1990)

IXNK

Contents

List of Figures

List of Tables

Glossary of Terms

AS	Autonomous System
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CBT	Centre Based Tree
CE	Customer Edge
CoS	Class of Service
CSPF	Constrained Shortest Path First
CR-LDP	Constrained based routing LDP
DAG	Directed Acyclic Graph
DAM	Dynamically Adaptive Multipoint to Point
DVMRP	Distance Vector Multicast Routing Protocol
EGP	Exterior Gateway Protocol
FEC	Forwarding Equivalence Class
FR	Frame Relay
FTN	FEC-to-NHLFE
GB	Gafni – Bertsekas algorithm
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
ILM	Incoming Label Map
LAN	Local Area Network
LDP	Label Distribution Protocol
LER	Label Edge Router
LRR	Link Reversal Routing
LSA	Link State Advertisements
LSD	Link State Database
LSP	Label Switched Path
LSR	Label Switched Router
MOSPF	Multicast Extensions to OSPF
MPLS	Multi-Protocol Label Switching
MP2P	multipoint-to-point
NHLFE	Next Hop Label Forwarding Entry
OSPF	Open Shortest Path First

PE	Provider Edge
PIM	Protocol Independent Multicast
P2MP	point-to-multipoint
P2P	point-to-point
RIP	Routing Information Protocol
RPF	Reverse Path Forwarding
RSVP	Resource reSerVation Protocol
SPF	Shortest Path First
TCP	Transmission Control Protocol
TE	Traffic Engineering
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VPWS	Virtual Private Wire Service
WAN	Wide Area Network
1 + 1	One-plus-one protection
1 : 1	One-for-one protection

Chapter 1

1 Introduction

An ISP or the management authority of an AS seeks to obtain efficient utilisation of resources. Traffic engineering has a central role and the success of this largely depends on the tools used as these can often impose constraints on the manner in which they can be used. Multi-Protocol Label Switching (MPLS) provides a mechanism for fast layer-2 forwarding together with flexible path selection, minimising the need for the cumbersome longest prefix match used in traditional IP routing. Virtual connections are pre-established and are realised in the form of label switched paths (LSPs). However, by using this connection-oriented approach, the resilience and flexibility provided by the connectionless forwarding of traditional hop-by-hop IP routing is, to some extent, lost. Conventional use of MPLS requires the use of point-to-point (P2P) LSPs and thus the maintenance of individual connections throughout the network. The label swapping capability of MPLS facilitates this by allowing for labels to be bound to each leg of the path making up the complete point-to-point LSP. Particularly for large networks, where the greatest benefits of layer-2 forwarding can be derived, such an arrangement is difficult to maintain and does not scale well without aggregation through label stacking. What is required is a means whereby the benefits of MPLS can be brought to the core of the network whilst still maintaining qualities normally associated with traditional IP routing. Identified in even the earliest drafts of the IETF's MPLS architecture [RFC3031] is the concept of multipoint-to-point (MP2P) LSPs.

A multipoint-to-point LSP can be considered as an inverted tree routed at the destination whereby all sources are connected to a particular destination via a single MP2P tree. Such a configuration replaces each point-to-point LSP connection with a single LSP for each destination and this is believed to be key in achieving the goal of fast forwarding coupled with resilience and flexibility. The path to all destinations in a network is realised as an overlay of these MP2P trees as opposed to a mesh of P2P LSPs. Naturally a destination can have more than one tree associated with it as service classification and path constraints dictate, providing for an efficient method of implementing different classes of service.

Although MP2P LSPs may seem to be an elegant way to provide a more scalable approach within MPLS networks the key problem, as identified by Saito *et al* [SAI00], is the difficulty in creating them. As a consequence of this only a small number of schemes have been proposed and the issue of resilience in this research area has only partially been addressed. Bhatnagar *et al* [BHA03] present simulation results showing that their scheme for MP2P LSP creation can reduce label usage by between 60% and 70% depending on the number of P2P connections being replaced. The use of MP2P LSPs as opposed to P2P LSPs provides much more than just the ability to reduce label space; the effective management of MP2P LSPs can provide other benefits.

In this thesis an improved method for the creation MP2P LSPs is presented. Furthermore, the scheme addresses the issue of resilience in such configurations by adopting a highly adaptive algorithm to make the created MP2P LSPs robust in an attempt to bring to layer-2 forwarding what is taken for granted with traditional IP routing. This new scheme and its associated protocol is called **D**ynamically **A**daptive **M**ultipoint to Point (**DAM**).

1.1 Motivation

An MPLS domain can become difficult to maintain or unmanageable. This situation is increasingly likely to arise as the MPLS network grows. A particularly large network presents the following problems: (1) *individual connections must be maintained for every ingress – egress pair, and (2), the failure of a single link or node (LSR) may invoke the restoration of multiple connections.*

MP2P LSPs can provide a solution to these problems by:

A: significantly reducing the number of LSPs required to provide connectivity across the MPLS domain (a single MP2P LSP is required for each egress node) and…
B: reducing the number of LSPs having to be restored as a result of failure.

The use of MP2P LSPs as a solution to these problems can provide the following benefits:

- The required label space is reduced
- An elegant way to conduct traffic engineering

- The nature of the MP2P trees provides a model readily expandable for support of QoS and CoS as any egress node can have more than one MP2P LSP associated with it
- ISPs are provided with a simpler management option because of the relative scalability of the MP2P approach
- ISPs can provide support for VPNs using MP2P LSPs

To-date four schemes (not including DAM) have been proposed for the formation of MP2P LSPs. Saito *et al* [SAI00] propose a scheme whereby routes are initially selected between each ingress/egress node pair and then the MP2P LSP is designed to include the selected the routes. The actual design or creation of this MP2P LSP is a consequence of 0–1 integer programming and this is formulated for each individual egress node. Failure and recovery is addressed in this scheme by means of a complete MP2P LSP acting as a backup to the working MP2P LSP such that, should a route between an ingress/egress pair be severed, the backup MP2P LSP is chosen by the ingress as it is already set up. Bhatnagar *et al* [BHA02] state that this scheme cannot be used frequently for large MPLS domains as integer programming is, in general, NP–hard and propose another scheme for the formulation of MP2P LSPs. The scheme proposed by Bhatnagar *et al* [BHA02] is similar to that proposed by Saito *et al* [SAI00] in that both use P2P LSPs as a basis in for the formation of MP2P LSPs. Bhatnagar *et al* [BHA02] describe their scheme as a simple polynomial time heuristic based algorithm and have shown that a benefit of using MP2P LSPs over P2P LSPs is a reduction in the required label space. Their scheme is a merging algorithm that takes as its input P2P LSPs and merges them to create an MP2P LSP routed at the egress. It does so by merging the LSPs at a contiguous set of common nodes starting at the egress whilst providing constraints on merging for loop prevention. The scheme presented by Applegate *et al* [APP03] adopts a linear programming calculation in order to formulate the trees and reduce the label space. The calculation made by Applegate *et al* [APP03] is conducted using a commercial linear programming software package CPLEX [ILOG1]; arriving at solutions is typically very slow and therefore the authors suggest that their approach be used on a daily basis. Gonzlez-Valenzuela *et al* [GON02] have proposed another method for the creation of MP2P trees. It uses a colony of mobile agents to search out and discover the MP2P tree. The use of mobile agents in this field relative to existing technologies is still in its infancy and the overhead in terms of processing and signalling for what is a comparatively low level task is questionable. Whilst both of the schemes proposed by Saito *et al* [SAI00] and Bhatnagar *et al* [BHA02] have their merits in the creation of MP2P LSPs, both rely on the existence of an

appropriate selection of P2P LSPs that can be subsequently merged. None of the schemes has fully addressed the issue of resilience in MP2P LSPs.

Clearly a scheme that is both dynamic in the sense of creation of MP2P LSPs with a built-in resilience tactic would fill the void in this area of research and development. The DAM scheme and protocol presented in this thesis aims to fill this void and differs from the aforementioned schemes in that, rather than proposing a new algorithm for the creation of MP2P trees, it leverages existing ones adapting them where necessary. In addition to this, the DAM scheme provides for localised healing as a method of resilience in the MP2P tree. This thesis develops DAM further by addressing the issue of load balancing multiple overlaid trees.

1.2 Novel Contribution

The schemes proposed by Saito *et al* [SAI00], Applegate *et al* [APP03], and Bhatnagar *et al* [BHA03] calculate the MP2P tree offline, none of these schemes is dynamic with little or no provision for resilience. Saito *et al* [SAI00] address resilience through the use of complete back up trees this method relies on the appropriate use of alarm signals and the propagation of signalling back to the ingress in order to switch from the failed tree to the back up tree. What is clearly needed is a scheme that is both dynamic in its creation of MP2P LSPs and resilient in the face of network failure. It is this need that is the driver for the work described in this thesis with the added motivation that service providers would benefit from such a scheme particularly those providing VPN services as an alternative to leased lines an area where there has been much commercial interest of late. The major contributions of this work are:

- A novel scheme for the online creation and maintenance of MP2P LSPs called Dynamically Adaptive Multipoint to Point (DAM) [NEO04]. This protocol is the first of its type and goes further than previous work by directly addressing the issue of resilience by providing a strategy for successful localised healing.
- A restoration strategy that alleviates the need for the reliance on higher layer routing protocols to re-converge in order for rerouting to take place.
- A novel extension to DAM called DAM with Load Balancing (DAM-LB) for performing load balancing between multiple overlaid MP2P LSPs. This scheme is believed to be unique and the first of its kind.
- An alternative restoration strategy for DAM called Partial Pre-Planning.
- Simulation models that have been implemented in the context of the other principle protocols required for fully operational network nodes.

Chapter 2

2 Research Landscape

This section gives an overview of the relevant technologies that impact on the author's work. It briefly introduces the principle protocols and technologies referred to throughout this text. It then goes on to summarize efforts made in this field prior to this work.

2.1 Routing / Switching

Conventional routing within an IP network can be separated into to two tasks, the determination of paths through a network and the processing of individual datagrams as they traverse a network to get to their destination. The determination of paths normally involves the maintenance of routing tables at routers throughout a network (with the exception of source routing where paths are determined at the edge of the network). This task is the responsibility of routing protocols that are categorised according to their scope, that is to say whether they function within a domain or between domains. These categories are more formally known as Interior Gateway Protocols (IGP) those that function within a domain (Autonomous System – AS) and Exterior Gateway protocols (EGP) that function between domains. Broadly speaking these routing protocols are also classified according to the class of algorithm they use to maintain the routing tables; these are usually either Distance Vector or Link State.

Distance Vector algorithms (also known as Bellman–Ford [COR01]) maintain a distance at each router to every known destination in the network. Each router is required to send periodically all or part of its routing table to its neighbours. The main advantage of using this type of algorithm is that it is simple to implement [IBM01], however there are some disadvantages in that after a failure the routing tables at all nodes may take a long time to re - converge. To counter this in practical implementations the maximum number of hops for a given route is limited. Routing Information Protocol (RIP) [RFC2453] is an example of a protocol that implements distance vector routing. The maximum distance supported by RIP is 15 hops; a cost of 16 signifies infinity, meaning a particular node is unreachable. This makes RIP suitable only for small networks.

Conversely, link-state algorithms (also known as shortest path first) flood routing information to all nodes in the network [COM00]. Each router sends link state advertisements (LSA) that essentially describe the state of the router's own links (the cost to all its directly connected neighbours). In link-state algorithms, each router maintains a (LSD) Link State Database made up of the LSAs it has received from other routers leading to a topological picture of the entire network. In a converged network, all routers should contain effectively the same information within their local LSD. Unlike distance vector protocols, link-state protocols typically only flood updates when there is a change of state in the LSD. This implies that they are "quieter" protocols. In addition, the fact that each router has a full topological view allows it to locally and rapidly determine an appropriate next-hop to a given destination in response to link failures and so forth.

Open Shortest Path First (OSPF) is a link state routing protocol that has been designed for use as an IGP. OSPF has been specified by the IETF in [RFC2328]. In order to work more effectively as the size of a network grows, OSPF divides the AS into areas. All areas are connected to each other via a main area called the backbone. This limits the amount of control traffic flowing around the network by allowing nodes on the edge of an area (i.e. area border routers – ABR) to summarise the topology of their area and relay this information to other areas via the backbone area. The exchange of link state advertisements leads eventually to converged LSDs; that is to say all nodes have an identical picture of the network in their Link State Databases. Subsequently routing tables can be calculated using a shortest path first algorithm such as Dijsktra's algorithm [DIJ59]. OSPF is widely used as an IGP, [IBM01] in fact [RFC1812] – Requirements for IPv4 Routers, lists OSPF as the only required dynamic routing protocol.

Border Gateway Protocol (BGP) is a special type of protocol in that it is neither a distance vector nor a link state protocol. BGP is a path vector protocol; this type of protocol is one that associates the path to a destination with that destination in information it exchanges with other routers. This is done to overcome the problem that there is no single routing metric in use between different ASs, something particularly relevant, as BGP is the protocol principally used as an EGP. The latest version, BGP-4, has been specified by the IETF in [RFC1771][RFC4271]. BGP speaking routers start sessions with peers (other BGP speakers) using Transmission Control Protocol (TCP) [RFC793] connections, when two speakers initially commence a BGP session they exchange their entire routing table. Once the entire table has been exchanged, changes to the table are communicated as incremental updates.

Traditional IP routing involves a forwarding decision being made at each intermediate node. This forwarding decision is made using a longest prefix match, whereby the destination IP address in the IP header is examined and the most specific match is made with the routing table entries that have been created using a routing protocol such as OSPF or BGP. The IP datagram is then forwarded to the next hop based on this match where this process is carried out again until the final destination is reached. This forwarding process takes place at each intermediate hop and the check / decision is made for each individual packet on a hop-by-hop basis. Although this method fits the best effort service model of IP well in that it is completely connectionless and packets may take different routes to the same destination, it can lead to inefficient processing of packets as each packet is dealt with on an individual basis. Perhaps more important are the commercial implications of such a model.

Service providers require an effective way of implementing a multiple Class of Service (CoS) model and the single service model that IP provides is clearly not adequate. There have been a number of efforts by the IETF namely IntServ and DiffServ [RFC2475] to move from the single service model of IP by defining different classes of service. Even so, the connectionless nature of IP does not lend itself to this. Closely linked to providing QoS within a multi-service environment is the capacity to perform Traffic Engineering (TE). Network bandwidth is a finite resource that needs to be managed correctly; under-utilized links may represent economic loss. TE is the ability to place traffic where there is available capacity and as a result provides a way of improving resource utilisation within a network [RFC3272].

One of the problems faced by service providers with conventional routing is the processing overhead. In order to achieve high–speed data transfer Layer-2 technologies such as Asynchronous Transfer Mode (ATM) and Frame Relay (FR) [RAH91] are employed to accomplish this using simple label swapping techniques. ATM presents a connection-orientated approach that is able to switch data at high speed using simple switching equipment and high speed Virtual Paths (VP). Although many service providers employ technologies such as FR and ATM as part of their infrastructure, they are not as flexible as they might be and furthermore the integration of IP and ATM is complex because of their converse nature (connectionless versus connection–orientated). A solution that allows Layer-2 and Layer-3 to fit better is Multi-Protocol Label Switching (MPLS). This provides what is widely known as a "shim layer" (Layer-2 ½). MPLS goes part way to providing the speed and simplicity of Layer-2 along with the flexibility of Layer-3.

2.2 Multi-Protocol Label Switching (MPLS)

MPLS is a standard that has been developed within the confines of the Internet Engineering Task Force (IETF) and has been specified by RFC 3031 and other related RFCs. MPLS can be considered to be an Open System Interconnection (OSI) Layer-2½ technology that remains independent of Layer-2 and Layer-3 protocols. MPLS provides a framework for fast forwarding based on a short fixed length label, by doing so it is able to decouple route selection from the forwarding mechanism. This leads to a more flexible forwarding than traditional IP routing by allowing for paths to be set up that do not necessarily follow the least cost path. Unlike IP forwarding, a packet is assigned to a Forwarding Equivalence Class (FEC) only once at its entry point to the MPLS domain (A FEC is a group of IP packets that are forwarded in the same manner). The assignment of a packet to a particular FEC can be based on any chosen criteria, providing much of the flexibility and advantages that MPLS has to offer over traditional IP routing.

Since a packet is assigned to a FEC when it enters the network, the ingress router can make that assignment using any information it has about that packet. This information, unlike conventional forwarding, does not have to come solely from the packet header. Other parameters can be considered. These may include factors like the ingress router or even the particular interface that a packet arrived on or indeed the egress router it is destined for within the current domain. As a result a packet that enters the network at a particular router can be labelled differently than if it were to enter the same network at a different router. This allows forwarding decisions to be made based on a packet's entry point to a network, something that cannot be done with conventional forwarding, as the details of a packet's ingress point do not travel with the packet (unless it is source routed in which case its path has already been mapped).

Traffic engineering considerations may require the path of a packet to follow a particular route that is explicitly chosen before the packet enters the network. In conventional forwarding, each packet that is source routed is required to carry with it the details of its route. Where packets are explicitly source routed, details of the complete route are carried; in loose source routing this route is only partially specified. This can lead to large overheads that can be overcome with MPLS. In MPLS the details of a particular route need only be carried once during path set up, subsequent packets that must follow the same route can use a label that represents that route.

Labels can also be used to reflect a particular FEC's CoS. This can be used to implement a multiple CoS model tailored to a service providers particular needs.

2.2.1 Label Switching

An MPLS enabled router is referred to as a Label Switched Router (LSR), an LSR that resides on the edge of an MPLS domain is called a Label Edge Router (LER) (an LER can be an egress or ingress LSR). A Label Switched Path (LSP) is a high-speed path that traverses an MPLS domain normally beginning at an ingress LSR and culminating at an egress LSR via any number of intermediate LSRs within the domain.

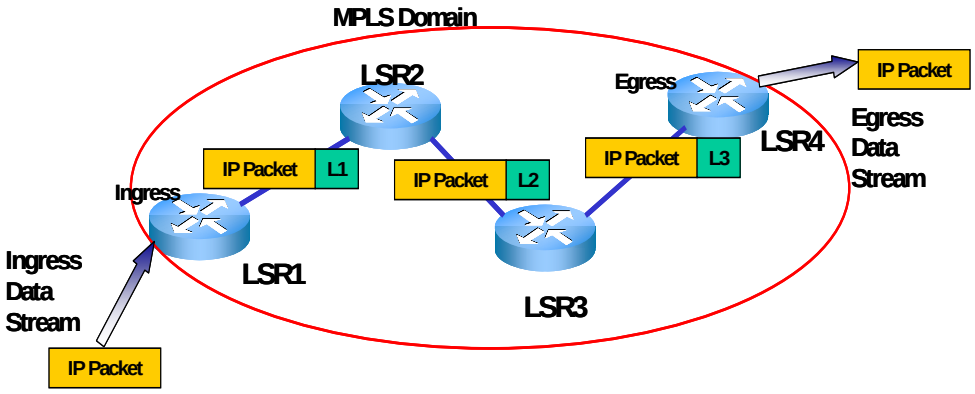


Figure 1:- Label switching in an MPLS domain

Figure 1 shows label switching through an MPLS domain. When the IP packet arrives at the ingress LSR (LSR1) the packet is assigned to a FEC and a label is added to it (L1). LSR1 then forwards the packet to the next hop in the Label Switched Path (LSP). Intermediate hops (LSR2 & LSR3) along the LSP carry out a simple look up and label swap before forwarding the packet on to the next hop. When the packet arrives at the egress LSR (LSR4) its label is removed before it is forwarded using conventional routing. As shown in figure 1, labels along the LSP are only locally significant, so there is no need for a global allocation scheme. In MPLS labels can be assigned on a link-by-link basis enabling the easy merging of paths and allows for localised healing. The merging of paths is no so straightforward in ATM as paths are always set up in an end-to-end manner [CAL97].

In order for an LSR to be able to perform this label switching function a number of data structures have been defined to allow this to take place.

These data structures include the Next Hop Label Forwarding Entry (NHLFE), the Incoming Label Map (ILM) and the FEC-to-NHLFE (FTN).

An LSR maintains a "**Next Hop Label Forwarding Entry**" (NHLFE) [RFC3031]. The NHLFE is used when forwarding a labelled packet. It contains the following information:

1. The packet's next hop

2. The operation to perform on the packet's label stack; this is one of the following operations:
    a) Swap the label at the top of the label stack with a specified new label

    b) Pop the label stack

    c) Replace the label at the top of the label stack with a specified new label, and then push one or more specified new labels onto the label stack.

It may also contain:

d) The data link encapsulation to use when transmitting the packet

e) The way to encode the label stack when transmitting the packet

f) Any other information needed in order to properly dispose of the packet.

The "**Incoming Label Map**" (ILM) [RFC3031] maps each incoming label to a set of NHLFEs. It is used when forwarding packets that arrive as labelled packets. If the ILM maps a particular label to a set of NHLFEs that contains more than one element, exactly one element of the set must be chosen before the packet is forwarded.

The "**FEC-to-NHLFE**" (FTN) [RFC3031] maps each FEC to a set of NHLFEs. It is used when forwarding packets that arrive unlabeled, but which are to be labelled before being forwarded.

When forwarding a labelled packet, an LSR examines the label at the top of the label stack. It then uses the ILM to map this label to an NHLFE. Having done this it uses the information in the NHLFE in order to determine where to forward the packet, as well as performing any operation on the packet's label stack specified in the NHLFE. The LSR then encodes the new label stack into the packet, and forwards the encapsulated packet to the next hop.

When forwarding a packet that has arrived unlabeled, an LSR analyses the network layer header and assigns the packet to a FEC. It then uses the FTN to map this FEC to an NHLFE. Having done this it uses the information in the NHLFE in order to determine where to forward the packet, as well as performing any operation on the packet's label stack specified in the NHLFE (Popping the label stack would not make sense and in any case would be illegal in this case). The LSR then encodes the new label stack into the packet, and forwards the encapsulated packet to the next hop.

2.2.2 Label Distribution

Label distribution is a necessity in MPLS networks as it ensures that adjacent LSRs share a common view of FEC to Label bindings. In order for this to occur a **Label Distribution Protocol** (LDP) [RFC3036] is required. An LDP is a set of procedures by which one LSR informs an adjacent LSR of the FEC to Label bindings it has made. Two LSRs that use an LDP to exchange FEC to Label binding information are known as "label distribution peers" with respect to the binding information they exchange.

Label distribution peers may exchange binding information in one of two methods prescribed in [RFC 3031]; these are "downstream unsolicited" and "downstream on demand" as shown below (Please note other methods are possible but are not discussed here).

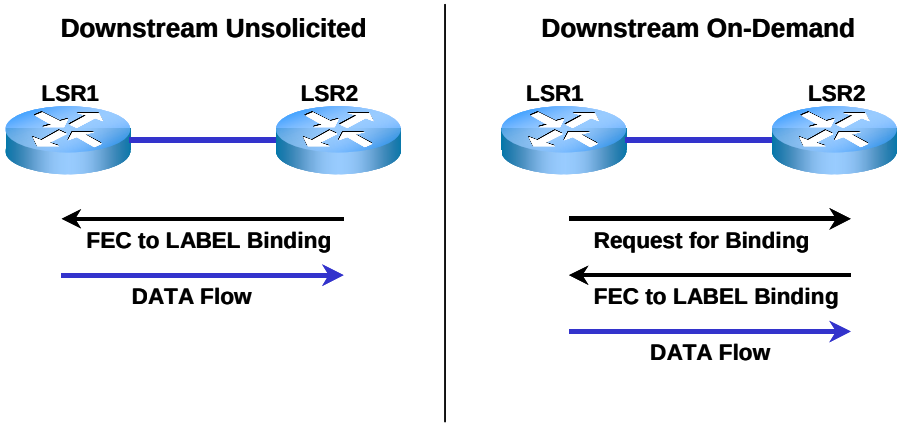


Figure 2:- Label Distribution Methods

Figure 2 shows the label distribution methods that have been defined in [RFC3031]. In the **downstream-unsolicited** method the downstream LSR (LSR2) communicates its FEC/label binding to LSR1 (which is upstream of LSR2 with respect to data flow). If LSR1 receives traffic belonging to that particular FEC it can use the label knowing it will be understood, allowing data to flow downstream to LSR2. In the **downstream on-demand** method LSR1 identifies LSR2 as the next hop for a particular FEC and requests a binding for that FEC. If LSR2 recognizes the FEC and has a next hop for it, it creates a binding and relays this binding to LSR1. LSR1 can now use the label received knowing it will be understood.

There are a number of protocols that are used as an LDP to fulfil the necessity of label distribution. These include Vanilla LDP, CR-LDP [RFC3212] [RFC3213] and RSVP-TE [RFC3209] [RFC4420], though other proprietary and non-proprietary solutions may exist as RFC 3031 places no restrictions on the LDP used. Label distribution can take place by piggybacking binding information on an existing routing protocol, or through the creation of a dedicated protocol to disseminate the bindings.

Vanilla LDP allows LSPs to be set up that follow the shortest path. This is done by using existing IP routing tables to forward its control messages. CR-LDP and RSVP-TE on the other hand allow the management authority of an MPLS domain to set up LSPs that follow explicit routes that are not necessarily the shortest path. Constrained based routing LDP (CR-LDP) adds extensions to Vanilla LDP so that explicitly routed LSPs or LSPs following other constraints can be set up. Vanilla LDP and CR-LDP use the same mechanisms and messages for peer discovery, session establishment/maintenance, label distribution/management and error handling and can coexist in the same MPLS domain.

Resource reSerVation Protocol (RSVP) with Traffic Engineering extensions (RSVP-TE) like CR-LDP is used to set up LSPs that can be made to follow paths that do not follow the SPF route. LSPs set up using RSVP-TE are of a "soft-state" nature; this is because of the workings of RSVP. RSVP resource reservations are cancelled if RSVP does not send refresh messages along the path of an existing reservation, because of this manner in which maintenance occurs resource reservations in routers are deemed to be "soft-state". Figure 3 illustrates this soft-state nature of RSVP-TE in comparison with Vanilla LDP or CR-LDP.

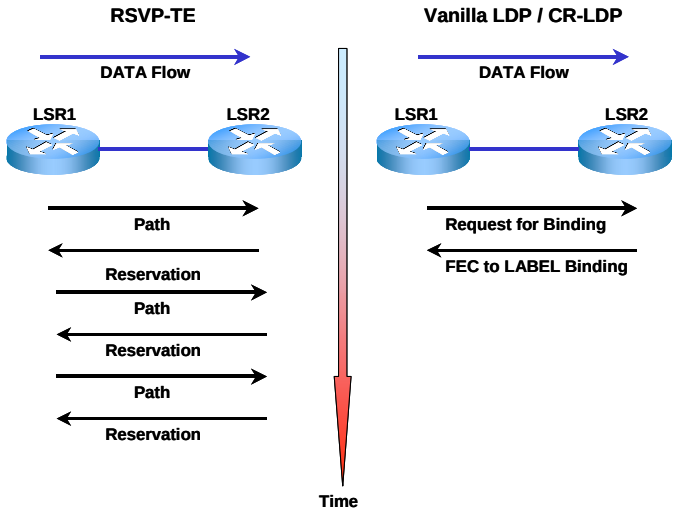


Figure 3:- RSVP-TE compared with Vanilla LDP /CR-LDP

The LSP on the left of figure 3 is continually refreshed unlike the LSP on the right that is set up using Vanilla or CR –LDP.

2.2.3 Aggregation

The MPLS framework allows aggregation to take place with respect to FECs. This can be leveraged to aid in traffic engineering and is indeed a powerful feature of the MPLS architecture. The process of aggregation involves the binding of a single label to a FEC that is comprised of a union of FECs (within the same domain), and subsequently applying that label to all traffic in that union. As a consequence aggregation can lead to a reduction in the number of labels that are required to handle a particular set of packets. Given a set of FECs that can be

aggregated into a single FEC the extent to which this union can occur is variable leading to different levels of granularity. Factors that can affect this level of granularity may include the FEC's CoS along with the egress node that it is destined for within that particular domain as an example.

2.2.4 Summary

This section has given an overview of MPLS, including what it has to offer over conventional routing along with a description of the basic features of its architecture that are related to this research. MPLS is a Layer 2 ½ technology defined by the IETF that facilitates high-speed switching. It is a framework that decouples route selection from the forwarding mechanism and by doing so allows those employing it to tailor their own traffic engineering solutions. Much of the flexibility afforded by MPLS is owed to the manner in which packets can be assigned to a FEC. Unlike conventional routing, this assignment is not based solely on the contents of the packet header but can be done using other information such as the packets entry or exit point to the MPLS domain. Source routed signalling packets forwarded in an MPLS domain need only carry information about the route they must follow once during LSP set up. For the subsequent data, forwarding is based on the "virtual circuit" labels associated with a given pre-established path. Labels are only locally significant so there is no need to have a global addressing scheme. The MPLS architecture allows aggregation to take place with respect to FECs. This is a powerful feature that can be leveraged to aid traffic engineering. In order to set up an LSP an LDP is required to distribute labels, the resulting LSP can have a soft-state nature if the LDP used to do this is continually refreshing the path as in RSVP-TE, or hard-state where refreshing is unnecessary, as with CR-LDP. The challenges in MPLS are in providing TE solutions that utilise the rich feature set that MPLS has to offer because as a technology it provides a flexible framework with which to facilitate fast forwarding coupled with effective network management. As a result of these benefits that it has to offer, MPLS is gaining in popularity and deployment amongst service providers [CUR04].

2.3 MPLS Resilience

This section describes the mechanisms currently being used to counteract resource failure in MPLS networks; this may include the failure of a single link or a multitude of links as is the case when a node fails. Recovery strategies are, in general, categorised according to the scope of the reaction they employ. This reaction is normally described as being either a local repair or an end-to-end restoration; end-to-end restoration is sometimes referred to as path restoration. These two types of reaction are distinct, as suggested by their names. In local repair, traffic flow is only disturbed near the point of failure. The path is diverted around the failure to re-establish connectivity between the end points of the complete path. In end-to-end restoration, connectivity is re-established not by the diversion of path flow around a failure but instead through the use of an entire alternative route between path end points that avoids the point of failure.

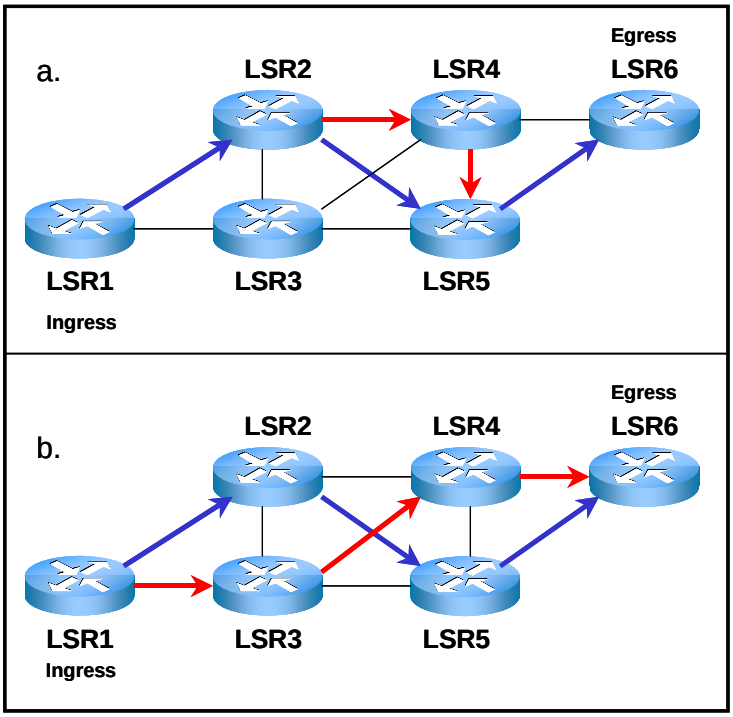


Figure 4:- Recovery Strategies - a) local repair, b) end-to-end restoration

In figure 4, a link failure occurs between LSR2 and LSR5, the working path is shown in blue and the backup path is shown in red. In the local repair scenario shown in figure 4a the path segment flowing from LSR2 to LSR5 has a local backup that flows from LSR2 to LSR4 to

LSR5. In figure 4b the backup path is a complete alternative path that goes via LSR3 and LSR4 to provide connectivity using end-to-end restoration. In figure 4b the working path and the backup path are both node and link disjoint, if this backup path was only intended to protect against link failure between LSR2 and LSR5 a backup path flowing through LSR3 and LSR5 would also be a valid choice.

These reactions, localised or end-to-end, are further characterised by the manner that the diverted or alternative path is constructed. This can be classified as pre-emptive or reactive with respect to the failure it is intended to counteract. A pre-emptive policy is one where the backup path is pre-planned and used in the event of a failure of the working path; this is often referred to as protection switching. Protection switching is typically associated with end-to-end restoration whereby complete backup paths are set up and ready to be used. However a number of schemes exist that offer pre-planned backup to only a section of the complete path; this is known as segment protection. In the reactive approach backup paths are calculated and used only as a result of failure. This is called rerouting.

2.3.1 Protection Switching

In protection switching, data is switched from the failed working path onto the backup path. For MPLS networks, this conventionally takes place [HAR04] at the ingress LSR which is regarded as the point of repair. It uses pre-established backup paths that are created to protect a complete path or a path segment [RFC3469]. The resources consumed by this backup path may be set aside exclusively for the purposes of protection or used to carry other traffic until the point they are used for the recovery of the path being protected [HUA02]. This leads to two subtypes of protection [RFC3469] "1 + 1" and "1 : 1". In 1 + 1 (one plus one) protection the resources on the backup path are fully reserved. In fact the backup path carries the same traffic as the working path. Selection between the working path and the recovery path is made by the path merge LSR (PML)[RFC3469]. The PML is defined as the point where the working path and the backup path merge. This can be the point of repair and in many set ups this is also the egress LSR as a complete path is usually protected in this way. In 1 : 1 (one for one) protection there is normally no simultaneous transmission of traffic on both the working path and backup path, resources allocated on the backup path for protecting the working path can be used by other traffic. This other traffic is usually lower priority traffic with limited service guarantees if at all; moreover such traffic may not even be MPLS traffic but simply IP traffic [HAR04]. Even

though the backup path is pre-signalled and resources are reserved they are not used until a failure occurs, allowing lower priority traffic to utilise these resources, leading to a less costly form of protection switching. Another approach is to allow a single backup path to protect multiple working paths. This is considered for segment protection by the IETF in their specification for "Fast Reroute" in [RFC4090]. In [RFC4090] bypass tunnels are defined, these are effectively pre-planned path segments used to protect a set of LSPs. [RFC4090] also describes "one to one" protection whereby path segments are protected using exclusive backup paths. However this has clear scalability issues; the authors state, "To protect an LSP that traverses N nodes fully, there could be as many as (N - 1) detours". This standard [RFC4090], as well as defining these types of segment protection, specifies extensions to RSVP to support local protection. One of the motives for segment protection is that with end-to-end protection notification of the failure must propagate back to the ingress. This can adversely impact on the recovery time. It follows that since path segments are shorter entities than a complete path, the signalling adjacent to the failure to the point of repair is shorter therefore segment protection or local repair is much faster than end-to-end protection [KOD01][MAR03]. There is an obvious trade off between end-to-end protection and segment protection, in an attempt to shorten recovery times there may be additional costs in terms of the total resources used for protection. To address this consideration Qiao *et al* [QIA03] have formulated a method using integer linear programming to provide shared segment protection. This method seeks to determine an optimum set of segments to protect a given path. The focus here is in improving restoration times for existing protection schemes through the use of optimal segment sets. The authors [QIA03] state that the calculation of these segments sets for large networks is time consuming; however their method works well for networks with less than 100 nodes.

In addition to end-to-end and segment protection switching there exists a form of protection switching that belongs in neither of these two categories. It is global in that an end-to-end backup path exists and local in terms of the point of repair being immediately upstream from the failure. This type of protection involves the use of a backward or reverse LSP from the point of failure to the ingress and, in addition to this, an alternative or backup LSP from the ingress to the egress [HUN02][MAR03]. When there is a failure, the LSR upstream of that failure becomes the point of repair and switches data transmission to the pre-established backward LSP, the arrival of this data at the ingress signals the failure to it. This traffic and all subsequent traffic is sent via the backup LSP from the ingress. This model ties up more resources and presents similar problems to that of conventional end-to-end protection switching [MAR03a]

because a reversed packet provides failure notification. This packet must travel back to the ingress point.

2.3.2 Rerouting

The use of rerouting to perform the task of recovery involves the establishment of new paths or path segments in response to a failure. This is a reactive process and such paths are established after the failure has been notified. Signalling takes place after failure notification and the new path or path segment is set up. Huang et al [HUA02] state that rerouting is inherently slower than protection switching because more must be done after the failure takes place. This is not necessarily true for all cases as the actions involved in executing a recovery are not the same with respect to the time taken for their execution. The major drawback of rerouting is that in its simplest form it relies on the re-convergence of higher layer IP routing protocols [HAR04], the time delay associated with this can range from tens of seconds to minutes in the case of BGP. This delay is clearly too slow when recovery times in the order of tens of milliseconds to hundreds of milliseconds are desired. Rerouting has much to offer if this challenge can be overcome as reroute mechanisms tend to be simpler and much more economical [HUA02] this is because no resources are committed prior to failure except for the bandwidth that may be consumed for signalling. In terms of MPLS recovery, very little research exists with respect to rerouting; the majority of literature on MPLS recovery focuses on protection switching in one form or another. This may be a symptom of the false perception that rerouting relies on routing protocol convergence. Ahn *et al* [AHN02] propose a scheme for MPLS recovery using rerouting is presented. This scheme uses a number of least cost path calculations to perform local recovery. The method presented works by “trial and error” in that it may take more than one attempt for this scheme to successfully perform a local reroute around a failure. This has probably been done to overcome the challenge of waiting for a routing protocol to re-converge. Ahn *et al* [AHN02] consider each LSR on the working path that is downstream of the failure to be a candidate PML, in other words each candidate PML is a point that the local detour around the failure can rejoin the working path. In this scheme the node upstream of the failure conducts a least cost path calculation from itself to each of the candidate PMLs. The local detour around the failure is signalled to the candidate PML with the lowest cost. A mapping message is then sent back from this PML to the upstream node that signalled the local detour. If this is successful then rerouting around the failure can take place. In the event that this process is

unsuccessful the next candidate PML with the lowest cost is signalled and the process is repeated until a route around the failure is found and restoration has taken place.

2.3.3 Summary

This section has presented the principles associated with recovery in MPLS networks. At present there is a focus on protection switching rather than rerouting to meet this end. This is reflected by the IETF's current efforts in this area [RFC3469][RFC4090]. Protection switching offers pre-established backup paths. These may be used to protect complete paths or path segments. The backup paths in protection switching may also be used to protect single or multiple LSPs. The major issues of protection switching are that it can be costly to the service provider in terms of resources and that signalling the failure to the point of repair can be slow. In order to mitigate the impact of cost on resources some schemes use the backup path to protect multiple LSPs. Allowing lower priority traffic to use the resources taken up by the backup path during periods when there is no disruption to the network further reduces the cost impact. It is acknowledged that signalling the failure to the point of repair can be slow. This is more apparent with end-to-end protection switching, as failure notification must propagate back to the ingress. To address this, schemes have been devised to bring the point of repair closer to the point of failure. Segment protection addresses this directly by dividing a path into segments offering local repair for each segment. Another method for bringing the point of repair closer to the failure and mitigating signalling delays is through the use of a backward LSP coupled with an end-to-end backup. This, however, is costly in terms of resources and the benefits of this over end-to-end protection switching are negligible as both schemes share the same drawbacks. Most MPLS recovery schemes are centred on some form of protection switching. There is very little mention of rerouting in publications within this subject area. Rerouting is less costly in terms of resources as none are committed prior to the failure. Nevertheless, recovery times are a major consideration for restoration strategies and present a major challenge for rerouting. This is because in its simplest form rerouting relies on the re-convergence of IP routing protocols. This takes considerable time and is a factor that makes rerouting unattractive despite its benefits in terms of cost and simplicity. Ahn *et al* [AHN02] present a scheme that attempts to address this challenge but is based on multiple least cost path calculations and “trial and error” rerouting around the failure that may lead to more signalling than compared to protection switching.

2.4 Virtual Private Networks (VPN)

A Virtual Private Network is a network perceived to be private, constructed across a public or shared network infrastructure such as the public Internet. A more formal definition is given by Ferguson *et al* [FER98] "A VPN is a communications environment in which access is controlled to permit peer connections only within a defined community of interest, and is constructed through some form of partitioning of a common underlying communications medium, where this underlying communications medium provides services to the network on a non-exclusive basis".

In the past if an organisation wished to set up a private WAN they would do so by employing dedicated leased lines between their different sites. Service providers would provide such leased lines by exploiting protocols such as ATM or FR. However basic economics dictate a trend to move from WANs using leased lines to VPNs. Leased lines are both an expensive and restrictive means to connect two sites. Leased lines are usually of fixed bandwidth making it difficult if not impossible to provide extra bandwidth dynamically to cater for short-term peaks in demand. VPNs are more cost effective than private WANs because they allow a service provider to share a common backbone between customers, by doing so service providers can provide a more flexible service sharing resources like capacity and redundancy. Apart from the economies of scale experienced by the service provider, the customer also gains [BRI00][CIS04] by not having to be responsible for the complex task of planning, provisioning and managing a geographically distributed network as this is outsourced to the service provider.

As a general rule, VPNs are categorised according to the OSI layer at which they provide the partitioning element in the public or shared infrastructure in which they operate. Although VPNs can be implemented at the higher layers in this protocol stack from the perspective of service providers and commercial customers, VPN technology is usually implemented at Layer 2 and/or Layer 3. It is here where there has been much commercial interest and investment as service providers compete to offer these services to their customers. An ideal opportunity for service providers in this area is the exploitation of MPLS as a generic tunnelling technology in the implementation of such VPNs. VPN solutions that fall into the category of MPLS VPNs are Martini, Kompella and RFC 2547 VPNs.

2.4.1 Layer 2 MPLS VPNs

A simple way [HAR03][FIN04] of creating a layer 2 MPLS VPN is to use ATM or Frame Relay Virtual Circuits (VCs) between the provider edge (PE) devices and customer edge (CE) devices, and to cross-connect each of these VCs to separate MPLS LSPs to traverse the provider network. Although this is a straightforward approach it does not scale very well because of the management overhead associated with cross connecting the VCs to the MPLS LSPs and furthermore a large number of LSPs may be required in the provider network to achieve connectivity. Although no formal standards have been published in RFC form by the IETF, as part of the ongoing development in this field a number of Internet drafts (provisional IETF draft architecture documents) have been published on more comprehensive layer 2 VPN solutions. Even though (due to their provisional nature) these Internet drafts expire, leading industrial players still quote them and use them as a basis for solutions. Solutions generally fall in one of two categories, Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS).

A **VPWS** architecture is one where CE devices participating in a particular VPN are interconnected by point to point virtual circuits traversing the provider network. The PE devices behave like a virtual circuit switch. Martini and Kompella VPNs (named after the IETF drafts in which they were originally described) are examples of this type of VPN.

Martini VPNs attempt to address the shortcomings faced in simple layer 2 MPLS VPNs. The Martini architecture uses emulated point-to-point layer 2 connections called *pseudo wires* to address the scalability issues discussed earlier. VCs between the PE devices and CE devices are cross connected with Martini pseudo wires that traverse the provider network by being tunnelled through MPLS LSPs. This means that the number of LSPs in the provider network can be limited to a fixed number and that the processing of these Martini pseudo wires is pushed to the edge of the provider network. Even so, scalability can still be an issue particularly in large VPNs where there is a large management overhead associated with managing the pseudo wires. The Martini VPN model has been defined more comprehensively since it was first introduced as a IETF draft bearing the authors name. In its most recent form, this type of VPN has come to be known as PWE3. This stands for Pseudo Wire Edge to Edge Emulation. Setting up the pseudo wires has been formally defined in the context of existing MPLS specifications in an RFC entitled “Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)” [RFC4447].

Kompella VPNs alleviate the problems of management overhead by providing a solution that simplifies the configuration and management required in the provider network. Kompella VPNs are similar to Martini VPNs in that new point-to-point connections are realised by tunnelling over an existing set of LSPs that traverse the provider network. The management of these connections is to some extent automated, the Kompella solution [KOM03] uses BGP as an auto discovery and signalling protocol to coordinate the interaction between the PE devices. This means that the LSPs within the provider network are set up in an automated fashion leaving the service provider to carry out configuration only at PE devices with respect to the VCs and CE device identifiers. Tunnels between PE devices are set up as required; this is automated as knowledge of which other PE devices form part of the VPN is learnt using BGP.

In the **VPLS** architecture a key design feature is to make the provider network transparent to the CE devices participating in the VPN. A LAN is emulated across the provider network in order to achieve the desired effect. The LAN at each customer site is extended as far as the edge of the provider edge. The provider backbone then gives the perceived function of a LAN switch or bridge connecting all of the CE LAN interfaces creating the VPN. Although this solution appears to be very different from VPWS, in fact, it is simply the result of more complex processing (multiplexing and assignment) at the provider edge. VPLS extends the basic features of VPWS by adding MAC address learning, MAC based forwarding and packet replication [KNI04] in order to accomplish the perceived function of a LAN. There are similarities between PE devices offering either service, VPWS-PE and VPLS-PE devices [AND04] are functionally very similar, in that they both cross-connect CE-PE circuits to pseudowires. The difference is that this mapping in a VPWS-PE device is a one-to-one mapping between the CE-PE circuit and pseudowire (point to point) as opposed to a more complex mapping in a VPLS-PE device. In a VPLS-PE device a Virtual Switching Instance (VSI) implements this mapping. The mapping is more complex in that multiple CE-PE virtual circuits are mapped to multiple pseudowires. This allows a VPLS arrangement to implement both point-to-multipoint and multipoint-to-point connections in addition to point to point.

2.4.2 Layer 3 MPLS VPNs

RFC2547 VPNs is the most mature MPLS VPN solution. The original specification published in 1999 in RFC2547 is now obsolete, as it has undergone revision a number of times in the form of IETF Internet drafts. This process of revision has now been concluded with the publication of

a new RFC entitled “BGP/MPLS IP Virtual Private Networks (VPNs)” [RFC4364]. RFC2547 VPNs are commonly referred to as BGP/MPLS VPNs and allow a service provider to use its IP backbone to provide an IP VPN service to customers. The service provider uses BGP to exchange VPN routes. This is done in a way that ensures that routes from different VPNs remain distinct and separate, even if two VPNs have an overlapping address space. From the view of the service provider, different customers can have overlapping address spaces without causing any problems. An MPLS label is assigned to each VPN route, the process of label distribution is carried out using BGP (by piggybacking BGP messages). Whilst BGP is distributing the VPN route, it also distributes an MPLS label for that route. In the customer’s VPN, data travelling across the VPN is encapsulated with an MPLS label that corresponds to the route that is the best match to the packet’s destination address. Before being able to cross the provider’s backbone the data is further encapsulated so that it can be tunnelled across the backbone to the correct PE device. As a result, processing in the provider network is pushed to the edge, as core routers in the provider backbone do not need to know the VPN routes [RFC4364]. BGP is used to exchange VPN routes across the provider backbone this only ensures correct VPN connectivity between PE devices, VPN routes between PE and CE devices are exchanged using other means specified in [RFC4364]. This PE to CE route distribution includes using static routing, RIP and OSPF, the use of OSPF has been further specified in RFC 4577 “OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)” [RFC4577].

RFC2547 VPNs are able to offer a number of benefits [SEM01]. The key drivers for these benefits are to simplify network management from the view of the customer whilst allowing the service provider to offer a scalable, revenue-generating, value-added service.

Benefits to the customer include not having to manage the virtual backbone themselves and as a result they do not have to worry about routing issues across it. There are no constraints on the address plan used by each VPN customer, globally unique or private IP address spaces can be used. The addressing constraints are dealt with by extending BGP, allowing a new BGP address family (VPN-IPv4) to be defined for VPN routes. Addresses in this VPN-IPv4 family consist of a Route Distinguisher (RD) plus an IPv4 prefix; the Route Distinguisher enables a VPN addresses to be unique without identifying the VPN or route used. The security afforded to the customer is equivalent to layer 2 solutions without the need for encryption.

From the perspective of the service provider benefits include the ability to use a common infrastructure to deliver both VPN and Internet services. Service providers only need to administer a single virtual backbone for all their VPN customers and as a result they can deploy traffic-engineering techniques on LSPs to provide flexible QoS for their customers.

2.4.3 Summary

This section has given an overview of VPNs and what they have to offer to both service providers and customers. VPNs not only offer economic savings over leased lines, all of the solutions discussed here, some greater than others, to some extent diminish the responsibility of the customer in the management of a geographically distributed network. At the same time, VPNs allow the service provider to consolidate their efforts in the management of a single provider backbone for a multitude of customers. This section has discussed both layer 2 and layer 3 solutions. Regardless of which layer the forwarding decision is made for data flowing between CE devices over the provider backbone, all solutions employ a form of tunnelling mechanism within the core of the provider backbone. The exploitation of MPLS as a generic tunnelling technology provides service providers with an opportunity to perform traffic engineering in order to enhance the service provided. Ultimately the benefits that MPLS VPNs are able to provide translate into reduced costs [BOA02]; the total cost of ownership is reduced [CIS04a] for the service provider and, as a result, savings can be passed on to the customer.

At present there are a number of solutions under ongoing development within the framework of the IETF. The solutions discussed here are those that have received the greatest attention from industry and the IETF. The Martini and Kompella layer 2 solutions both employ pseudo wires in order to limit the number of LSPs traversing the provider network. These pseudo wires (emulated point to point links) are tunnelled through this fixed number LSPs pushing the processing to the edge of the provider network in a ploy to address scalability issues encountered with multiple point-to-point meshes in the provider core. The pseudo wires themselves could in fact be implemented as LSPs, leveraging the MPLS architecture by having LSPs within LSPs. The Kompella solution tries to address scalability further by automating the routing and formation of the fixed number of LSPs traversing the core. A more complicated processing model is found in the VPLS layer 2 solution which, like the Martini and Kompella solutions, uses pseudo wires to alleviate scalability issues within the provider core but adopts a more complicated mapping model to enable multiple CE-PE VCs to be mapped to multiple

LSPs. The VPLS solution as a result allows for various mapping configurations in addition to point-to-point, point to multipoint and multipoint to point are supported. The most mature solution is RFC2457 based VPNs which uses BGP and MPLS to implement the VPN. This solution allows the service provider to use their IP backbone to provide an IP-based VPN. VPN address overlap within the provider network is addressed with extensions to BGP. Although this is a layer 3 solution, forwarding within the provider core is still the result of layer 2 MPLS switching.

Standards in this area are under ongoing development within the IETF with some solutions still in IETF draft form. What is certain, however, is that all the discussed solutions shift the management onus from the customer to the service provider. As a result, a key factor in the provider management of the VPN backbone will be traffic-engineering tools that can work in harmony with a multitude of VPNs over a shared backbone.

2.5 Multicast Routing

Routing protocols such as OSPF, RIP and BGP are designed to create routing tables that describe point-to-point routes across a network, these routes each relate to the path taken by a packet from a single source to a given destination. A source may elect to send the same data to multiple destinations; this data transmission may be to all nodes in a network (broadcasting) or to a subset of these nodes. A subset of destination nodes in a network receiving the same data transmission is called a multicast group. Multicast routing is concerned with the calculation of a Multicast Delivery Tree that includes all the nodes in the multicast group. In general, the multicast delivery tree can take one of two forms; a source or shared tree [FOR03].

A source tree is a point to multipoint tree, where there exists a dedicated tree for each source-group pair. Such a dedicated tree usually uses the shortest path through the network and, as a consequence, is often referred to as the shortest path tree for a particular source-group pair.

A shared tree is a multipoint-to-multipoint tree, where a dedicated tree for each multicast group exists. Unlike the source tree, the shared tree is unable to use the shortest path tree for each source, as it must provide paths for multiple sources. Whilst this may be perceived as a disadvantage in the sense that non-optimal routes are being used, a shared tree offers reduced overheads in terms of routing table space.

2.5.1 Reverse Path Forwarding

Reverse Path Forwarding (RPF) is a multicast forwarding algorithm used by multicast protocols to implement source trees in a network [IBM01]. In order to operate, the RPF algorithm must have access to a conventional routing table with the shortest path to all destinations. The algorithm is a distributed one, at each node a reverse path table is maintained. The reverse path table is constructed using information in the conventional routing table, it contains the interface used by the node to reach a given source node. Naturally this interface leads to its upstream neighbour that forms part of the shortest path back to the source node. This interface is known as the preferred interface for a particular source node. RPF examines the source address of a received packet; a check is then made against the reverse path table. This check determines whether or not the packet has arrived on the preferred interface for the specific source address. If the received packet has arrived on the preferred interface it is forwarded on, it is sent out on

all interfaces except for the preferred interface on which it arrived. A packet identified as having arrived on an interface other than the preferred interface is discarded. This process propagates throughout the multicast group; the resulting multicast delivery tree is loop free, as a result of discarding packets arriving on interfaces that are not marked in the reverse path table.

Using RPF to form the multicast delivery tree has the following benefits:

- The tree formed is loop free and provides the fastest delivery of packets to members of the group, as the tree is also the shortest path first tree for a particular source-group pair.
- Because each source-group pair has a dedicated tree, traffic does not congregate on a subset of links. Network resources are better utilised as the trees from many source-group pairs are distributed over many links.

2.5.2 Centre Based Tree Algorithm

The Centre Based Tree (CBT) algorithm is used to form a shared tree for a particular multicast group [FOR03]. As the name suggests this algorithm is based on centre point in the tree that acts for all members of the multicast group. A central node or core router is chosen in the network, source nodes send packets to this central node where it is forwarded to other members in the multicast group. The multicast delivery tree is formed as a result of nodes sending join requests directed to the central node. This join request may traverse a number of intermediate nodes as it travels to the central node. When a node receives such a request and if it is already part of the delivery tree, it adds the node making the request to its list of nodes that form part of the delivery tree for this particular group. The main disadvantage of CBT is that the resulting multicast delivery tree is sub-optimal.

2.5.3 Multicast Routing Protocols

A number of protocols exist dedicated specifically for the task of setting up multicast delivery trees. One of the first multicast routing protocols to be adopted is **Distance Vector Multicast Routing Protocol** (DVMRP). DVMRP is defined in [RFC1075] and is based on RIP with extensions for multicast. The protocol forms source trees using the RPF algorithm; DVMRP creates its own unicast routing table in order to do this. **Protocol Independent Multicast** (PIM)

consists of two separate protocols developed to meet the demands of different multicasting environments; these are PIM-Dense Mode (PIM-DM) and PIM-Sparse Mode (PIM-SM). The protocol independence of PIM relates to the fact that the protocol works regardless of the unicast protocols already deployed in a network. PIM-DM has been designed to suit networks with low delay and high capacity [COM00] such as local area networks. PIM-DM [RFC3973] assumes the existence of a unicast protocol and uses the conventional routing table in its execution of the RPF algorithm in order to form the multicast delivery tree. In contrast, PIM-SM forms shared trees using CBT algorithm; defined in [RFC2362] [FEN06]. PIM-SM designates a router to act as a Rendezvous Point (RP) that assumes the role of the core node described in the section on CBT. PIM-SM also designates backup RP nodes in the event that the current RP becomes unreachable as a result of a failure. PIM-SM has been designed for use in wide area networking where the location of multicast group members is more widely spread and member density is said to be sparse. Although PIM-SM primarily forms shared trees, it is able to switch to source trees when necessary and the simultaneous deployment of both PIM protocols is also allowed. **Multicast Extensions to OSPF**, or MOSPF [RFC1584] as it is better known, is not a dedicated multicast protocol, but as the name suggests an extension to the unicast protocol OSPF. Naturally this means that MOSPF can only operate in networks where OSPF is already deployed as a unicast routing protocol. The multicast extensions use the information contained in the OSPF Link State Database to compute shortest path first source trees using shortest path algorithms such as Dijsktra's [DIJ59]. Information relating to group membership is distributed within an OSPF area using a group membership link state advertisement. Such information on group membership is stored in the link state database along with LSAs received for the purpose of unicast routing and is used collectively to compute the necessary multicast delivery trees. When a group spans more than one OSPF area the principles adopted by MOSPF are similar to the operation of the unicast protocol. Multicast Area Border Routers are used by MOSPF to propagate group information in their area to the backbone area.

2.5.4 Summary

This section has focused on the basic principles associated with multicast routing; central to this is the formation of a multicast delivery tree. In general a multicast delivery tree can take one of two forms; a source or shared tree. A source tree is a P2MP tree and usually follows optimal paths from the source to group members; a single tree exists for each source-group pair. A source tree is normally formed using shortest path first algorithms such as Dijkstra's [DIJ59]

this calculation is normally implemented as part of the RPF algorithm. The RPF algorithm relies on selecting preferred interfaces back to the source. It is the selection of this optimal interface that is calculated using an algorithm such as Dijkstra's. RPF is used by the DVMRP and PIM – DM multicast routing protocols. Source trees are also used by MOSPF. These are calculated using information held in the Link State Database. Shared trees are MP2MP trees; in this type of tree all sources for a particular group use the same tree. Shared trees are unable to provide optimal multicast delivery, as they must serve the interests of multiple sources. CBT is the principal algorithm used to form shared trees and relies on the use of a core node to act as the central point for this type of delivery tree. CBT is used by the PIM – SM multicast routing protocol.

2.6 Link Reversal Routing (LRR)

Link Reversal Routing (LRR) refers to a class of routing protocols that are highly adaptive and resilient in the face of network changes [PER00]. This class of protocols has been designed for use in ad-hoc networks where the topology of the network is subject to frequent alterations and in comparison to fixed networks is often regarded as ephemeral. This has led to a set of protocols that by their very nature are robust and decentralised.

Fundamental to the approach of LRR is the objective to minimise the amount of routing overhead between nodes when reacting to topological changes, be they the result of failure or new nodes joining a network. Minimising the signalling overheads is achieved by having a localised reaction take place in response to these changes in topology. Central to such an approach is, for any given destination, the maintenance of a Directed Acyclic Graph (DAG) routed at that particular destination. A DAG is a loop free graph routed at a particular destination, thus only the destination in a graph of this type may have incoming links only. LRR is different from conventional routing approaches, most of which are either based on distance vector or link state methodologies. LRR protocols do not maintain an additive distance metric as used by distance vector protocols, nor do they maintain information about the topology of a network in a link state database as used by link state protocols. The only information maintained by nodes in the LRR approach is that which is sufficient to maintain a DAG for each particular destination. The implications being that the only information kept by a node running an LRR protocol is information from its immediate neighbours with respect to the destination and, as a result a node has insufficient information to perform a shortest path calculation or for that matter knowledge of its precise distance from a particular destination. To form a DAG the information kept need only be “relative” and only locally significant.

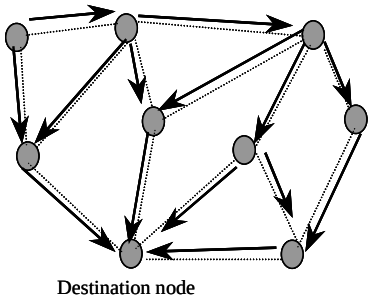


Figure 5:- A DAG relative to the destination node

As shown in Figure 5 the DAG is effectively a graph of multi-paths to the destination node, so maintaining the DAG is essential in ensuring that each node has a path to the destination.

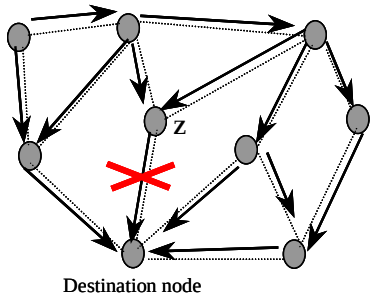


Figure 6:- A DAG showing node Z losing its last outgoing link

A common feature of all LRR protocols is the trigger used to instigate a reaction to a failure; this trigger being the loss of a node's last outgoing link as shown in figure 6. Naturally this does not apply to the destination node. Although the loss of a link as a trigger is by no means unique to LRR protocols the fact that only the loss of a node's last outgoing link instigates a reaction by these protocols means that the frequency with which LRR protocols react is suppressed. Furthermore, unlike many routing schemes where the loss of a link would entail a network wide reaction (link state database updates) or a communication to nodes using the failed link (or node) as part of their shortest path calculation, LRR limits its reaction to a limited number of localised nodes.

There are a number of proposed protocols that have been proposed that fit into the general LRR classification. The most recent effort was the result of research sponsored by the US DoD for use in large-scale dynamic, heterogeneous military networks and the protocol produced is called the Temporally Ordered Routing Algorithm (TORA) [PAR97][PAR01]. The earliest work in the field of LRR, which incidentally TORA is based on, is the Gafni – Bertsekas (GB) algorithm [GAF81]. This work first published in 1981 has two variants: the Full Reversal method and the Partial Reversal method. Of particular interest is the Partial Reversal method using the notion of height. The height-based version of the Partial Reversal method is central to the resilience strategy employed by DAM and is described here in detail.

In the height-based version of the Partial Reversal method [PER00] each node is associated with a height triple $\{\alpha, \beta, z\}$ that is lexicographically ordered [GAF81] where z denotes a node's

unique identifier and α & β are both integers. α is a reference level and β an offset to that reference level. The initial values of {α, β, z} for all nodes are such that α is zero and β is adjusted such that for a link flowing from z to y the triple {α, β, z} > {α, β, y}; naturally the {α, β, z} triple is always the lowest for the destination node. This forms the initial DAG. The *k*th iteration is implemented as follows [GAF81]. A node *z* other than the destination for which $(\alpha_z^k, \beta_z^k, z) < (\alpha_y^k, \beta_y^k, y)$ for all neighbours *y* increases α_z^k to :-

$$\alpha_z^{k+1} = \min\left\{\alpha_y^k \mid y \text{ is a neighbour of z}\right\} + 1$$

and sets β_z^k to :-

$$\beta_z^{k+1} = \begin{cases} \min\left\{\beta_y^k \mid y \text{ is a neighbour of z with } \alpha_z^{k+1} = \alpha_y^k\right\} - 1 \\ \text{if there exists a neighbour y with } \alpha_z^{k+1} = \alpha_y^k \\ \text{otherwise:} \\ \beta_z^k \end{cases}$$

All other nodes maintain the same (α_y, β_y, y).

The operation of the height-based Partial Reversal algorithm is best described in words and an example to show its operation follows this description.

The algorithm is distributed and a new iteration is invoked at any node that loses its last outgoing link (except the destination). This first causes α at that node to be set to (α + 1) of the lowest α of its neighbours. If, after this operation, there exists a neighbour(s) with the same value of α as this newly set α, β at that node takes on the value of (β - 1) of the lowest β in this set of neighbour(s) with equal αs, if no such neighbour(s) exists β at this node remains unchanged. All other nodes keep their {α, β, z} triple constant. This is illustrated in figure 7.

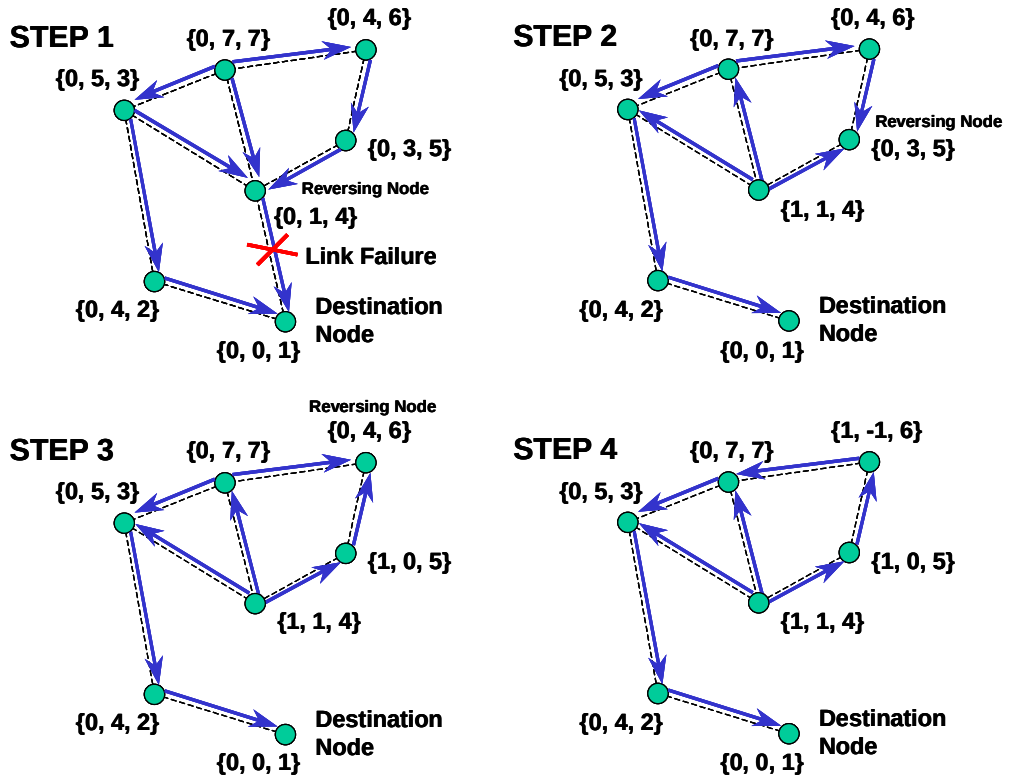


Figure 7:- Shows the Height triple of the Reversing Node changing during subsequent iterations of the Partial Reversal Method

The example shown in figure 7 has been constructed to show the major aspects of this algorithm and hence the number of steps to DAG restoration is non typical. In a typical case the number of steps required to reconstitute the DAG would generally be less, usually only one or two iterations. In this figure the Partial Reversal Method is invoked as a consequence of the link failure between node 4 and node 1. Note that in step 1 the destination "node 1" has the lowest height triple in this initial DAG and that all values of β are such that the heights of nodes are in accordance with the directions of links in this graph. In step 1, as a result of the failure node 4 loses its last (and only) outgoing link. Node 4 becomes the reversing node and initiates the Partial Reversal Method upon itself (this is the distributed nature of this algorithm). Node 4 first examines all (α)s of its neighbours and chooses the lowest of this set; in this case the lowest α is zero (all its neighbours have a value of zero for α in this iteration) node 4 then sets its own α to this value +1. Having set its α to 1 it then proceeds to identify neighbours with the same α in order to set β. Since none exist, node 4 does not adjust its value of β and proceeds with link reversal using {1, 1, 4} as its new height triple as shown in step 2. As a consequence, node 5 becomes the reversing node in step 2 having lost its last outgoing link. Node 5 then identifies

the α values of nodes 4 and 6 as being 1 and 0 respectively. Node 5 subsequently sets its own α to (0 + 1) 1. Node 5 then identifies neighbouring nodes with the same value of α that it has calculated and chooses the lowest β in this set. In this case the minimum of this set is β = 1(from node 4). It proceeds to set its own value of β to this minimum – 1. Setting its β to (1 – 1) gives zero. Having calculated its new height triple as {1, 0, 5} node 5 proceeds with link reversal as shown in step 3. In step 4, node 6 performs a similar operation to that performed by node 5 in step 3, arriving at a new height triple of {1, -1, 6} and thus completing the final link reversal in the example.

TORA builds on the partial reversal method adding functionality to counter instability in the face of network partition. The GB algorithm is proven to converge very quickly [PER00] when not faced with network partitions. A network partition occurs when part of the network becomes completely disconnected from the destination node; the partial reversal method becomes unstable when this happens. To address such a shortcoming the designers of TORA use a five-tuple height metric that includes a time tag. However, the extended height metric adds to the complexity of the protocol and is not necessary in all applications. The GB partial reversal method is simple and highly effective within the confines of its stable operation; that is, when not faced with network partitions.

2.6.1 Summary

This section has introduced some of the key aspects of LRR. LRR protocols are primarily designed for use in ad-hoc networks and, by nature, are a class of routing protocols that are highly adaptive and resilient in the face of network change. In LRR, reaction to topological change is limited to a localised reaction. Central to this class of protocols is the maintenance of a DAG. A DAG is effectively a graph of multi-paths to the destination node. LRR algorithms are distributed and are neither distance vector or link state. In this section, attention has been given to the GB algorithm and in particular the height-based Partial Reversal method that is central to the work in this thesis. In the partial reversal method, it is typical that only a few iterations of the algorithm are required in reaction to a topological change or failure.

2.7 Multipoint-to-Point (MP2P) Trees

A MP2P tree in terms of network topology is an inverted tree routed at the destination. All sources to a particular destination are connected via a single MP2P tree as shown below in figure 8b; figure 8a shows the P2P equivalent. In the context of MPLS, such a configuration replaces each point-to-point LSP connection with a single LSP for each destination, thus defining a MP2P LSP for each destination. Rosen *et al* formally defined this type of LSP in the first draft of what is now [RFC3031] the MPLS architecture RFC. The path to all destinations in a network is realised as an overlay of these MP2P LSPs as opposed to a mesh of P2P LSPs.

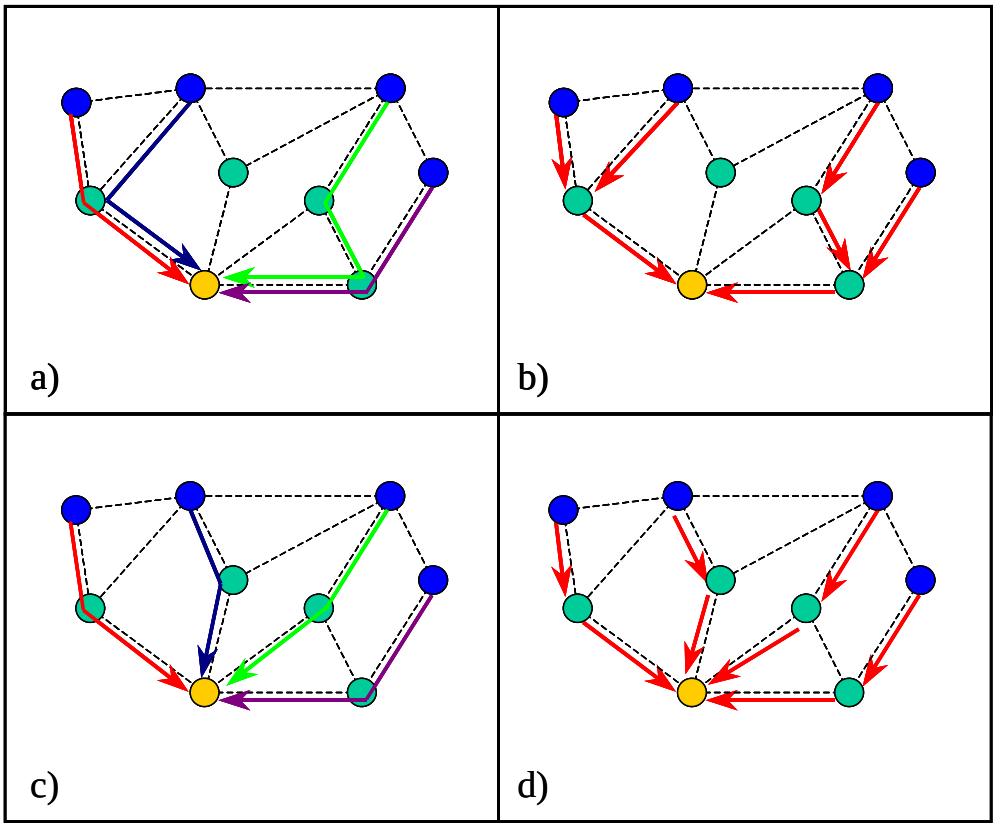


Figure 8:- a) & c) P2P LSPs b) & d) MP2P Tree/LSP

Figure 8a shows four P2P LSPs, the blue nodes in this arrangement are ingress points and the gold node is the destination/egress node. Equivalent connectivity can be achieved by using the MP2P LSP shown in figure 8b. In this arrangement the four LSPs shown in figure 8a are replaced by a single MP2P LSP. In figure 8b the label space is also reduced at both green nodes immediately upstream of the destination node in the MP2P tree. The reduction in label space is

effectively due to these particular upstream nodes acting as merge points where the P2P LSPs in figure 8a congregate along the same route. P2P LSPs destined for the same destination node are increasingly likely to share the same links as they approach the destination; in fact, near to the destination this can become unavoidable as the local topology dictates the routes available to reach the destination. Figure 8c shows four P2P LSPs providing connections between the four blue ingress nodes and the destination node, in this arrangement the P2P LSPs do not share any common links. Figure 8d shows the equivalent MP2P LSP; whilst the number of LSPs is reduced from four to one there is no reduction in label space, as the equivalent P2P LSPs shown in figure 8c do not share any common links. In the P2P LSP arrangements in figure 8 if the green intermediate nodes are also made to act as ingress nodes with their own LSPs to the destination then the number of P2P LSPs using common links to a particular destination is increased and therefore label space is reduced at each node on each of these common links. The label space reduction is achieved by using a single label to serve more than one ingress-to-egress connection for a particular link direction. The number of labels used in a P2P LSP arrangement is the summation of the total number of links traversed for each P2P LSP in figure 8a this is equal to 9 (2 + 2 + 3 + 2) and in figure 8c the total number of labels used is 8 (2 + 2 + 2 + 2). In the case of a MP2P LSP the total number of labels used is simply the total number of links traversed for a particular MP2P LSP (for fig. 8b this equal to 7 and for fig 8d this is equal to 8). The amount of label space reduction is thus dependant on the number of P2P connections being replaced by a particular MP2P tree and the extent of this reduction is dependant on how link disjoint these P2P connections are. In the examples in figure 8 there is no label space reduction between figure 8c and 8d whilst the total number of labels used is reduced by 22% for figures 8a and 8b. Section 6.4 gives examples with label space reduction using MP2P LSPs ranging from 76% - 86% compared to the equivalent P2P LSPs. Clearly label space reduction is not always achievable by using MP2P LSPs and where it is achievable the reduction factor is not fixed. Label space reduction should not therefore be the sole motivating factor for using MP2P LSPs over P2P LSPs, the ability to manage a group of connections collectively in the form of MP2P LSPs is more important since this presents opportunities in terms of scalability and resilience. In MPLS VPNs the number of LSPs in the service provider core network is usually a fixed number so that connectivity can be provided between all PE devices achieving this in a provider network with 20 PE devices would require 380 P2P LSPs [(N * (N - 1)) where N is the number of nodes / PE devices]. Using MP2P LSPs the same connectivity can be achieved using 20 MP2P LSPs this is equal to N the number of PE devices. Whilst the level of label space reduction is dependent on how many intermediate nodes are acting as PE devices,

managing 'N' MP2P LSPs, as opposed to (N * (N - 1)) P2P LSPs, provides for a more scalable solution.

RFC 3031 defines MP2P LSPs as an entity within the MPLS framework; this definition however is conceptual in the sense that the implementation of such LSPs is not discussed. The formation of MP2P LSPs within the context of efforts made by the IETF remained undocumented until February 2006 when the first Internet draft on this subject was published entitled "Supporting Multipoint-to-Point Label Switched Paths in Multiprotocol Label Switching Traffic Engineering" [YAS06]. This Internet draft [YAS06] focuses on extensions to RSVP-TE for the support of MP2P LSPs. Signalling issues are discussed for the merging of P2P LSPs at appropriate merge points. However, strategies for the identification of candidate P2P LSPs and merge points are not discussed in [YAS06]. The implementation of MP2P LSPs thus remains very much an open issue, as to date, only a few of schemes have been proposed for the implementation of such MP2P trees. This section gives an overview and discusses the merits of the schemes that have been proposed.

2.7.1 Integer Programming

Saito *et al* published what appears to be the first scheme to address the issue of MP2P LSPs after their definition by the IETF within the framework of RFC 3031. In the scheme proposed in [SAI00] MP2P LSPs are pre–assigned; they are created offline on the basis of network topology alone. Flow assignment and actual MP2P LSP set up are independent of one another. This is illustrated in the figure 9 below.

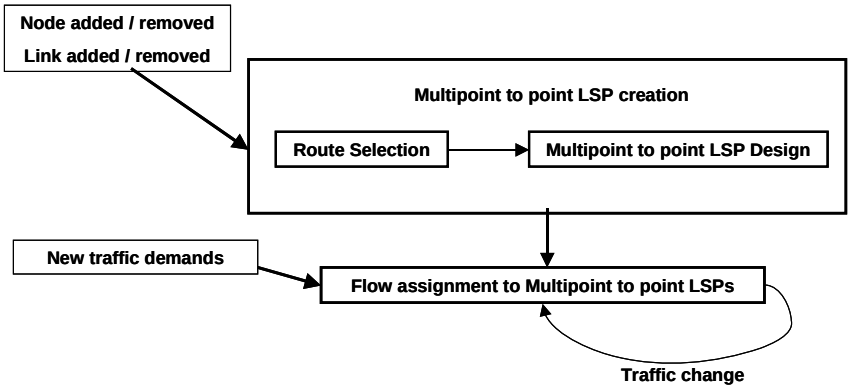


Figure 9: - Operation and Design Flow of Saito Scheme [SAI00]

The design of the MP2P LSPs is formulated as a 0-1 integer programming problem [SAI00] whereby the MP2P tree is the result of the appropriate selection of P2P paths between the set of associated ingress nodes and the egress that will form the root of the tree. Saito *et al* quite rightly recognise that including all possible routes in such a calculation would require excessive calculation time. In order to address this issue the proposed scheme limits the number of selected routes involved in the calculation by excluding long routes where possible from the list of candidate routes by means of a combined Dijkstra and depth first search algorithm. The criteria they have chosen for this selects a set of routes for each ingress/egress instance whereby each route in this set has a length no greater than shortest path plus ξ which can be a given number such that in the set of routes there exists at least one route that does not share a single node with any other route in this set. If no such route set can be found to meet these criteria than a pair of routes must be chosen that do not share a single intermediate node and have the smallest and second smallest path lengths.

Having made the appropriate selection of candidate P2P routes { $p(i,e)$ }, MP2P trees can be calculated. The constraints placed on the resulting MP2P trees by this scheme are that a MP2P tree must not contain a loop structure and must comprise of some of the selected routes for that particular egress. In addition, each of the selected routes must be included in at least one of the MP2P LSPs. These constraints fall under the objective function of the scheme that is to minimise the number of candidate MP2P LSPs. This scheme can be formulated into an integer programming problem by associating each link with a set of variables $h_{(l,m)}^{t_e}$ which takes on the value of '1' if the candidate MP2P tree t_e uses link (l,m) or '0' if does not. The loop constraint is thus formulated into the condition that for each node the summation of $h_{(l,m)}^{t_e}$ in outgoing links is equal to '1'. The constraint that each MP2P tree must include at least one of the selected routes takes the form of the condition that for each route the summation of $h_{(l,m)}^{t_e}$ along the route is equal to the summation of hop counts for that route. In order to formulate the third constraint the designers of this scheme introduce a 'route accommodation indicator' $\delta_{p(i,e)}^{t_e}$ whereby a selected path is represented by the term $p(i,e)$. The third condition takes the form that the summation of $\delta_{p(i,e)}^{t_e}$ over a MP2P LSP candidate is greater than or equal to '1' that is to say that each MP2P LSP candidate must contain at least one of the selected routes. The conditions are met under the objective function whereby r^{t_e} is set to '1' if t_e includes part of the selected $p(i,e)$ routes such that the summation of all r^{t_e} for a particular egress is minimised.

In their description of this scheme Saito *et al* have presented a numerical evaluation of the reduction of label space in comparison to the P2P approach using their integer-programming scheme for the formulation of the MP2P trees. Bhatnagar *et al* [BHA02] state that the scheme is in general NP ("non-deterministic polynomial time") hard and cannot be used frequently in large MPLS domains because of its calculation complexity. This statement is supported by the benchmark text in the area of complexity theory and NP completeness [GAR79] where 0-1 integer programming is listed as an NP complete problem. Simply put, NP complete problems are a class of problem for which it is acknowledged to be difficult in finding solutions (in fact solutions may not exist for all cases). The complexity associated with NP complete problems can also lead to long calculation times. Resilience in this scheme is addressed by provisioning complete alternate trees to act as backup trees for other MP2P trees, however such a mechanism relies on an alarm signal propagating back to the source in order to switch to use of the backup tree.

2.7.2 Simple Path Merging

Bhatnagar *et al* [BHA02] [BHA03] [BHA05] present a simple path-merging scheme for the formation of MP2P LSPs. This scheme takes, as its input, the paths to a particular egress node and then merges them where possible to create a set of MP2P LSPs. The authors have shown [BHA03] that the problem of forming the minimum number of trees by path merging is NP complete. The authors have proposed a simple polynomial time heuristic based algorithm with the aim of reducing label space by creating the minimum number of trees. The authors describe their algorithm as follows:

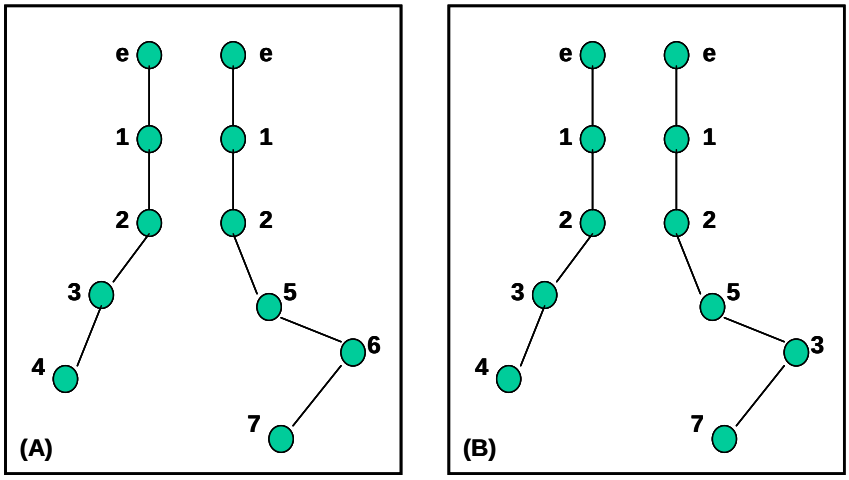


Figure 10:- Example showing the path merging concept [BHA03] [BHA05]

The algorithm is repeated separately for each egress, associated with each egress is a set of paths P_e from the sources to the egress. It works by computing a merging index for every pair of paths in the set P_e such that the merging index for $p_i, p_j \in P_e$ is m_{ij}. The merging index m_{ij} is computed by counting the number of contiguous common nodes starting at 'e' the egress forming a common chain for the pair of paths being considered. To avoid loops as a result of merging, the merging index is set to zero if there exists one or more common nodes that are not part of the contiguous common chain. This is illustrated in figure 10. The merging index for figure 10(A) is 3 as the common chain is comprised of three nodes {e, 1, 2}. Even though figure 10(B) shares this property with the latter, its merging index is set '0' as this pair of paths also shares a common node {3} that is not part of the common contiguous chain and is hence not eligible for merging. It is clear from figure 10(B) that if these two paths were merged there would be a loop between nodes 2 & 3 via node 5. Once the merging index has been calculated for all pairs in the set P_e the authors define a heuristic for merging the resulting trees into denser trees minimising the tree set associated with any particular egress. This heuristic states that in selecting the next pair of trees with which to apply the merging algorithm - consider the pair $p_i, p_j \in P_e$ for which the merging index m_{ij} is greatest. This is done so as to form the longest common chain possible between pairs so that the resulting tree allows the greatest scope for other paths to be merged with it. This process of merging continues until either all paths have been merged into a single tree or until there is a stalemate in that all merging indices are zero and no further merging can take place resulting in more than one tree for a particular egress.

The authors of this work state in [BHA02] [BHA03] and [BHA05] that the merging problem is NP complete and as a result their algorithm is just a heuristic and there can be cases in which the result of this algorithm is sub-optimal. Moreover, the algorithm in some cases will not arrive at the minimum number of trees that is possible for a particular input set of paths. Even though this may be the case a sub–optimal solution still offers a reduction in label space over P2P LSPs still satisfying the design goal that the authors set out to achieve. Bhatnagar *et al* [BHA03] present results showing the reduction in label space that their algorithm has to offer over P2P LSPs stating the importance of this reduction by referring to the need to support CE-based VPNs with up to 50000 CE devices that with a P2P approach would require 50000 * 49999 LSPs.

2.7.3 Linear Programming

Applegate *et al* [APP03] present a method for reducing the number of MPLS labels to a bounded number of N + M labels, where N is the number of nodes and M is the number of links in the network. Applegate *et al* do this without increasing any link load [APP03] using a rerouting scheme to achieve this. The rerouting aims to find an alternative route using constraint-based routing based on link capacity, such that no link gets an increased load [APP03]. Based on this the authors formulate a multi-commodity flow problem corresponding to the constraint-based routing problems in order to formulate MP2P trees and the associated labels. This multi-commodity flow problem is a special type of linear programming problem for which the simplex method (refer to [SIE02] for details of the simplex method) can be used to find a solution. The authors use commercially available linear programming software packages such as CPLEX [ILOG1] to do this. Applegate *et al* [APP03] show that their method is very effective in reducing the number of MPLS labels. The authors also give an experimental example indicating the time taken to calculate the trees and labels; this is about 24 minutes for a network with 300 nodes and about 1000 links. This is perhaps the reason that the authors suggest their method be used on a daily basis and that the use of the new labels is done incrementally. The method presented [APP03] appears to be slow in finding a solution; in any case this linear programming problem is calculated offline in a centralised method in order to be able to meet the constraints that the authors have introduced in relation to link capacity and load. This consideration casts doubts over the practicality of such a scheme. In addition, the

authors use a commercial linear programming software package to compute their offline solution this alone may render this type of approach impractical.

2.7.4 Mobile Agents

Gonzalez-Valenzuela *et al* [GON01] [GON02] present an agent-based methodology for the discovery and formation of MP2P trees. Mobile agents are employed to seek out and finalise the MP2P trees. This process involves using colonies of agents to search out and find all possible shortest path trees from every ingress to the egress of interest. Once these trees have been obtained, the intersections of all the trees are evaluated by the colony of agents to identify the links of individual paths that make up the final MP2P tree. This is done by pruning the paths that are considered to be non-essential within this set of candidates and, by doing so, obtaining the desired MP2P tree. This process is shown in figure 11.

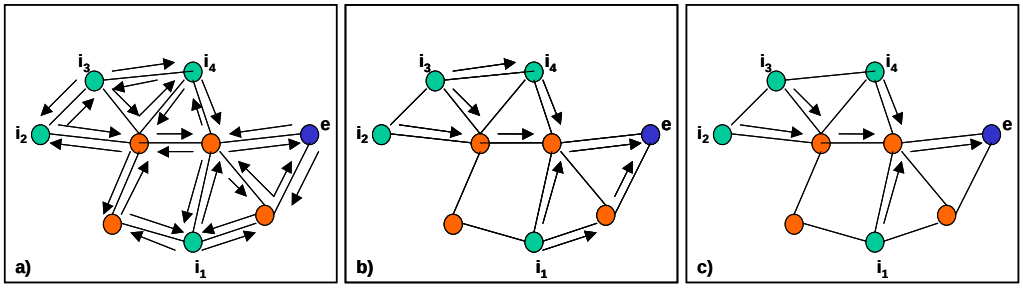


Figure 11:- Graphical representation of the MP2P discovery using mobile agents [GON02]

In figure 11, a) shows the agent search for the shortest path trees; in b) possible merge nodes are identified; and finally in c) after appropriate pruning the final routes are defined.

Although this approach is highly innovative there are certain practical considerations that at present, make it unsuitable and thus not a subject for further discussion within this thesis. Setting up paths for fast forwarding within MPLS is a comparatively low level task and the overhead incurred as a result of using agent technology is an important consideration. In particular intra-agent communication and control overhead [KAR98] and the added security considerations [KAR98] of using mobile agents make this solution impractical.

2.7.5 Summary

MP2P LSPs present an important traffic engineering opportunity with the potential to reduce the amount of label space required within an MPLS domain. Such a reduction is particularly relevant as the size of a MPLS domain increases. Moreover, particularly large MPLS domains present a challenge as managing them can become difficult, particularly if full mesh connectivity is required. To-date four schemes have been proposed for the creation of such MP2P LSPs including one that employs a colony of mobile agents to seek out and create the MP2P trees. Of particular relevance to this study are the schemes proposed by Saito *et al* [SAI00] and Bhatnagar *et al* [BHA02][BHA03]. Though no complete protocol is discussed in either of these publications, both present methodologies for the creation of MP2P trees. Saito *et al* propose a scheme that formulates the creation of such MP2P LSPs as a 0-1 integer-programming problem. It is accepted that calculating the MP2P trees is not a trivial task and that doing so using 0-1 integer-programming acknowledged to be NP complete can lead to excessive calculation time. Bhatnagar *et al* state that for this reason the scheme proposed by Saito *et al* [SAI00] cannot be used frequently. Bhatnagar *et al* [BHA02][BHA03] propose a simple path-merging algorithm that is considered to be a simple polynomial time heuristic. Although this algorithm does not posses the calculation complexities of the proposal by Saito *et al* [SAI00] it does not always arrive at the optimal solution in terms of the number of MP2P trees associated with a particular egress an example showing this is described in [BHA03]. Unlike the mobile agent scheme both the schemes of Saito *et al* [SAI00] and Bhatnagar *et al* [BHA03] require the appropriate selection of P2P LSPs that in turn are used as input into their respective algorithms. In both cases, this calculation of the tree is done offline. The scheme presented by Applegate *et al* [APP03] is effective in reducing the number of MPLS labels; however, it is based on finding the solution to a linear programming problem that is slow and is conducted offline. This method is considered to be impractical within the context of this thesis. Resilience is only partially addressed by Saito *et al* [SAI00] with the use of complete MP2P LSPs as backup trees; this method relies on the appropriate use of alarm signals and the propagation of signalling back to the ingress in order to facilitate the switch to the back up tree.

2.8 Summary

The sections that comprise Chapter 2 have highlighted aspects of importance in this research area and indeed the need for this work to be done. Traditional IP routing and forwarding has been discussed in the context of service provider requirements. Layer 3 forwarding requires a forwarding decision to be made at each intermediate node on a packet-by-packet basis; this is clearly inefficient in terms of processing within the provider core. Instead, many service providers employ fast forwarding technologies such as ATM or Frame Relay within their core network. ATM and Frame Relay are able to provide fast layer 2 switching by simplifying the hardware and software used within the core [RAH91]. Even so, such an approach is purely connection orientated and is less flexible than the connectionless best effort service of IP forwarding where resources can be shared and used only as and when required, and where packets are delivered individually based on hop-by-hop basis.

MPLS is able to act as a shim layer between layer 2 switching and layer 3 forwarding mechanisms in a bid to combine the flexibility and resilience of IP forwarding along with the speed and simplicity of ATM. MPLS provides a framework with which to facilitate this but it does not provide the traffic engineering solutions desired by service providers in maximising utilisation, efficiency and range of services that they are able to offer customers. Instead, MPLS provides a rich feature set with which a service provider can tailor their own TE solutions that can be used to provision multiple service classes as opposed to the single service ‘best effort service’ provided by IP.

The management issues associated with providing a connection-orientated service remain unsolved, despite the fact that the MPLS feature set is able to provide some of the flexibility normally linked with a pure layer 3 approach such as IP. The fact remains that if a P2P approach is adopted in implementing an MPLS domain it can become unmanageable, particularly in large networks where thousands of LSPs must be maintained. A fully meshed arrangement in such an instance would require a single LSP to be maintained for each ingress-egress pair resulting in large overheads both in terms of label space and management of the domain by the provider in its entirety. This is of particular concern to service providers providing VPN solutions to their customers. The most prolific VPN solutions currently under development have been discussed earlier. Although no hard standards have been published, this has not deterred service providers from investing in these solutions as market forces dictate a move from leased lines to “MPLS VPNs”. What is particularly pertinent to the work in this thesis is that, regardless at which layer

the forwarding decision is made in such VPNs, all the discussed solutions use MPLS as the generic tunnelling mechanism with which to forward data. Effective traffic engineering in such a context not only alleviates the problems associated with a fully meshed arrangement (which are still applicable to solutions limiting the number of LSPs in the core through the use of pseudo wires) it ultimately translates into more profitable service provider networks that are able to operate more effectively.

Another major consideration is resilience in MPLS networks and the need to provide timely and cost effective restoration. MPLS recovery schemes centre around protection switching which requires pre-established backup paths in one form or another impacting directly on how economic such solutions are. Recovery times for protection switching strategies can be very fast but there is a direct trade-off between speed and cost. Apart from the costs associated with protection switching the distance failure notification messages must travel to get to the point of repair can be a drawback. This is often cited [HUA02] [MAR03a], especially in end-to-end protection when these messages must propagate back to the ingress/egress. Rerouting as an alternative to protection switching is able to provide simple cost-effective recovery but remains largely unappealing because it tends to rely on the re-convergence of higher layer routing protocols that cause recovery times to be slow in comparison with other technologies. Overcoming this challenge in reducing recovery times for rerouting strategies offers benefits that would make rerouting an attractive option for MPLS recovery. Ahn *et al* [AHN02] propose a scheme that attempts to address the issue of MPLS recovery using rerouting.

Defined in RFC3031, MP2P LSPs offer a traffic engineering opportunity that is able to offer scalable solutions in terms of both a reduction in the required label space as well as the magnitude of LSP maintenance within a domain. Furthermore, multiple MP2P LSPs can be deployed in providing multiple service classes. The importance of MP2P LSPs is highlighted by the example given [BHA03] where 50000 * 49999 P2P LSPs would be required to support a VPN with 50000 CE devices. The use of MP2P LSPs in such an arrangement would dramatically reduce the number of LSPs required to support all the CE devices not only making the volume of LSPs in the provider core more manageable but also providing an architecture in which multiple service levels are more easily implemented and managed. Moreover, multiple MP2P LSPs can coexist within the core supporting different CoS traffic whilst still maintaining the connectivity of a fully meshed arrangement. Four schemes have been discussed for the creation of MP2P trees. Of the greatest relevance to this thesis are those proposed by Saito *et al* [SAI00] and Bhatnagar *et al* [BHA03]. Whilst both solutions address the problem of creating

MP2P LSPs, they both rely on the appropriate selection of P2P LSPs as input to their algorithms. The scheme proposed by Saito *et al* [SAI00] formulates the MP2P tree creation problem as a 0-1 integer programming problem that is, in general, NP complete leading to an algorithm of significant calculation complexity, for this reason [BHA03] Bhatnagar *et al* state that it cannot be used frequently because of the processing overheads associated with its time complexity. The scheme proposed by Bhatnagar *et al* [BHA02][BHA03] is more straightforward and is described by its authors as a simple polynomial time heuristic. The algorithm of Bhatnagar *et al* [BHA03] is a simple path-merging algorithm coupled with a path selection heuristic for the merging of paths into subsequent trees. However, as a result of this heuristic the algorithm can lead to sub-optimal results.

Whilst MP2P LSPs can provide a way to reduce label space it is not the most important consideration in terms of managing an MPLS domain. In terms of service provision MP2P LSP resilience is more important than label space reduction. The schemes that have been described in section 2.7 focus on the creation of the MP2P LSPs none of these schemes propose a dedicated resilience scheme nor to they propose a scalable protocol. Saito *et al* [SAI00] address MP2P LSP resilience by proposing the use of complete back-up trees. However, this method relies on signalling travelling back to the ingress in order to switch from the failed tree to the back up tree. What is clearly needed is a scheme that is dynamic and scalable in its creation of MP2P LSPs and resilient in the face of network failure. The remainder of this thesis is dedicated to presenting a solution that is practical, scalable and robust, in that a complete protocol is proposed for the dynamic online creation and maintenance of MP2P LSPs. This protocol is called DAM and it addresses the issue of resilience using a structured localised reaction to failure.

Chapter 3

3 Dynamically Adaptive Multipoint to Point (DAM)

This Chapter presents the **D**ynamically **A**daptive **M**ultipoint-to-point (**DAM**) scheme and its associated protocol. DAM is a supporting protocol or scheme designed specifically to complement MPLS. It does so by providing a mechanism to set up and maintain MP2P LSPs in a dynamic decentralised fashion. Moreover, it is an MPLS Traffic Engineering (TE) solution that includes built in resilience that can lead to fast, localised healing in the event of link or node failure.

3.1 Introduction

DAM is characterised by the manner in which MP2P LSPs are formed and subsequently maintained; particularly in the face of network disruption. DAM is able to provide a fast-localised reaction in the face of network disruption by taking advantage of Link Reversal Routing (LRR) algorithms [PER00] originally designed for use in ad-hoc networks. The formation of MP2P LSPs is undertaken by one of two methods using either an *edge orientated* approach or a *fully distributed* methodology. The edge-orientated approach differs from the distributed approach in that calculation of the initial tree is pushed to the edge of the domain reducing the processing overhead within the core. The calculation that takes place at the edge of the network takes the form of an adapted Dijkstra calculation [DIJ59] that takes the cost of the reverse path as a parameter.

3.2 Creating the MP2P tree

This section describes both the distributed and edge oriented methods used by DAM in the formation of the MP2P tree. What is common to both methods is that nodes require access to a Link State Database. This should not present a problem within an MPLS domain as RFC 1812, “Requirements for IPv4 Routers” [RFC1812], states that a router that implements a routing protocol must implement OSPF and may implement other additional IGPs, making OSPF a mandatory requirement for routers using dynamic routing protocols.

3.2.1 The Distributed Model - Inverted Multicast Trees

One of the concepts upon which DAM is based is that a MP2P tree is a special case of a Point-to-Multipoint (P2MP) tree. Simply put, a MP2P tree can be considered to be an inverted P2MP tree. The formation of P2MP trees as part of multicast routing protocols is well established and in the fully distributed version of DAM this is used in the creation of MP2P trees.

In multicast routing, the creation of the multicast delivery tree is undertaken by a multicast forwarding algorithm. There are two main algorithms that can perform such a task: Reverse Path Forwarding (RPF) [IBM01] and the Centre Based Tree algorithm (CBT) [FOR03]. The majority of multicast routing protocols are based on either of these two, with the exception of Protocol Independent Multicast-Sparse Mode (PIM-SM) [RFC2362] that operates with either algorithm and Multicast OSPF (MOSPF) [RFC1584] that uses the OSPF Link-State database. In CBT, a centre point is chosen in the multicast group and each recipient sends a *join request* to that point, the result is a shared tree for that multicast group. There exists a many-to-one mapping of sources to a particular group, the tree is identical for all sources so the resulting topology is a Multipoint-to-Multipoint (MP2MP) tree or shared tree. In contrast to this, RPF builds a single tree for each source, there is a one to one mapping between each source and an associated group of destinations resulting in a P2MP tree or source tree. It is clear that RPF, in its ability to create a source tree, clearly lends itself to being adapted and used to create MP2P trees. The RPF algorithm uses a *flood and prune* mechanism to create the tree. In [RFC3353] the flood and prune mechanism has been identified as generating volatile tree structures when using its traditional multicast functionality with MPLS. However, in the context of DAM, this is not believed to be an issue, as the volatility is due to the non-static nature of a multicast group such that the flood and prune is an ongoing phenomenon. In the proposed scheme, the flood and prune mechanism is used sparingly whereby floods are infrequent & periodic; and the prune mechanism is used only in a localised and structured fashion, as shall be explained.

The tree is initially created using RPF; the algorithm works as follows: trees are calculated and updated dynamically. The algorithm maintains a reverse path table used to reach each source. In the DAM application of the algorithm the source in this table eventually becomes the sink in the MP2P configuration. This table maps every node to the preferred interface used to reach this sink. The preferred interface is simply the interface that is part of the shortest path tree, calculated locally. During the initial flood process, if the datagram arrives through the preferred interface used to transmit datagrams back to the sink, the datagram is forwarded through every

appropriate upstream interface. Otherwise, if the datagram has arrived through a sub-optimal path, it is subsequently discarded. Using this process, duplicate packets caused by network loops are filtered out. This corresponds to the flood part of the flood and prune process used in multicast protocols; the prune simply involves the pruning of branches of the tree for nodes which do not wish to receive data from a multicast group and in the DAM application do not require to send data to a particular sink node. The manner in which the preferred interface is chosen directly impacts on how optimal the tree is. For example, if used in conjunction with a routing protocol like OSPF, then the preferred interface as a result of Dijkstra's algorithm is found to be on the shortest path leading to the least cost delivery tree. RSVP-TE [RFC3209] could be used to traffic engineer constrained paths forming a sub-optimal tree where required. Unlike the scheme proposed in [SAI00], where the MP2P LSPs are pre-assigned and are created on the basis of network topology, the DAM distributed scheme forms the MP2P dynamically on the basis of preferred interfaces. This can allow for adaptive tree growth based on chosen criteria, meaning that at each node the preferred interface is calculated and marked locally. Each node may require access to a Link State Database where a conventional shortest path calculation takes place using Dijkstra's algorithm. Alternatively, some form of Constrained Shortest Path First (CSPF) calculation [LAK06] may be used in order to mark the preferred interface as an entry in the reverse path table used by RPF, in a CSPF calculation undesirable links are removed from the SPF calculation based on constraints such as bandwidth requirements [LAK06].

3.2.2 The Edge Orientated Model

In the edge-orientated formation of the MP2P tree, Dijkstra's algorithm is run in reverse at the edge of the domain. Egress nodes run a reverse shortest path calculation to calculate the MP2P tree; the routes in this tree are then source-routed from the destination thus the calculation is only performed once at the egress. The adapted version described here uses the cost of the reverse path as opposed to the forward path employed in a conventional Dijkstra calculation; this is the only change made to the original algorithm. An adapted version of Dijkstra's algorithm [DIJ59] is now described; this is based on the description of Dijkstra's algorithm given in [DOY98]. The adapted version of Dijkstra's algorithm operates as follows: Links are represented by a triple {Node ID, Neighbour ID, Cost from neighbour to the node}

a) The egress node initialises itself as the root of the tree in the Tree Database

b) All triples in the Link State Database describing links from the neighbours of the egress to the egress are added to the Candidate Database.

c) The cost to the egress from each link in the Candidate Database is calculated. The link in the Candidate Database with the lowest cost is moved to the Tree Database, if there exists more than one link with an equally lowest cost to the egress choose one and move it to the Tree Database.

d) The Neighbour ID in the triple describing the link just added to the Tree Database is examined. If this node's links with its neighbours are not in the Candidate Database they are added to it (except links already in the Tree Database). If there exists entries with the same Neighbour ID in the Candidate database, purge all except those with the lowest cost to the egress.

e) If there are links in the Candidate Database return to step c). If the Candidate Database is empty stop running the calculation. Each participating node should appear once as a Neighbour ID in the Tree Database.

An example showing this adapted Dijkstra calculation for the reverse shortest path is given below:

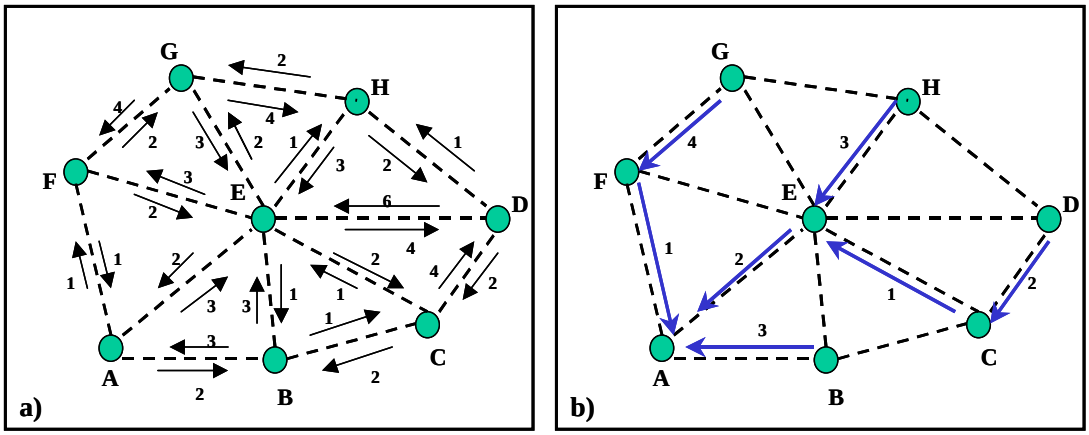


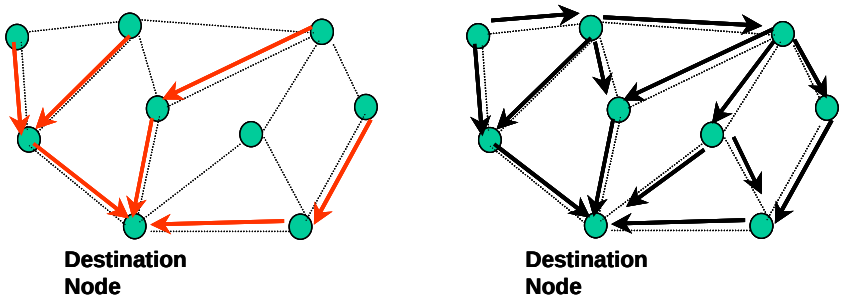
Figure 12:- Example of reverse shortest path calculation

Figure 12 a) shows the network with the cost of all links in all both directions, figure 12 b) shows the calculated tree with node A as the root. The calculation of figure 12 b) is given below

Candidate Database (CD)	Cost back to root	Tree Database	Description
		{A, A, 0}	Egress Node 'A' adds itself as the root in Tree Database (TD)
{A, B, 3} {A, E, 2} {A, F, 1}	3 2 1	{A, A, 0} {A, F, 1}	The links to all 'A' s neighbours are added to the CD, Link {A, F, 1} is identified as having the lowest cost back to the root and is moved to the TD
{A, B, 3} {A, E, 2} {F, G, 4} ~~{F, E, 3}~~	3 2 5 4	{A, A, 0} {A, F, 1} {A, E, 2}	The links to all 'F' s neighbours are added to the CD, Link {A, E, 2} is identified as having the lowest cost back to the root and is moved to the TD. In CD {F, E, 3} is purged from the list as {A, E, 2} has a lower cost back to the root
{A, B, 3} {F, G, 4} ~~{E, B, 3}~~ {E, C, 1} {E, D, 6} {E, H, 3} {E, G, 3}	3 5 5 3 8 5 5	{A, A, 0} {A, F, 1} {A, E, 2} {A, B, 3}	The links to all 'E' s neighbours that are not in the TD are added to the CD, Link {A, B, 3} is identified as having the lowest cost back to the root and is moved to the TD. {E, B, 3} is purged from the CD list
{F, G, 4} {E, C, 1} {E, D, 6} {E, H, 3} {E, G, 3} ~~{B, C, 2}~~	5 3 8 5 5 5	{A, A, 0} {A, F, 1} {A, E, 2} {A, B, 3} {E, C, 1}	The links to all 'B' s neighbours that are not in the TD are added to the CD, Link {E, C, 1} is identified as having the lowest cost back to the root and is moved to the TD. {B, C, 2} is purged from the CD list
{F, G, 4} ~~{E, D, 6}~~ {E, H, 3} ~~{E, C, 3}~~ {C, D, 2}	5 8 5 5 5	{A, A, 0} {A, F, 1} {A, E, 2} {A, B, 3} {E, C, 1} {F, G, 4}	The links to all 'C' s neighbours that are not in the TD are added to the CD, Link {F, G, 4} is chosen from the links having the lowest cost back to the root and is moved to the TD. {E, D, 6} and {E, G, 3} are purged from the CD list
{E, H, 3} {C, D, 2} ~~{G, H, 2}~~	5 5 7	{A, A, 0} {A, F, 1} {A, E, 2} {A, B, 3} {E, C, 1} {F, G, 4} {E, H, 3}	The links to all 'G' s neighbours that are not in the TD are added to the CD, Link {E, H, 3} is chosen from the links having the lowest cost back to the root and is moved to the TD. {G, H, 2} is purged from the CD list
{C, D, 2} ~~{H, D, 1}~~	5 6	{A, A, 0} {A, F, 1} {A, E, 2} {A, B, 3} {E, C, 1} {F, G, 4} {E, H, 3} {C, D, 2}	The links to all 'H' s neighbours that are not in the TD are added to the CD, Link {C, D, 2} is identified as having the lowest cost back to the root and is moved to the TD. {H, D, 1} is purged from the CD list. The CD is now empty and the tree shown in figure 12 b) has been calculated

3.2.3 Key Points in the Creation of MP2P LSPs

Whether fully distributed or edge orientated, there are a number of key steps that are undertaken in the set-up phase of the MP2P LSP. This phase has two primary objectives, a) the calculation and set-up of the MP2P LSP for a particular egress node and b) the initialisation and establishment of a Directed Acyclic Graph (DAG) for the same egress node, as shown in figure 13.



MP2P LSP Formed Through the Use of RPF or Edge Orientated Source Routing

Corresponding DAG.

Figure 13:- Forming the MP2P Tree

The steps undertaken to achieve these objectives are as follows:

1. At start up, routers use appropriate signalling in order to achieve converged link state database.
2. The destination node calculates a reverse shortest path tree using adapted Dijkstra calculation based on information gleaned from the link state database.
3. RPF flooding process (or the source-routed process in the edge orientated model) is initiated from the destination node (egress node).
4. At the same time as 3. As part of maintaining a DAG using LRR partial reversal method the destination advertises its height of zero to its neighbours. This acts as a trigger for 5.
5. Node receives height of neighbour for a particular destination.
    - The node checks to see if it has an entry for this destination, if the node has an entry it checks if it is the same. If it is the same the node does IP address comparison and adjusts and re-advertises its new height
    - If the node's own entry is higher or lower than the received advertised height it does nothing

- If the node does not have a height entry for this particular destination, it adopts the (received height + 1) and advertises it to its neighbours.

6. Label bindings are piggybacked on DAM signalling messages (RPF or source-routed) in order to form the MP2P LSP

In step 5, if a node receives a height advertisement and finds it already has set its height entry for that destination, the node compares the advertised height to its own. If this height is the same, the IP address comparison checks to see which router has the highest IP address value for the contested height. The router with highest IP address keeps the contested height. The node with the lower IP address is forced to adjust its height and re-advertise.

Once these steps have been undertaken for a particular destination, a MP2P LSP is set up along with its associated DAG.

3.3 MP2P Maintenance

This section describes the method taken to maintain the MP2P LSP once it has been set up. It also describes how a node that is not part of a particular tree initiates the process to join a tree in a bid to send data to a particular egress.

3.3.1 Maintenance in the Distributed Model

The maintenance mechanism in the distributed version of DAM implements fast reactive healing of the tree. At each node, running alongside RPF, a Link Reversal Routing (LRR) algorithm, normally used in ad-hoc networking, is also run. In this scheme, the height-based version of the Partial Reversal method is used. A Directed Acyclic Graph (DAG) is formed running from all nodes to the sink node. This is a loop-free graph of multi-path routes to a particular sink node. A "height" metric is used to establish the DAG; links between nodes are assigned a direction based on the relative height metric of neighbouring nodes, i.e. the direction points towards a neighbouring node of a lesser relative height. When a node (except the destination node) loses its last outgoing (downstream) link it generates a new reference level. The affected node's new height triggers a localised structured reaction to the failure, resulting in the reversal of one or more of its incoming links (refer to Section 2.6 on LRR for a detailed explanation). This takes place until all nodes have a route to the sink node and the DAG is again complete. As part of this scheme, a distributed check of the preferred interface in the reverse path table used by RPF is made against the DAG. If the preferred interface is not one that corresponds to one of the multi-paths on the DAG, a change is made to the affected area re-establishing its connection to the sink and subsequently healing the MP2P tree. Note that this path re-establishment does not involve the re-invoking of the RPF flood process. Changes are only made to the affected nodes. The revised path after a failure is mapped on the DAG following the links with lowest height (lowest reference level) back on to the tree. Moreover, the comparison done between the reverse path table and the DAG ensures that the path re-establishment takes place in a structured way by following links marked with the lowest height on the DAG.

The result of many failures can lead to a deformed, sub-optimal tree. To counter this, a periodic RPF flood is used to form a new tree. A time of thirty minutes between floods is recommended but this time is readily configurable. In order to avoid network disruption during the course of,

and immediately after, a periodic flood, the existing tree is not discarded until the new tree is fully established i.e. the branches of the tree have extended to the last upstream nodes (the ingress nodes). This is a gradual prune process; as the paths belonging to the new tree establish connections to the sink, their respective paths in the old tree wither away. To avoid isolation of nodes from the sink because a downstream node has cut a branch still in use by the old tree, a node can only prune itself from a tree once it has lost its last incoming path. Otherwise it assumes other nodes are still using that branch and it does nothing. The last upstream node (normally an ingress node) in any branch always initiates this gradual pruning process.

3.3.2 Maintenance in the Edge Orientated Model

The mechanism employed for maintenance in the edge orientated model differs from that employed in the fully distributed version. Nevertheless, fast reactive healing is again the central objective in this operation. As in the fully distributed version, an LRR algorithm is employed, again the height-based version of the partial reversal method is used as part of the strategy to provide fast, reactive, localised healing of a tree.

Once the MP2P tree has been established along with its respective DAG the maintenance mechanism employs a *heartbeat* that acts as a keep-alive message for a particular tree, naturally each tree has a unique ID associated with it that is destination unique. This tree ID is transmitted within the heartbeat along with a sequence number. The destination node initially disseminates this tree ID in the initial set up of the tree. In the maintenance phase, it periodically sends out the heartbeat message to its neighbours participating in the tree. Intermediate nodes then forward this message on to other nodes participating in the tree. The reason the heartbeat message is not source-routed is that, as a consequence of maintenance or indeed the joining of a new node that did not exist in the initial set up phase, a source-routed message may be rendered outdated. In other words, it may be based on a tree topology that is no longer valid as a consequence of revision. A tree is thus considered to be alive by nodes in the tree upon timely receipt of these *heartbeat* messages.

The *heartbeat* message has two roles within this scheme. As well as signifying that a particular tree is still valid (alive) it acts as a trigger for the resilience mechanism. It is thus necessary to employ two different periods of timeout in order to take these two roles into account. A maintenance timeout of 180 seconds is used as a trigger to find a new route on to the tree. A

longer period of 10 minutes without heartbeat messages is used to act as a tree *dead time*. If a node does not receive a heartbeat message for a particular tree within 180 seconds it enters the tree maintenance phase whereby the actions necessary for path re-establishment take place. After a period of ten minutes if the appropriate heartbeat message has not been received it considers the tree to be dead and makes no further effort to re establish a connection with the tree. Nodes maintain information on a particular tree for a further 20 minutes during which the tree can be reactivated by the receipt of a heartbeat message. Every 30 minutes the destination recalculates its reverse shortest path tree and initiates, through appropriate signalling, the establishment of a new tree and respective DAG. Note that each new tree has a unique tree ID.

In the event of a failure, the healing of the tree is undertaken in a localised structured fashion. There are two possible scenarios that are considered at a node where the resilience mechanism has been activated.

In the first case, a node realises that it has lost its link to its current downstream neighbour in the tree. The trigger for this can be either the 180-second heartbeat timeout or signalling that maintains the DAG through periodic transmission of height data between neighbours (hardware notification, if present, may also be used as a trigger). This lost link, however, is not its last outgoing link so there is no requirement to invoke the LRR algorithm to re-establish the DAG. In such a case, the tree is healed by a directed search back on to the tree using the heights in the DAG as a guide. The affected node elects to re-map the affected branch back on to the tree by signalling to its downstream neighbour with the lowest height on the DAG. The neighbour with the lowest height then extends this search by signalling to its neighbour with the lowest height in the same DAG. This search propagates though the elected nodes in the DAG until a path has been established back on to the tree. As shall be subsequently seen this process is very similar to the joining of a tree by a new node.

In the second case, the affected node has not only lost its current link to its downstream neighbour in the tree but this link is also this node’s last outgoing link in the respective DAG. As a consequence, before a path back onto the tree can be established from that node, the LRR algorithm is invoked in order to re-establish the DAG. Once the partial reversal method has finished calculating the affected node’s new height, and after link reversal new outgoing links have been established in the DAG, the affected node then does the directed search for a path back onto the tree. This takes place in the same fashion as in the first case.

3.3.3 Joining an Existing Tree

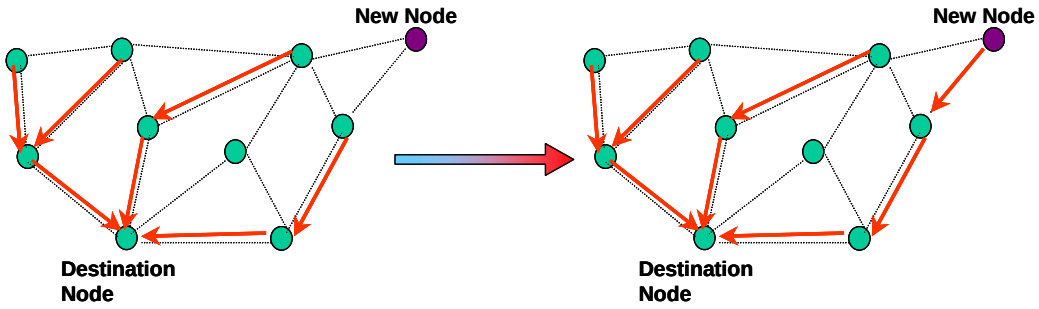


Figure 14:- New node joining an existing tree

In joining an existing tree, a new node has no knowledge of its neighbours' height or indeed if any are already part of the tree, so it must be proactive in obtaining such information. When a new node joins a network, the node completes its link state database via link state routing. Once the node identifies a destination it wishes to send data to, the node queries its neighbours for their respective heights with regard to this destination. If a neighbour is already part of the tree, the node adopts the LSP from the neighbour with the lowest height as shown in figure 14. The label binding lasts until the next periodic tree re-establishment or until maintenance takes place as a result of a failure.

If no neighbour is a member of the tree but replies with a height for the particular destination, the neighbour with the lowest height is sent a *query_propagate* message in order to query its downstream neighbours in the same manner until a node is reached on the DAG that is a member of the tree. The *query_propogate* is effectively a directed request for a label from upstream neighbours in the DAG. This process is the same as that employed in the maintenance phase described in the previous section. The new node maps the new branch onto the tree by signalling to its downstream neighbour with the lowest height on the DAG. The neighbour with the lowest height then extends this search by signalling to its neighbour with the lowest height in the same DAG. This search propagates though the elected nodes in the DAG until a new path has been established onto the tree.

If a new node queries and does not receive any height replies, meaning no neighbours are on the DAG, it queries again until such time that one of its neighbours is a member of the DAG giving direct or indirect access to the tree.

3.4 Example Operation

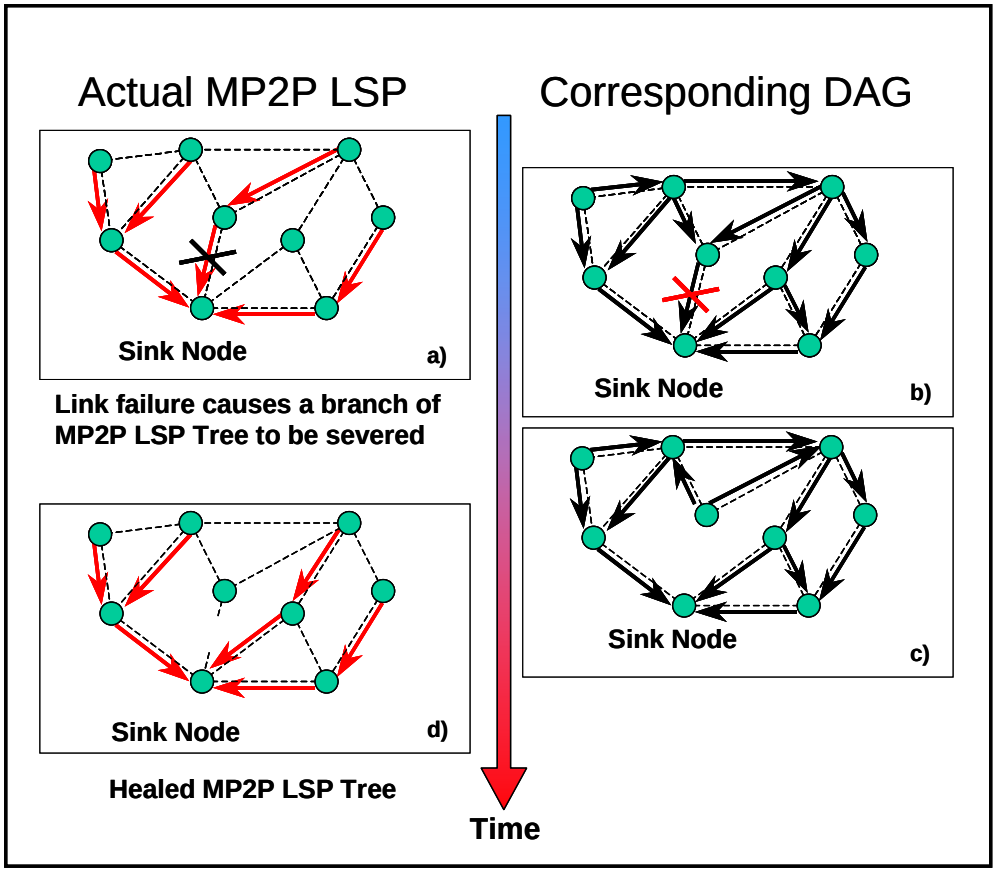


Figure 15:- Healing a MP2P tree after a failure

Figure 15 shows an example of how a MP2P LSP and its corresponding DAG are affected by the actions of DAM as a result of a failure. The example shows the major aspects of the protocol as they impact on the MP2P LSP and DAG in order to heal the affected tree. In figure 15 a) a link failure causes a branch of the MP2P LSP tree to be severed. Regardless of whether the edge orientated or distributed version of DAM is being run the link failure acts as a trigger, invoking the resilience mechanism to act. Central to DAM is the maintenance of a corresponding DAG for any particular tree. In figure 15 b) the link failure results in a node losing its last outgoing link and the partial reversal method taking place at the affected node. This results in link reversal actions in order to re-establish the DAG as shown in figure 15 c). Note that in this particular example the re-establishment of the DAG was not necessary for the affected ingress node to re-establish a path to the egress node, as it had not lost its last outgoing link. DAG maintenance took place as a matter of course because of the affected intermediate node in the

centre of the network. In figure 15 d) the MP2P tree has been healed with the establishment of a new branch by the affected ingress back to the egress node. This is the result of a directed search along the DAG in order to find a new path back onto the tree.

3.5 Summary

This chapter has described the basic principles that make up the DAM scheme and protocol. Moreover, it describes two versions of DAM; these are the distributed version and the edge-orientated variant. Common to both versions are a number of design goals that have been identified for the purpose of addressing issues that will be considered subsequently. These goals require that MP2P LSPs are created and maintained dynamically whist providing a resilience procedure that can lead to fast localised healing.

Both schemes leverage the partial reversal method in their respective resilience strategies and both schemes require the existence of a link state routing protocol in order to have access to a link state database. The distributed version of DAM uses the Reverse Path Forwarding algorithm in its creation of the MP2P LSPs. The Reverse Path Forwarding algorithm is used by multicast routing protocols in creating a source tree or P2MP tree. In the distributed version of DAM, the manner in which the preferred interface is marked in the reverse path table directly impacts on the form a MP2P LSP will take in any given topology. In the edge-orientated version of DAM, processing overheads in the core of the domain are reduced by pushing MP2P tree calculation to the edge of the domain. This is done by using an adapted version of Dijkstra's algorithm and source routing the tree by including it in signalling used in the initialisation phase of the protocol. This chapter has also described the adapted version of the Dijkstra's algorithm used to calculate the reverse shortest path first tree. Such a calculation can be employed by both versions of DAM either once at the edge of the domain or at each node in determining the preferred interface in the reverse path table. MP2P creation has been discussed in the context of both versions of DAM. In addition to this, MP2P maintenance has been discussed. Although the creation and maintenance of MP2P LSPs is different in the distributed and edge orientated versions of DAM, the basic principles are common to both versions in that each MP2P LSP has a corresponding DAG which is used to maintain and provide resilience for the MP2P LSP. This chapter has also discussed how a new node may join an existing tree and moreover how the heights in the DAG are used as a guide for this joining and indeed the healing of a tree.

Chapter 4

4 DAM – The Protocol – Implementation

This chapter lays out the specifics of the DAM protocol as well as discussing implementation issues. The focus here is on the edge-orientated version of DAM. Even so the specification here is easily adapted for an implementation of the fully distributed version. The edge-orientated version was chosen over the fully distributed version because the formation of the tree is simpler, processing in the core of the network is reduced by conducting the shortest path first calculation only once at the egress node as opposed to at all nodes in the tree.

4.1 DAM Protocol

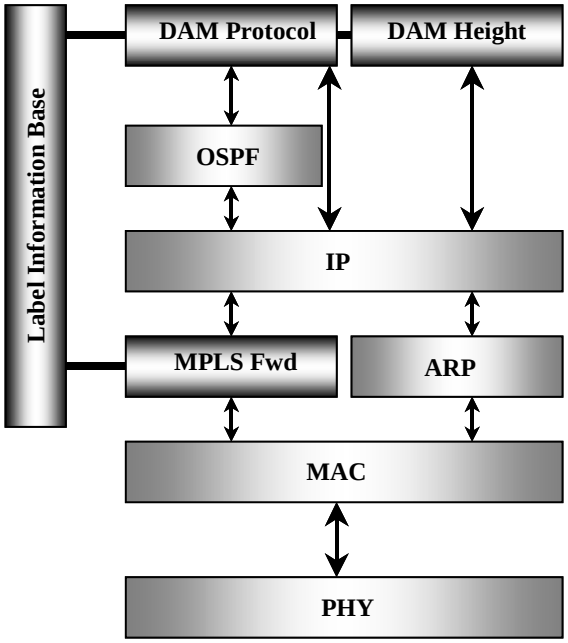


Figure 16:- Architecture showing DAM in relation other protocols

The DAM protocol comprises two parts: the *main protocol* and the *sub protocol* that is the DAM Height protocol as shown in figure 16. The main protocol is referred to as the DAM protocol and is described here along with its interactions with its sub protocol. The main purpose of this protocol is the calculation and establishment of MP2P LSPs. This is achieved

through the process of label distribution within the context of MPLS. Two message types are defined; these are the *HeartBeat* message and the *QueryPropagate* message.

The *Heartbeat* Message has two key uses; it is used by the DAM protocol as a keep alive for a particular tree (the MP2P LSPs are soft state and require refreshing periodically) and also used to disseminate label bindings upstream to all nodes in the tree. The initial heartbeat message for any given tree is also used to carry the source-routed list that is calculated at the egress.

The *QueryPropagate* message is used by the DAM protocol as a directed request for a label binding. Each node also maintains a list of trees that it is part of. This list comprises the Tree ID, which is made up of a unique number together with the ID of the root node (egress IP address), along with other information such as the Tree ID of the tree it is replacing and the current status of the tree. The following is an operational overview of the DAM protocol in terms of message exchanges and its interaction with the DAM Height sub protocol.

4.1.1 Interaction using the Heartbeat Message

If the node is an egress node it calculates the initial MP2P tree using a reverse Dijkstra calculation and then does the following:

- The node sends an initial heartbeat message to its one-hop neighbours. This initial heartbeat message has the newly calculated tree encoded in it along with a Tree ID and label binding.

- The node invokes the DAM Height Protocol, which sends a height advertisement to all its one hop neighbours. This is done in order to form a corresponding DAG for any given tree (a more detailed description is given in the description of the DAM Height protocol).

- Every 60 seconds the egress node sends a heartbeat message to all its upstream neighbours in the MP2P LSP. ONLY the initial *Heartbeat* carries the source-routed list.

The egress maintains a tree in this manner for a period of 30 minutes after which a replacement tree is calculated. The egress sends *Heartbeat* messages that relate to this replacement tree. This

includes the initial messages carrying the newly calculated source routed list. This is done to ensure the replacement of sub optimal trees that may have arisen if during any given period a tree has undergone localised healing a large number of times. Once a replacement tree has been calculated the egress node continues to send *Heartbeat* messages for the old tree for a period known as the *Tree Purge Time*, after which it ceases to refresh the old tree. During this period both the new and old tree are supported by the egress in order to ensure connectivity is maintained whilst a tree is being replaced.

If an intermediate or ingress node has received an initial *Heartbeat* message from one of its downstream neighbours the following steps take place:

- If it is an intermediate node it examines the received *Heartbeat* message and determines from the source-routed list encoded into this initial message which of its neighbours form part of the tree. It then encodes its own label into this initial *Heartbeat* message before sending it on to its neighbours that are upstream with respect to this MP2P LSP.

- If it is an intermediate node it then adds the label in the received *Heartbeat* to its Next hop Label forwarding Entries (NHLFE) and its own label transmitted in the revised initial *Heartbeat* to its Incoming Label Map (ILM).

- If the node is an ingress node it then adds the label in the received *Heartbeat* to its NHLFE and makes any necessary adjustments to its FEC-to-NHLFE map (FTN).

Subsequent *Heartbeat* messages are revised by intermediate nodes to carry local label bindings and sent on. Moreover, each node that receives a *Heartbeat* message then sends on its revised version of this message to its upstream neighbours until this message has propagated to all the ingress nodes in the MP2P LSP. The revised *Heartbeat* messages have the local label bindings encoded within them acting as a refresh mechanism for the MP2P LSP. In this way, any changes in topology that are not yet known to the egress are masked from it, regardless of any localised healing that takes place as a result of a failure. It is important to note that timers such as the time between *Heartbeat* messages are readily configurable to suit particular operator preferences.

4.1.2 Interaction using the QueryPropagate Message

The *QueryPropagate* message is used by the DAM protocol as a directed request for a label binding either as part of the localised healing process after a failure or when a new node wishes to join an existing tree. A node receiving a *QueryPropagate* message acts in the following way:

- The node receiving the *QueryPropagate* message is part of the MP2P LSP with the corresponding tree ID in this message, in which case it adds the node that sent this message to its list of upstream nodes in this MP2P LSP. As a consequence of this, the node sends a *HeartBeat* message with label binding to its neighbour that sent the *QueryPropagate* message.

- The receiving node is not part of the MP2P LSP with the corresponding tree ID in this message, in which case it invokes the DAM Height protocol so as to obtain its neighbours height for the DAG corresponding to the tree in question. The node then periodically sends a query propagate message to its downstream neighbour in the DAG with the lowest height until it receives a *Heartbeat* message. Upon receiving this *Heartbeat* message the node appends it and sends it to its upstream neighbour from which it received the *QueryPropagate* message. This process continues until the node that sent the initial *QueryPropagate* message has received a *Heartbeat* message with label binding resulting in either a successful join on to an existing tree or a successful localised healing.

4.2 DAM Height Protocol

In order to establish and maintain a DAG for any particular egress node a sub-protocol has been designed to act within DAM. This sub protocol is referred to as the DAM Height protocol; its position relative to other protocols in the architecture is depicted in figure 16. The purpose of this protocol is to disseminate height information between neighbours whilst implicitly indicating the presence of an active link between them. By doing so the protocol is able to deduce when the Partial Reversal Method must be run at a node for a particular DAG. The Partial Reversal Method may be invoked either because a link is assumed to be dead, because a periodic height advertisement has not been received, or, where available, hardware notification of a link failure. The DAM Height protocol can be defined in terms of its message types and how they are used. The DAM Height protocol uses two messages; these are the *HeightAdvertisement* message and the *QueryHeight* message.

The *HeightAdvertisement* message is used to disseminate height information between one-hop neighbours for the purpose of forming and maintaining a DAG. It also signifies implicitly the loss of a link as height advertisements are sent between adjacent neighbours on a frequent periodic basis, similar to the OSPF Hello message.

The *QueryHeight* message is used in response to a null height advertisement from a neighbour or when a new node wishes to join an existing tree. This message is sent as a prompt to a neighbour to send a height advertisement with respect to a particular DAG.

The following is an operational overview of the DAM Height protocol in terms of its messages and its interaction with the DAM protocol.

4.2.1 Interaction using the HeightAdvertisement Message

The first phase of this protocol involves the initialisation of a DAG with respect to a specific node. This involves creating a new instance of the height database where the heights of a node's neighbours are recorded along with the node's own height for a particular DAG. A node can only carry out this initialisation if it is the egress node or if it has received a height advertisement from one of its neighbours. This ensures that the DAG is instantiated by the egress (which is the root node) and is subsequently grown from this root by height

advertisements. Encoded in a height advertisement is the unique Tree ID identifying the MP2P tree that the DAG is associated with. A node sends its height to all of its one-hop neighbours in a height advertisement every 15 seconds by default.

In this initial DAG creation stage, the following rules apply so that no two neighbours in the DAG can adopt the same height:

1. The Egress node sends a height advertisement of zero {0, 0, EgressID} to all its one-hop neighbours.
2. Upon receipt of the first height advertisement for a DAG that is unknown to the node, it instantiates a new height database. It adds the received height advertisement to this height database. It adds 1 to this received height and enters the result as its height into the height database. It then advertises its own height to all its one-hop neighbours including the neighbour that sent it a height advertisement.
3. Once a node advertises its height for the first time, if a neighbour replies with the same height this indicates that this neighbour advertised its height for the DAG before it had received the nodes height advertisement. In such a case, an IP address comparison takes place and the node with the numerically highest IP address wins the contested height. The node with the numerically lower IP address is forced to recalculate and re-advertise its height. The rules for calculating the height are as follows:
    - Before a node makes its first height advertisement to its neighbours it adopts the [highest received height + 1] in the set of height advertisements it has received (that may be one or more depending on the location of a node in any given topology as it is possible to receive height advertisements for a DAG on multiple interfaces).
    - If, as a result, a node receives an equal height advertisement from a node not already included in its height database it must do the IP address comparison described above.
    - Once a node has received a height advertisement from all its one-hop neighbours the DAG initialisation phase is considered complete. Neighbours that do not reply within the height *dead interval* are not included in the DAG, future height advertisements from such nodes are replied to with a null height (whereby the node receiving the null reply must send a QueryHeight message back to the node sending the null height in order to get a height for a particular DAG and ensure the DAG remains acyclic).

Once a DAG has been set up to support a particular MP2P LSP there are one of two possibilities:

1. A node receives a height advertisement from one of its neighbours with a new height different to that held in its current height database. This indicates that the Partial Reversal Method has been used at this neighbour. In response, the Partial Reversal Method is invoked at this node, whereby if this received height signifies the loss of the node's last outgoing link a new height is calculated and advertised invoking the necessary link reversal(s) to maintain the DAG. If this is not the case it simply enters this new received height into its appropriate height database.

2. If a node has not received a height advertisement from one of its neighbours within the height *dead interval* (60 seconds) it presumes that its link with this neighbour is down (dead). Having made this presumption, it proceeds to use the Partial Reversal Method. Link failure notification from the hardware may also trigger this behaviour. The height of the neighbour that is down is marked as null in the height database until such times that it does receive a height advertisement. Should a neighbour that is down reply with a height that is the same as the node's height, a null height advertisement is sent to that node forcing that neighbour to query and re-advertise a new height.

4.2.2 Interaction using the QueryHeight Message

The *QueryHeight* message is employed either in response to a null height advertisement from a neighbour or when a new node wishes to join an existing tree. Having received a *QueryHeight* message, a node replies with a height advertisement. The following rules apply:

- The node sending the *QueryHeight* message examines the received height in the reply. If this received height is not the same as its own then it replies with its own height. If this received height is the same as its own it takes the following action:
    1. If the node is an ingress node it adds the received height to its height database and adopts the [highest height + 1] in this set, ensuring that it is a local maximum in that all its links in the DAG flow out from it.
    2. If the link is an intermediate one, it adds the received height to its height database and adopts a height such that it has at least one incoming and one outgoing link in the DAG.

In order to choose a height that meets these requirements the height database is examined. The highest height is chosen in this set and one is subtracted until a height is reached that is not in the existing set whilst satisfying the condition: Lowest Height < Calculated Height < Highest Height.

As with the main protocol timers, timers in the DAM Height protocol are readily configurable to suit particular operator requirements. However, it is important to note that the frequency of messages used to maintain the DAG should be greater than those used in the main protocol in order to ensure that the MP2P LSP can be healed using a suitably reconstituted DAG.

4.3 Implementation

This section includes the methodology employed in transforming the specification of both the DAM protocol and the DAM Height protocol as described in sections 4.1 and 4.2 respectively into a working simulator. This simulator has been implemented and validated in OPNET. The following data structures have been defined in the DAM protocol:

```
/* used for keeping track of trees */

typedef struct
      {
      IpT_Address       egress_addr;
      unsigned int treeNo :32;
      unsigned int PtreeNo :32;  /*tree no of previous tree i.e the
      one being replaced*/
      double crtime;           //creation time
      int T_D_status;         /*set to 1 for MntTimeout and set to 2
                              if tree is considered dead */
      int Purge_T_status; /*set to 1 if tree has been replaced and
      will be suported for the purge time*/
      double Last_HB_Time;       /*time last HB was recieved if
      ingress or intermediate node / set to zero if this is the egrees
      and not used */
      unsigned int label : 20;      // label used for this tree
      IpT_Address upstrm_nbour[8]; /*upstream neighbours in this tree-
      [8]for eight intf */
      } tree_entry;
```

Each node maintains a tree database that is implemented as a list of tree_entry elements defined in the structure type shown above. This list is managed and maintained by the main protocol. The DAM protocol maintains two other lists that make up the remainder of the Label Information Base these are the NHLFE and FTN lists defined below.

```
/* NHLFE    */
typedef struct
      {
      int in_intf;
      unsigned int in_label : 20;

      int operation; /* 1 is swap label - 2 is strip label and deliver
      using IP */
      int out_intf;
      unsigned int out_label : 20;
      IpT_Address       egress_addr;
      unsigned int tree_no : 32;

      double cr_time; /*time NHLFE was created/updated*/

      } NHLFE;
```

```
/* FTN */

typedef struct
      {
      int in_intf;
      IpT_Address       dest_addr; /* destination longest prefix */
      int out_intf;
      unsigned int out_label : 20;
      } FTN;
```

In order to maintain the Label Information Base the DAM protocol uses the following message types. These are transmitted as packets between nodes that are encapsulated and delivered using IP.

Heartbeat Message: used by the DAM protocol as a keep alive for a tree also used to disseminate label bindings. Initial *Heartbeat* message carries source-routed list calculated at the egress.

```
/* HB Packet */
typedef struct
      {
      IpT_Address       egress_addr;
      unsigned int tree_no ;
      unsigned int ptree_no ;
      unsigned int init_hb ;
      unsigned int seq_no ;
      unsigned int label ;
      double cr_time;
      unsigned int reserved;
      } hb_header;

/*source routed list to be assigned to datablock in HB message */
typedef struct
      {
      IpT_Address       d_strm_addr;
      IpT_Address       u_strm_addr;
      } t_db_pair;

typedef struct
      {
      t_db_pair tree_list[60];
      } t_db_pair_struct;
```

Query Propagate Message: used by the DAM protocol as a directed request for a label binding for a particular MP2P tree.

```
/* QP packet */

typedef struct
      {
      IpT_Address       Source_addr;
      IpT_Address       egress_addr;
      unsigned int tree_no;
      unsigned int seq_no;
      unsigned int reserved;
      } qp_fields;
```

The DAM protocol specification was then decomposed into the following state transition table before being implemented as a process model within the OPNET modeller.

State	Event	Condition	Action	Final State
INIT	Begin simulation	Always	Carry out initial registrations Initialise relevant data structures Set interrupt for tree calculation if node is egress	IDLE
IDLE	Calculate tree interrupt	Timer expired	Reset timer	CalcTree
IDLE	Receive HeartBeat	Always		RecvHBeat
IDLE	Receive Qprop	Always		QueryProp
IDLE	Maintenance interrupt	MNT timer expired -180s	Mark tree status	DamJoin
IDLE	QProp interrupt	Timer expired and HB not received		DamJoin
IDLE	HeartBeat interrupt	Timer expired and is egress node	Resend HB to refresh tree	IDLE
IDLE	Tree Dead timeout	Expired timer (10 minutes)	Set Status in Tree database	IDLE

CalcTree	New Tree	Always	Send initial HB, Invoke DAM Height protocol	IDLE
CalcTree	Replacement Tree	Tree timeout	Send initial HB, Invoke DAM Height protocol, Set Purge timeout on old tree	IDLE
RecvHBeat	Initial Heartbeat message	Always	Replace Label, Send on revised HB to upstream neighbours	IDLE
RecvHBeat	Normal Heartbeat	Always	Reset timers and send on revised heartbeat	IDLE
RecvHBeat	Heartbeat in reply to a QProp message	Always	Update tree DB and remove source of this HB from upstream neighbour list if necessary	IDLE
QueryProp	Node part of tree	Always	Send label binding to source of QP and add this node to list of upstream neighbours	IDLE
QueryProp	Node not part of tree	Always	Store details of QP message	DamJoin
DamJoin	MNT / QP process	DAG exists and Tree status not MNT	Get relevant heights from DAM Height process, Send QueryProp message and set QP interrupt for some future time	IDLE
DamJoin	MNT / QP process	DAG does not exist	Set remote interrupt to DAM Height process in order to Query Height, set QP interrupt for some future time – 10s	IDLE

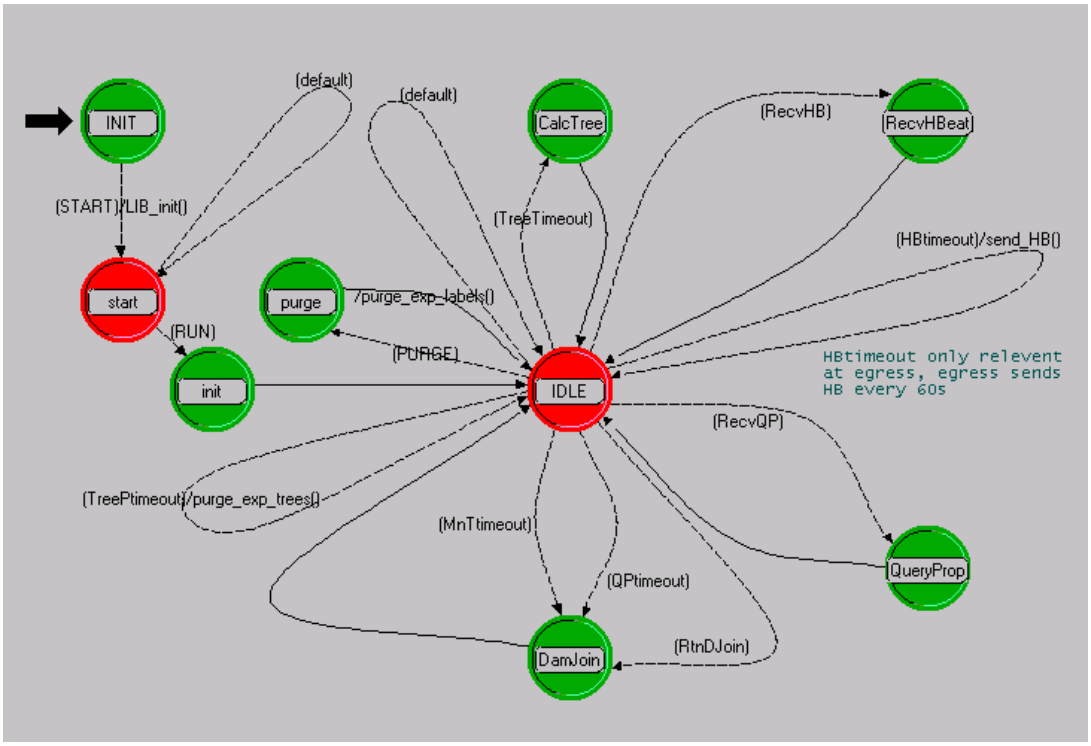


Figure 17: - The DAM Protocol

Figure 17. Shows the DAM protocol as implemented in OPNET.

The same steps were undertaken for the implementation of the **DAM Height** protocol. Central to the operation of this sub protocol is the maintenance of the DAG database that is implemented as a list made up of elements of the following type:

```
/* Structures relating to DAG */

typedef struct
      {
      IpT_Address       node_ID;
      int a;
      int   b;
      double Last_HA_Time;          /*Last time a Height AD was
                                    received*/
      } height;
```

```
typedef struct
      {
      IpT_Address       egress_addr;
      unsigned int treeNo :32;
      double crtime;          //creation time
      int D_status;           /*set to 0 for initial   status and 1
                              for maintenance status*/
      int QD_status;          /*set to 1 if a null HA has been
                              received*/
      int Purge_D_status;     /*set to 1 if DAG is to be purged this
                              is done remotely by DAM*/
      height my_height;
      height neighbour_H[8];
      }dag_entry;
```

A unique DAG entry is maintained for each DAG being supported. In order to maintain the DAG entry for a particular tree, the DAM Height protocol uses the following message types, as in the main protocol, these are transmitted as packets that are encapsulated and delivered using IP.

Height Advertisement Message: used by the DAM Height protocol to disseminate height information between one-hop neighbours for the purpose of forming and maintaining a DAG. This message type also signifies implicitly the loss of a link as height advertisements are sent on a frequent periodic basis.

```
/* HA packet */

typedef struct
      {
      IpT_Address       Source_addr; //z in height triple
      IpT_Address       egress_addr;
      unsigned int      tree_no ;
      int                     a; //alpha in height triple
      int                     b; //beta in height triple
      unsigned int      seq_no ;
      unsigned int      reserved;
      } ha_fields;
```

Query Height Message: used by the DAM Height protocol in response to a null height advertisement from a neighbour or when a new node wishes to join an existing tree. This message is sent as a prompt to a neighbour to send a height advertisement with respect to a particular DAG.

```
/* QH packet */

typedef struct
      {
      IpT_Address       Source_addr;
      IpT_Address       egress_addr;
      unsigned int tree_no ;
      unsigned int seq_no ;
      unsigned int reserved;
      } qh_fields;
```

The DAM Height protocol specification was then decomposed into the following state transition table before being implemented as a process model within the OPNET modeller.

State	Event	Condition	Action	Final State
INIT	Begin simulation	Always	Carry out initial registrations Initialise relevant data structures	IDLE
IDLE	New DAG remote interrupt from DAM	Always	None	NewDAG
IDLE	Receive HeightAD	Always		RecvHA
IDLE	Receive Query Height message	Always		RecvQH
IDLE	Height AD self interrupt	Timer expired -15s	Send Height to all neighbours Schedule self interrupt for next Height AD to neighbours	IDLE
IDLE	Height Dead interval expires – 60s	Timer expired, HA not received		H_Dead
IDLE	Remote interrupt from DAM process relating to joining unknown tree	Always		QueryHeight
IDLE	Remote interrupt from DAM to purge redundant DAG entry	Always	Remove redundant DAG entry from DAG DB	IDLE

NewDAG	Remote interrupt	Always	Create new Height database Send Height AD	IDLE
RecvHA	Receive height for unknown DAG	Always	Create new Height database Send Height AD	IDLE
RecvHA	Receive height for DAG equal to own height	Always	IP address comparison / resend height if required	IDLE
RecvHA	Receive Height AD from neighbour	Always	Reset timer Partial reversal method	IDLE
RecvHA	Receive Height AD from unlisted neighbour once DAG is maintenance status	Always	Reply with Null Height	IDLE
RecvHA	Receive Null Height AD from neighbour	Always	Reply with Query Height message	IDLE
RecvQH	Receive Query Height message	Always	Reply with height for particular DAG	IDLE
QueryHeight	Remote interrupt	Always	Send Query Height message to neighbours	IDLE
H_Dead	Periodic interrupt to check for dead links	Always	Partial reversal method Send revised height AD to neighbours	IDLE

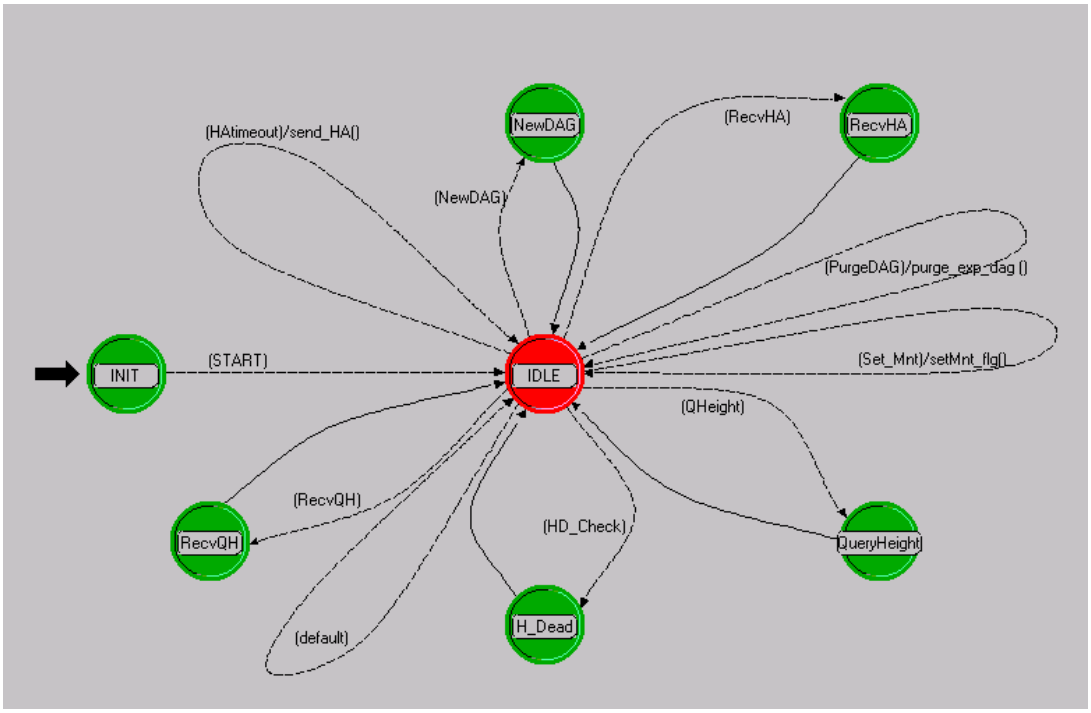


Figure 18:- The DAM Height Protocol

Figure 18: Shows the DAM Height Protocol as implemented in OPNET.

These process models have been integrated within a full protocol stack as shown in the architecture diagram in figure 16. The remaining layers in the stack within a node are the standard implementations provided by OPNET and conform to all the relevant RFCs excluding the MAC and MPLS forwarding mechanisms; which are custom models. The custom process models (including “DAM” and “DAM Height” models) interface with the standard OSPF model and standard IP stack that includes IPv4 and ARP models. The router node model that has been built in the simulator has a total of eight interfaces. This number of interfaces is sufficient to model the networks used in the experiments conducted. Functionality of the simulator was tested piecewise during implementation by tracing packets through example networks and printing out data structures in the OPNET ODB debugger to confirm operation conformance to the specification described earlier.

4.4 Summary

This chapter has given the details of the DAM protocol and its sub protocol, the DAM Height protocol. The level of detail given in this chapter along with the description in Chapter 3 is sufficient for a working implementation of the protocol. This has been shown in the methodology used to produce the simulator within OPNET. The main objective of the DAM protocol is the calculation and establishment of the MP2P LSP this is carried out in the initialisation phase of the protocol. Once a MP2P LSP has been established it is necessary to maintain it, the protocols maintenance phase ensures this. One of the responsibilities of the DAM protocol is to coordinate its interactions with its sub protocol, invoking the DAM Height protocol for the initialisation and maintenance of a MP2P LSPs corresponding DAG. A working simulator has been produced and is presented here in the context of the complete protocol stack shown in the architecture diagram (figure 16). This is the basis for nodes used within the subsequent OPNET simulator performance evaluation.

Chapter 5

5 DAM Extensions

This chapter presents extensions to DAM that change the behaviour of the protocol. The adaptations that have been designed and implemented make changes to the DAM main protocol. In the first instance a different strategy for execution of localised healing has been formulated so as to better assess the value of the standard implementation of DAM discussed in chapter 4. This scheme does away with the use of the sub protocol "DAM Height protocol". As a consequence the formation and maintenance of trees is accommodated within a single protocol called DAM with Partial Pre-Planning. Section 5.1 describes the principles that underpin this protocol; it also provides a specification, and implementation details. DAM with Load Balancing is presented in Section 5.2. This version of DAM broadens the specification of the main protocol given in Section 4.1. The DAM Height protocol specification is not altered; its existence alongside DAM with Load Balancing is central to the novel approach in providing load balancing between multiple overlaid trees.

5.1 DAM with Partial Pre-Planning

Two adapted versions of the DAM protocol specified in Chapter 4 are DAM with Partial Pre-Planning (DAM-PP) and DAM with Load Balancing (DAM-LB). In this section DAM-PP is considered. DAM-LB is considered in Section 5.2.

DAM-PP has been designed and implemented to perform localised healing based on shortest path first calculations; this has been done so that a comparison could be made between this and the recovery mechanism of the standard implementation of DAM that uses a DAG as part of its method for providing resilience.

The standard version of DAM comprises the main protocol and the sub protocol. The sub protocol or "DAM Height" protocol is used to establish and support a DAG that is used by DAM to provide localised healing in reaction to link failure. DAM-PP adopts an alternative recovery mechanism based on shortest path first calculations. In doing so, the DAM Height protocol is no longer needed to act alongside the main protocol. The description given of the

main protocol in Section 4.1 applies to DAM-PP with the omission of any reference to the sub protocol, as it is not used. In DAM-PP, localised healing is administered through the use of pre-planned alternative neighbours. Each node participating in a tree calculates an alternative neighbour to be used should the current downstream neighbour become unusable. Since a complete alternative backup branch on to the tree is not calculated. This strategy is regarded as partial pre-planning. When a node receives the source-routed list in the initial *Heartbeat* message it identifies its downstream neighbour for this particular tree and stores the address of this node in the Tree Database entry for this tree. The link cost to this downstream neighbour is artificially inflated and the reverse Dijkstra calculation is carried out locally in order to identify an alternative downstream neighbour (if it exists) for this particular tree. In the event of a failure, a *QueryPropagate* message is sent to the pre-calculated alternative downstream neighbour. This directed request for a label works in the same way as it does in the standard version of DAM described in Section 4.1 in that a number of *QueryPropagate* messages may pass between a number of nodes before a label is returned to the node making the original request. The following is an operational specification of DAM-PP in terms of message exchanges and is based on the specification given in Section 4.1.

5.1.1 Interaction using the Heartbeat Message

If the node is an egress node, it calculates the initial MP2P tree using a reverse Dijkstra calculation and then does the following:

- It sends an initial *Heartbeat* message to its one-hop neighbours. This initial *Heartbeat* message has the newly calculated tree encoded in it along with a Tree ID and label binding.

- Every 60 seconds the egress node sends a heartbeat message to all its upstream neighbours in the MP2P LSP. Only the initial *Heartbeat* carries the source-routed list.

The egress maintains a tree in this manner for a period of 30 minutes after which a replacement tree is calculated. The egress sends *Heartbeat* messages that relate to this replacement tree. This includes the initial messages carrying the newly calculated source routed list. Once a replacement tree has been calculated the egress node continues to send *Heartbeat* messages for the old tree for a period known as the *Tree Purge Time*, after which it ceases to refresh the old

tree. During this period both the new and old tree are supported by the egress in order to ensure connectivity is maintained whilst a tree is being replaced.

If an intermediate or ingress node has received an initial *Heartbeat* message from one of its downstream neighbours the following steps take place:

- If it is an intermediate node, the node examines the received *Heartbeat* message and determines from the source-routed list encoded into this initial message which of its neighbours form part of the tree. It then encodes its own label into this initial *Heartbeat* message before sending it on to its neighbours that are upstream with respect to this MP2P LSP.

- Upon receiving an initial *Heartbeat* message, a node records its downstream neighbour for this particular tree. It then artificially inflates[1] the cost to this downstream neighbour before performing the reverse Dijkstra calculation to identify an alternative downstream neighbour should one exist. If an alternative downstream neighbour exists then this is recorded in the tree database entry for this tree.

- If the node is an intermediate node it then adds the label in the received *Heartbeat* to its Next Hop Label Forwarding Entries (NHLFE) and its own label transmitted in the revised initial heartbeat to its Incoming Label Map (ILM).

- If the node is an ingress node it then adds the label in the received *Heartbeat* to its NHLFE and makes any necessary adjustments to its FEC-to-NHLFE map (FTN).

[1] – The cost to the current downstream neighbour is artificially inflated by adding an offset to its actual cost. This offset is dependant upon the specific tree for which the calculation is being made; all links and their respective costs in the tree are examined and the offset is set to the highest link cost in this set.

Subsequent *Heartbeats* are revised by intermediate nodes to carry local label bindings and are then forwarded. Moreover, each node that receives a *Heartbeat* message then sends on its revised version of this message to its upstream neighbours until this message has propagated to all the ingress nodes in the MP2P LSP. The revised *Heartbeat* messages have the local label bindings encoded within them acting as a refresh mechanism for the MP2P LSP.

5.1.2 Interaction using the QueryPropagate Message

The *QueryPropagate* Message is used by the DAM-PP protocol as a directed request for a label binding either as part of the localised healing process after a failure or when a new node wishes to join an existing tree. A node receiving a *QueryPropagate* message acts in the following way:

- The node receiving the *QueryPropagate* message is part of the MP2P LSP with the corresponding tree ID in this message, in which case it adds the node that sent this message to its list of upstream nodes in this MP2P LSP. As a consequence, the node sends a *Heartbeat* message with label binding to its neighbour that sent the *QueryPropagate* message.

- The receiving node is not part of the MP2P LSP with the corresponding tree ID in this message, in which case the node performs a reverse Dijkstra calculation in order to identify its downstream neighbour. The node creates a tree database entry for this tree and calculates the alternative downstream neighbour by performing the reverse Dijkstra calculation using an artificially inflated cost as described previously. It then periodically sends a *QueryPropagate* message to its downstream neighbour until it receives a *Heartbeat* message. Upon receiving this *Heartbeat* message, the node appends and sends the message to its upstream neighbour from which it received the *QueryPropagate* message. This process continues until the node that sent the initial *QueryPropagate* message has received a *Heartbeat* message with label binding resulting in either a successful join on to an existing tree or a successful localised healing.

This specification was decomposed into a state transition table so that the simulator of DAM-PP could be implemented within OPNET. Since this state transition table is based on the one for DAM in Section 4.3 only part of this table is shown here, the entire table can be seen in Appendix 1.

This table shows the sections of the state transition table that are specific to DAM-PP.

State	Event	Condition	Action	Final State
RecvHBeat	Initial Heartbeat message	Always	Replace Label, Send on revised HB to upstream neighbours, Schedule interrupt for PrPlan state to select alternative downstream neighbour	RecvHBeat
DamJoin	MNT / QP process	Tree status is MNT	Get pre planned alternative downstream neighbour, Send QueryProp message and set QP interrupt for some future time	IDLE
DamJoin	MNT / QP process	Tree entry does not exist	Set QP interrupt for some future time – 10s	IDLE
PrPlan	Calculate alternative downstream neighbour	Always	Calculate cost offset for tree, perform reverse Dijkstra using inflated cost	IDLE

The DAM-PP simulator is based on the DAM main protocol simulator described in Section 4.3. The data structures used in implementing DAM-PP are the same with the exception of the tree entry structure, which is used to form the elements of the Tree Database linked list. The tree entry structure has been adapted to meet the needs of the DAM-PP specification and is as follows:

```
/* used for keeping track of trees */

typedef struct
      {
      IpT_Address       egress_addr;
      unsigned int treeNo :32;
      unsigned int PtreeNo :32;  /*tree no of previous tree i.e the
                                   one being replaced*/
      double crtime;             //creation time
      int T_D_status;            /*set to 1 for MntTimeout and set to
                                 2 if tree is considered dead*/
      int Purge_T_status;        /*set to 1 if tree has been replaced
                                 and will be supported for the purge
                                 time*/
      double Last_HB_Time;       /*time last HB was received if
                                 ingress or intermediate node / set to
                                 zero if this is the egrees and not
                                 used */
      unsigned int label : 20;   // label used for this tree
      IpT_Address upstrm_nbour[8]; /*upstream neighbours in this
                                     tree-[8]for eight intf */
      IpT_Address dn_nbour;      //downstream neighbour
      IpT_Address qp_nbour;      /*router ID of preplanned
                                 neighbour in the event of failure*/
      int h_rcost;               /*highest rootcost in this tree used
                                 as cost offset in preplanned
                                 calculation */
      } tree_entry;
```

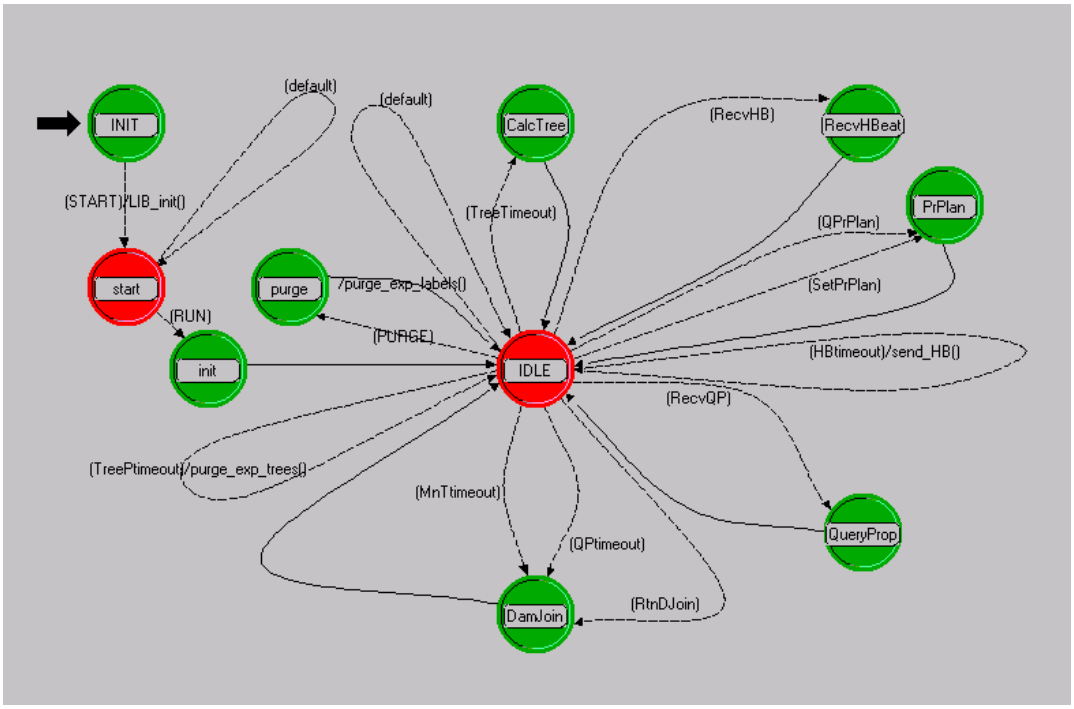


Figure 19:- The DAM-PP Protocol

Figure 19 shows the DAM-PP Protocol as implemented in OPNET.

5.2 DAM with Load Balancing

DAM with Load Balancing (DAM-LB) broadens the specification given in Chapter 4 to include a novel load balancing mechanism. DAM-LB extends the standard version of the DAM main protocol described in Section 4.1. The sub protocol "DAM Height protocol" described in Section 4.2 remains unchanged and works alongside DAM-LB in the same manner as it does in the edge orientated implementation of DAM in Chapter 4. The load balancing mechanism in DAM-LB takes advantage of the acyclic nature of the DAG being maintained by the DAM Height protocol. This ensures that DAM is able to direct label requests in a structured manner for any given root or egress node. Moreover, the view of a given network topology represented as a DAG allows nodes other than the root node to have one or more outgoing links. Nodes in a DAG that have more than one outgoing link for a particular egress node present an opportunity for the better balancing of multiple overlaid trees and loads across a network. In Section 2.6, a DAG is described to be effectively a graph of multi-paths to the destination node. This property is central to this load balancing extension to DAM. The load balancing extension within DAM-LB is a distributed mechanism that acts locally through the DAG. Each node is able to deduce from its tree database the number of MP2P LSPs associated with a particular downstream neighbour. In doing so, a decision can be made locally as to whether or not a label request should be made to an alternative downstream neighbour using the *QueryPropagate* message. Each label request made to an alternative downstream neighbour relates to a particular MP2P LSP. A node can therefore choose explicitly which MP2P LSPs can be re-routed via an alternative downstream neighbour, such decisions can be made on the basis of a multi-service model whereby some traffic flows are given priority over others. The receipt of a label from an alternative downstream neighbour in response to a specific label request is considered to be a positive acknowledgement and as such the alternative downstream neighbour becomes the active neighbour for that particular MP2P LSP. The node receiving this request can choose to respond to this request or indeed ignore this request if it has insufficient resources to accommodate additional transit traffic. DAM-LB provides the framework for load balancing to take place using the DAG to re-route sections of a tree. In the implementation described here for simplicity the decision to reroute is not based on link capacity or the size of the flows being rerouted, instead load balancing is achieved indirectly by balancing the number LSPs flowing in one direction on a link using label balancing to achieve load balancing. Further development of the framework provided by DAM-LB would include actual link capacity along with current/future resource demands and CoS considerations as part of the localised decision to

reroute tree branches. Figure 20 shows an example operation of DAM-LB showing a section of a tree being rerouted using alternative downstream neighbour and path in the DAG.

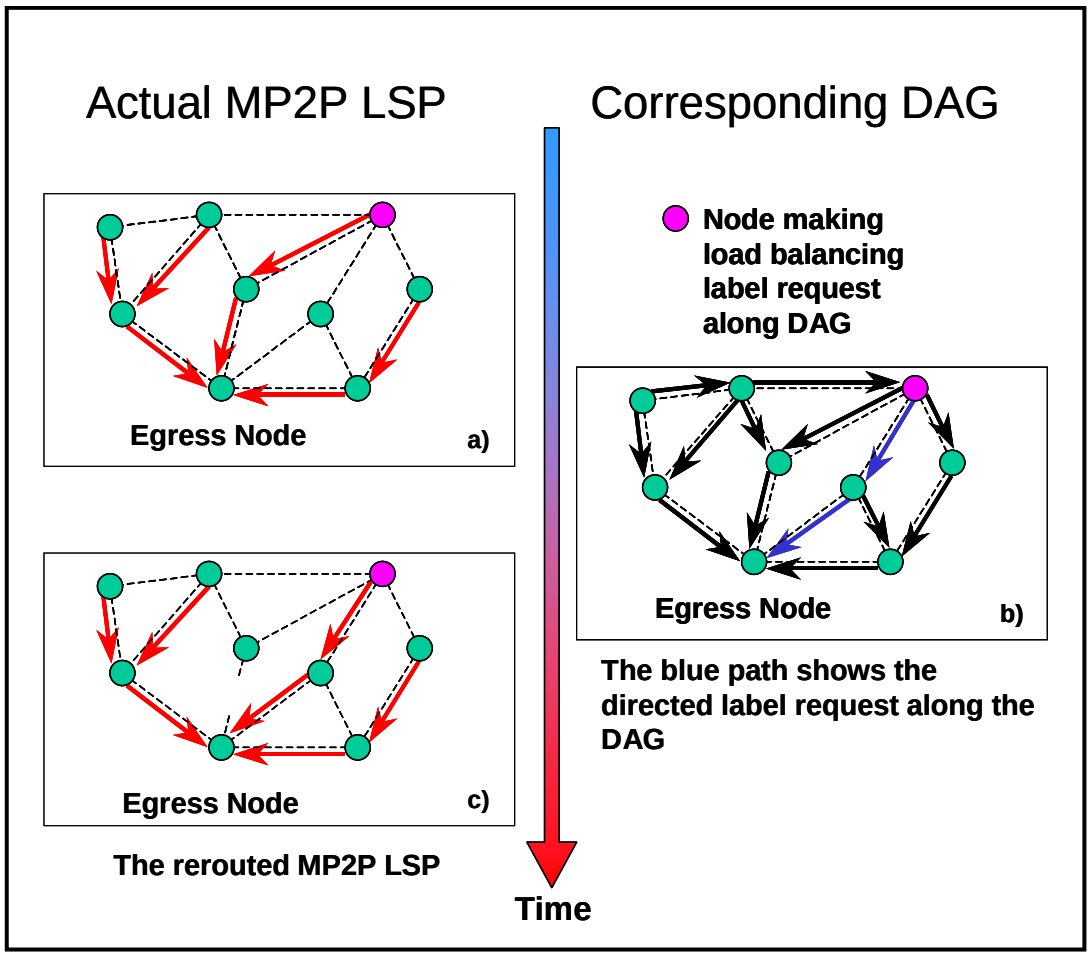


Figure 20:- Example operation of DAM-LB showing a section of a tree being rerouted along an alternative path in the DAG

To support this arrangement the tree entry structure that makes up the elements within the Tree Database has been modified. In DAM-LB, the tree entry structure is made to include the address of the downstream neighbour. This is the downstream neighbour identified within the source-routed list during the initial set up phase of the MP2P LSP. In addition, the address of the alternative downstream neighbour is determined and recorded using the DAG. To determine which downstream neighbour is currently being used for a particular tree a binary flag has been included to determine the "active neighbour" of the two stored in the database. The final addition to this structure is an index called the downstream utilisation index. This is a number that relates to the current active downstream neighbour and describes that neighbours ability to

accommodate additional traffic on a particular tree. This number is transmitted within the *Heartbeat* message; in this implementation of DAM-LB it describes the number of trees sharing the same outgoing link/interface at that node. This index could be made to reflect other factors that affect a node's ability to carry extra traffic such as available capacity and the number of hops it is from the egress. In the current implementation of DAM-LB, the downstream utilisation index reflects the number of MP2P LSPs using the same downstream link at a particular node. It is calculated and updated by a node examining its tree database to determine how many MP2P LSPs are using a particular downstream neighbour to reach the destination node. The calculated downstream utilisation index for a particular link can be used locally and is also communicated upstream to aid in determining which trees can have branches rerouted.

Extending the specification described here can provide support for multiple service classes. In order to implement multiple trees to a particular egress to satisfy different service levels tree number space can be partitioned to reflect grades of service, in other words the tree numbers that make up the unique tree ID (egress address and tree No.) are divided and assigned between the different service levels that are supported. It is also possible to support different service levels within the same MP2P tree by using the CoS field in the MPLS header. The CoS field is a three bit field transmitted along with the label in the header and is used to reflect the CoS type.

The modified tree entry structure is as follows:

```
typedef struct
      {
      IpT_Address       egress_addr;
      unsigned int treeNo :32;
      unsigned int PtreeNo :32;  /*tree no of previous tree i.e the
                                   one being replaced*/
      double crtime;             //creation time
      int T_D_status;            /*set to 1 for MntTimeout and set to
                                 2 if tree is considered dead*/
      int Purge_T_status;        /*set to 1 if tree has been replaced
                                 and will be supported for the purge
                                 time*/
      double Last_HB_Time;       /*time last HB was received if
                                 ingress or intermediate node / set to
                                 zero if this is the egrees and not
                                 used */
      unsigned int label : 20;   // label used for this tree
      IpT_Address upstrm_nbour[8]; /*upstream neighbours in this
                                     tree-[8]for eight intf */

      IpT_Address dn_nbour;      //downstream neighbour
      IpT_Address qp_nbour;      /*router ID of neighbour used for
                                 load balancing*/
```

```
int dn_qp_flag;             /*set to zero if dn_nbour is being
                            used and 1 if qp_nbour is active*/
int dn_utilization_index;  /*downstream utilization index of
      active downstream neighbour (either dn_nbour or qp_nbour)
      zero if egress*/
} tree_entry;
```

The following describes the message interaction of DAM-LB in terms of the *Heartbeat* message and the *QueryPropagate* message. The specification given in Section 4.1 is valid for DAM-LB this section therefore only includes the extensions to the standard specification.

5.2.1 Interaction using the QueryPropagate Message

The *QueryPropagate* message is used by the DAM protocol as a directed request for a label binding either as part of the localised healing process after a failure or when a new node wishes to join an existing tree. Within DAM-LB it is also used to implement load balancing along the DAG where this is possible. To facilitate this, the reserved field in the *QueryPropagate* packet header specification given in Chapter 4 is now used to carry a code relating to the message type. This code means that when the type is set to zero, the *QueryPropagate* message relates to a localised healing or joining of a tree, message interaction for this type of message is as specified in Section 4.1. When the type is set to 1 in the message header this specifies that the label request made by this *QueryPropagate* relates to load balancing. A node receiving a *QueryPropagate* message of type 1 acts in the following way:

- The node receiving the *QueryPropagate* message identifies the tree database entry relating to this message. The node then identifies which is its current active downstream neighbour. The node examines all other entries in its Tree Database and calculates the number of trees using this same downstream neighbour; this number is used to decide whether it will respond to this label request. If the node decides to respond it sends a *Heartbeat* (type 3) message with label binding to its neighbour that sent the *QueryPropagate* message.

In the current implementation a threshold is calculated for the number of trees using the same downstream neighbour. A node that has more trees than this threshold using the same downstream neighbour does not respond to this type of label request. This threshold is

calculated by :- Z / (degree of the network – 1) where Z is the number of MP2P LSPs being supported and Z is not less than the number of nodes in the network.

5.2.2 Interaction using the Heartbeat Message

In order to facilitate the load balancing extensions to DAM, the *Heartbeat* messages in response to load balancing label requests are of message type 3. This means there are four *Heartbeat* message types also including “type 1” for an initial *Heartbeat*, “type 2” for a *Heartbeat* used to perform localised healing or joining and “type 0” for subsequent *Heartbeat*s. The message interaction described in Section 4.1 is valid; however the following rules apply:

- On receiving an initial *Heartbeat* “type 1” the node calculates its downstream utilisation index and if necessary makes a label request to an alternative downstream neighbour.

- If a node receives a “type 2” *Heartbeat* message it must reset its active downstream neighbour flag to zero to reflect that its dn_nbour in the tree database is its active downstream neighbour as a result of the recovery process.

- Receiving a “type 3” *Heartbeat* requires a node to set its active downstream neighbour flag to one indicating that its qp_nbour (node being used to perform load balancing) in the tree database is now its active downstream neighbour.

- Subsequent “type 0” *Heartbeat* messages are not used to update the Label Information Base or tree database unless they originate from the active downstream neighbour.

As before, the protocol specification has been decomposed into a state transition table so that the protocol can be implemented within OPNET. The full table is shown in Appendix 2, the following are the sections of the table that are specific to DAM-LB.

RecvHBeat	Initial Heartbeat message (HB type 1)	Always	Replace Label, Send on revised HB to upstream neighbours, Set "Send_LBQP" interrupt if load balancing is required	IDLE
RecvHBeat	Normal Heartbeat (HB type 0)	Always	Only respond to HB from active downstream neighbour - Reset timers and send on revised heartbeat	IDLE
RecvHBeat	Heartbeat in reply to a QProp message for healing or joining a tree (HB type 2)	Always	Update tree DB (reset downstream neighbours and active neighbour flag) and remove source of this HB from upstream neighbour list if necessary	IDLE
RecvHBeat	Heartbeat in reply to a QProp message for load balancing request (HB type 3)	Always	Update tree DB (set alternative downstream neighbour and active neighbour flag = 0)	IDLE
QueryProp	QP message type 1 for Load balancing, Node can accept additional traffic	Always	Send label binding (HB type 3) to source of QP and add this node to list of upstream neighbours	IDLE
LB_QP	More than one downstream neighbour exists in the DAG	Always	Send QP type 1 to alternative downstream neighbour	IDLE

The DAM-LB simulator has been implemented as a process model within OPNET.

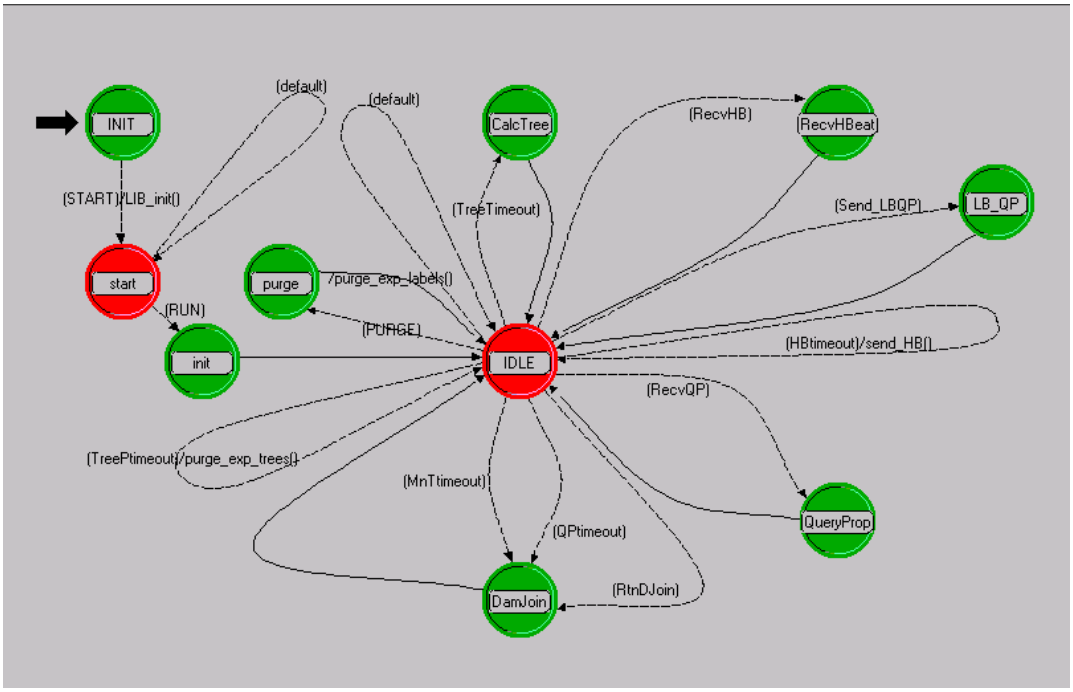


Figure 21:- The DAM-LB Protocol

Figure 21 Shows the DAM-LB Protocol as implemented in OPNET.

5.3 Summary

This chapter has presented two separate extensions to the DAM specification given in Chapter 4. In Section 5.1 DAM with Partial Pre-Planning has been presented, the motivation for the design and specification of this adapted protocol was to examine an alternative way of implementing localised healing and in doing so, to better assess the value of the strategy adopted by the original specification of DAM. DAM-PP adopts an alternative recovery mechanism; localised healing is administered through the use of pre-planned alternative neighbours. Each node participating in a tree calculates an alternative neighbour to be used should the current downstream neighbour become unusable. This strategy is considered to be partial pre-planning since a complete alternative backup branch on to the tree is not calculated. This section also presents details regarding the implementation. The protocol specification and state transition table show clearly how the changes from the original specification have translated into a working simulator of DAM-PP.

Section 5.2 presents a novel method for performing load balancing for multiple overlaid trees called DAM with Load Balancing “DAM-LB”. The load balancing mechanism in DAM-LB takes advantage of the acyclic nature of the DAG being maintained by the DAM Height protocol. This DAG is effectively a graph of multi-paths to a particular destination, DAM-LB uses signalling to reroute sections of a tree along these multi-paths. This signalling involves making directed label requests along the DAG on the basis of decisions made locally with respect to specific trees. Nodes receiving such label requests are at liberty to respond if they have resources to accommodate additional traffic, otherwise they may simply ignore such requests on a case-by-case basis. This section describes all the necessary details required to implement such load balancing. Moreover, the protocol extensions to the DAM main protocol are specified along with the state transition table and data structure changes needed to create the DAM-LB protocol within OPNET.

Chapter 6

6 Simulation and Analysis

This chapter presents an analysis of the message exchange used by the standard implementation of DAM. Following this analysis, various experiments are conducted using OPNET with the DAM protocol simulator. The aim of these experiments is to evaluate the effectiveness of the DAM approach as well as obtaining a greater understanding of the protocols performance in realistic network scenarios. This chapter then presents studies using the DAM extensions described in chapter 5. Recovery times are examined for DAM-PP so that a comparison can be made with the standard implementation. Link utilisation is then examined using the DAM-LB simulator.

6.1 Analysis

Before any simulation was conducted an analysis of the number of messages exchanged when performing localised healing in an example network was carried out.

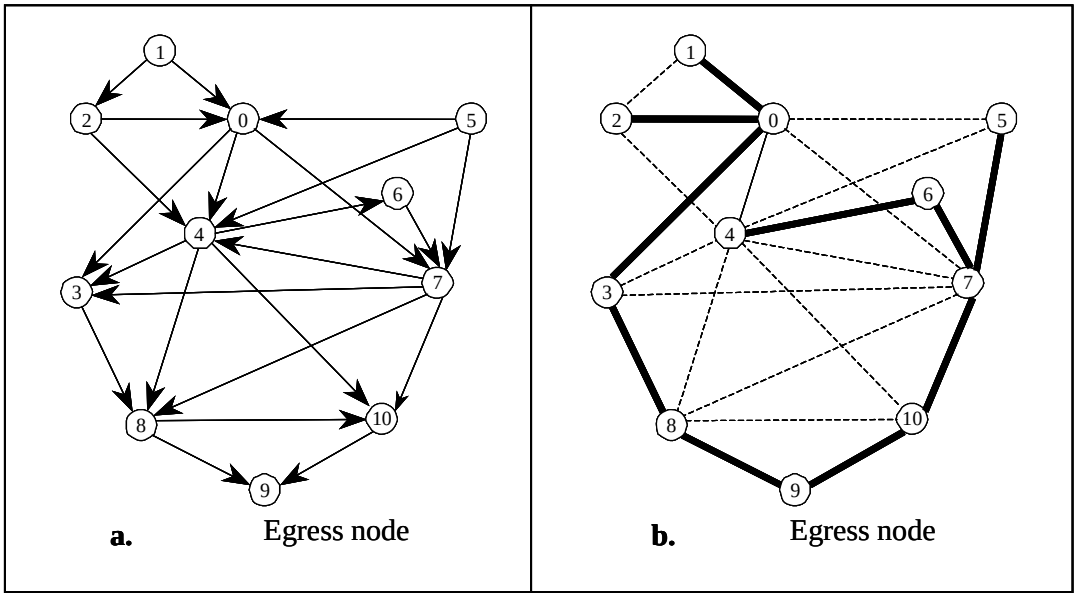


Figure 22:- Example network fig. 22a shows a DAG of the network and fig. 22b shows the corresponding MP2P tree

The network shown in figure 22 is based on the New Jersey LATA Network [MUR95][XIO99]. Figure 22a shows a Directed Acyclic Graph used by DAM to perform fast localised healing. The DAG in figure 22a was formed using the principles described in Section 3.2. Figure 22b shows a corresponding MP2P tree for the same network. Analysis was conducted to determine the number of messages required by DAM to reform the MP2P LSP. All possible single link failures in the MP2P tree were evaluated and the number of messages required to reform the tree was calculated in each case. The number of messages required consists of the number of messages required to reform the DAG (which in many cases is zero) plus the number of messages in the directed request using the *QueryPropagate* message in order to obtain a new label binding to reconnect to the tree. This was contrasted with sending notification of the failure back to the egress and then signalling a new branch in order to re-establish connectivity in effected areas of the tree. Using this method the egress reactively generates a new branch to replace the affected branch. In addition to this type of restoration signalling for pre-calculated restoration was examined. The number of messages used to perform 1:1 end-to-end protection switching was also calculated; for this calculation all nodes in the network were considered to be ingress points. The number of messages required to signal the failure to all affected ingress nodes was noted.

Analysis yields the following results:

Link Failed	No. Messages to signal failure to egress	No. Messages to signal backup branch	Total No. of messages using backup branches	No of effected Upstream nodes	Total No. of messages for Protection Switching
1 – 0	3	5	8	1	0
2 – 0	3	5	8	1	0
3 – 0	2	6	8	2	2
8 – 3	1	6	7	4	5
9 – 8	0	6	6	5	9
7 –5	2	3	5	1	0
6 – 4	3	3	6	1	0
7 – 6	2	3	5	2	1
10 – 7	1	5	6	4	4
10 – 9	0	6	6	5	8
			Average no: **6.5** Standard deviation: **1.18**		Average no: **2.9** Standard deviation: **3.27**

Table 1:- Shows message exchange analysis for recovery using selective backup branches and for end-to-end protection switching

Link Failed	Node degree upstream of failure	No. of messages to reform DAG	No. of messages to get label binding	Total No. of messages using DAM
1 – 0	2	0	2	2
2 – 0	3	0	2	2
3 – 0	6	0	2	2
8 – 3	4	1	2	3
9 – 8	5	0	2	2
7 –5	3	0	2	2
6 – 4	8	0	2	2
7 – 6	2	1	4	5
10 – 7	7	0	2	2
10 – 9	4	1	2	3
		Average no: **0.3** Standard deviation: **0.46**	Average no: **2.2** Standard deviation: **0.6**	Average no: **2.5** Standard deviation: **0.92**

Table 2:- Shows message exchange for recovery using DAM

The average number of messages required to re–establish connectivity using DAM is 2.5 for this particular example. Conversely the average number of messages required for signalling replacement backup branches reactively is 6.5, demonstrating the advantage of localised restoration. The average number of messages required for failure notification for end-to-end protection switching is 2.9. It is important to note that even though the analysis shown here is for this particular example the number of messages required by DAM to complete a localised restoration is uniform in most cases as more often than not no signalling is required to reconstitute the DAG. Signalling is only required to reform the DAG when the node upstream of the failure loses its last outgoing link. Only the egress node is allowed to have only incoming links. In both other methods examined in this analysis, the number of messages required for recovery is related to network size, in that signalling on average must travel further to reach ingress or egress nodes as the network grows.

The number of messages required to reform the MP2P LSP using DAM was noted for the following example using the Nation Science Foundation (NSF) network [AMY01].

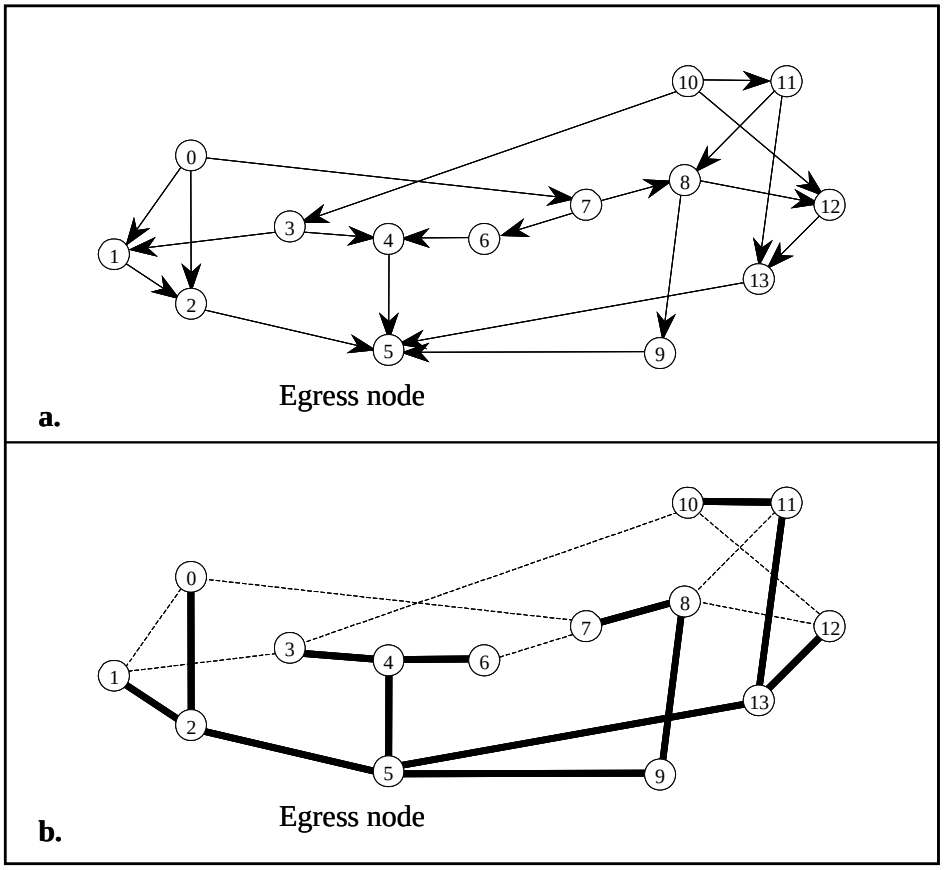


Figure 23:- Nation Science Foundation (NSF) network [AMY01] fig. 23a show the DAG for node 5 & fig. 23b shows the MP2P LPS with node 5 as the egress node

Link Failed	Node degree upstream of failure	No. of messages to reform DAG	No. of messages to get label binding	Total No. of messages using DAM
1 –2	3	1	2	3
0 –2	3	0	2	2
2 – 5	3	3	4	7
3 – 4	3	0	2	2
4 – 6	2	1	2	3
4 – 5	3	2	2	4
7 – 8	3	0	2	2
8 –9	4	0	2	2
5 –9	2	1	2	3
		Average no: **0.89** Standard deviation: **0.99**	Average no: **2.2** Standard deviation: **0.63**	Average no: **3.1** Standard deviation: **1.52**

Table 3:- Showing message counting for example shown in fig 23.

The number of messages required to reform a MP2P LSP using DAM is dependant on the local topology near to the point of failure. The healing process is comprised of two sets of signalling, these are the messages that are required to reconstitute a DAG and the messages required to obtain a label binding in order to reform the MP2P LSPs. Signalling is only necessary for reconstituting the DAG if a node loses its last outgoing link. All nodes in the DAG must have at least one outgoing link (only the egress node is allowed only incoming links) thus the degree of the node upstream of the failure impacts on the probability of the failed link being the last outgoing link. The greater the node degree the less likely it is that a failed link is the last outgoing link. Conversely the smaller the node degree the more likely it is that a failed link is the last outgoing link. The signalling associated with obtaining a label binding and as a result, a path back onto the MP2P tree, is determined by local topology.

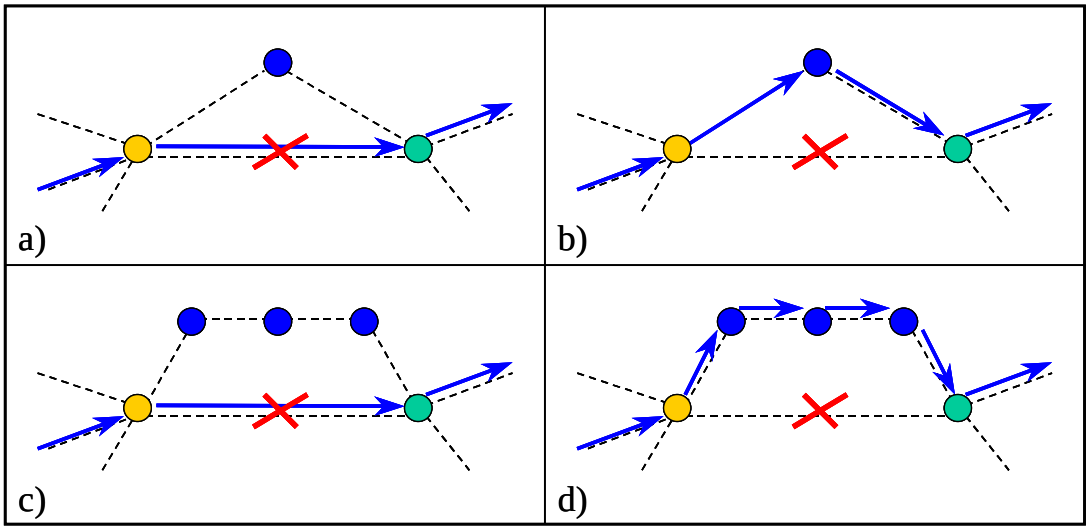


Figure 24:- Local topology near the point of failure; figs 24a & b show an example where the rerouted section of the MP2P LSP traverses one intermediate node; figs 24c & d show an example where the rerouted section of the MP2P LSP traverses three intermediate nodes

Figure 24a shows a link failure and the local topology near to this failure, the blue arrows show the path taken by the MP2P LSP. In figure 24b the MP2P LSP has been rerouted locally around the failure via the blue node. The number of messages required by DAM to complete the rerouting shown in figure 24b is two messages per hop back on to the tree. In figure 24b the number of messages used to reroute the MP2P LSP could be only two if the blue node is part of the MP2P tree. In figure 24b if the blue node is not part of the MP2P LSP then the number of messages required to perform rerouting is four since the directed label request/reply must propagate a distance of two hops to reach the green node. Figures 24c and 24d show the rerouting of the MP2P LSP around the failure. The local topology in this example has three blue

nodes along the rerouted section of the MP2P LSP. The total number of messages required to reroute the LSP in figure 24d could be as many as eight messages since the directed label request/reply must propagate four hops in the case when none of the blue nodes are part of the MP2P tree. The total number of messages required to reroute the LSP in figure 24d could be as little as two messages if the blue node closest to the gold node is part of the MP2P LSP. The number of messages required to obtain a label binding is two for each hop traversed by the directed label request/reply. To reroute via an alternative downstream node on the MP2P LSP, the tree density around the point failure and local topology determine how quickly a label binding can be obtained after a failure has occurred.

6.2 Single Link Failure

In any single link failure, the time taken to recover a MP2P LSP using DAM is dependent upon the complexity of the recovery involved. The complexity of the recovery is dictated by the network topology and the impact this has on the DAG set up by the DAM Height protocol. In the simplest case, the loss of a link does not require the DAG to be reformed. A recovery is carried out simply by a label request using the *QueryPropogate* message to the neighbour with the lowest height. The healing action is quickest for this type of recovery as the number of messages exchanged between nodes is minimal. This comprises only two messages; the directed label request and reply. In contrast to this, a non-trivial recovery requires more signalling and hence takes more time to heal the MP2P LSP. A special case that has been shown in the scenario in the previous section is when all of the affected node's remaining active links are attached to upstream routers that form part of the tree; in such a case one of its upstream routers will be forced to become a downstream in that particular MP2P tree. A more general description of this is as follows:

A node along the tree (not the egress) has one outgoing link, any number of incoming links and zero or more unused links. In the case when there are zero unused links that can be used for recovery in the event of a failure the last outgoing link is lost. When this occurs the node is forced to reverse one of its incoming links. This reversal results in the reversed link becoming the new outgoing link so that the recovery process can be completed.

A scenario has been devised to evaluate this through simulation as shown in figure 25. This also formed the basis for validation of the simulation models.

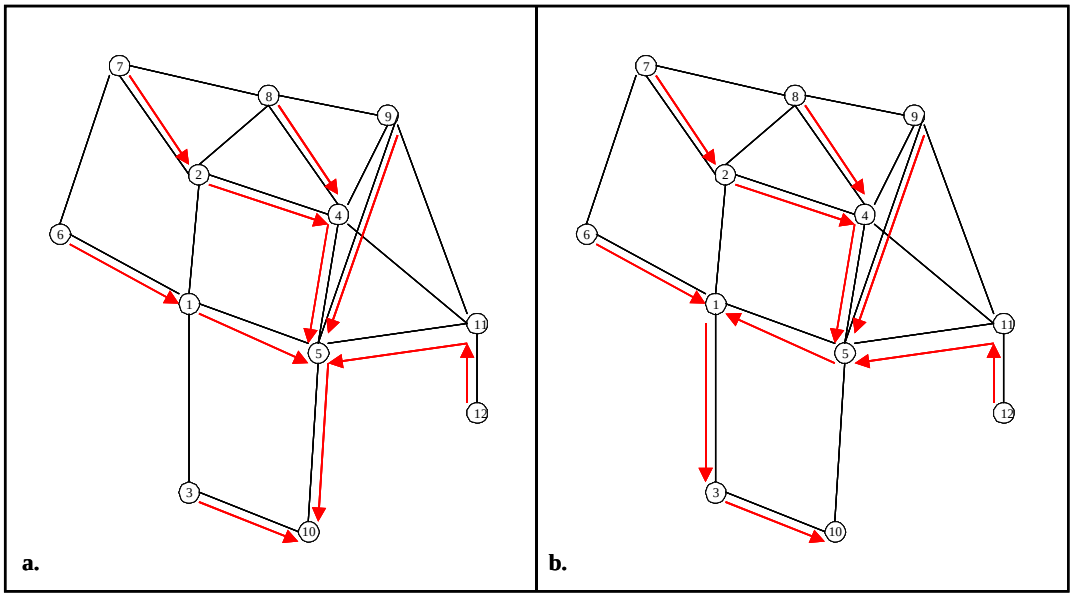


Figure 25:- Network Scenario simulated in OPNET - fig 25 a) shows MP2P LSP before failure, fig 25 b) shows MP2P LSP after restoration

The network topology in figure 25a has been simulated and the link between nodes 5 and 10 has been failed. In this scenario node 10 is the egress in the MP2P LSP; before the failure takes place (figure 25a) all of node 5's neighbours are upstream in this particular tree. In order to heal the MP2P LSP node 1 must become a downstream neighbour to node 5 and also get a label from node 3. As a result node 3 will become node 1's new downstream neighbour. The reformed MP2P LSP takes the form shown in figure 25b once this recovery has taken place.

In order for the recovery to complete, the following message exchange takes place between nodes 5, 1 and 3.

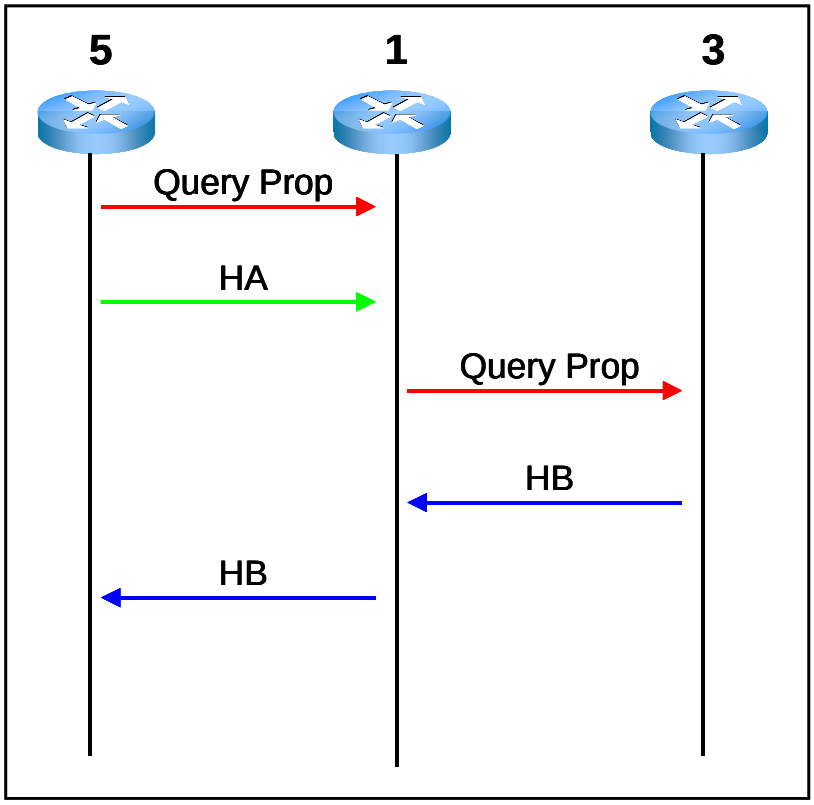


Figure 26:- Message exchange between affected nodes

In the figure 26 ***HA*** is a Height Advertisement sent by the DAM Height protocol in order to reconstitute the DAG (note that this message is not sent if the DAG does not need to be reformed). ***Query Prop*** is the *QueryPropogate* message sent by the DAM protocol. ***HB*** is a *HeartBeat* message sent by the DAM protocol with the new label in order to complete the recovery.

All simulations presented in this thesis were conducted using the same link set up regardless of actual topology. The main factors in deciding upon the chosen simulation set up were the ability to perform timely simulations in OPNET whilst obtaining meaningful results. As a result the following is true for all simulations conducted in this work:

- Propagation delay is set to zero, in other words it has no bearing on any of the results collected.
- point-to-point links between nodes with a data rate of 64000 bits/s were used
- Signalling delays were calculated for the following message types:

Message type	**Message size in bits**	**Signalling delay (ms)**
QueryPropogate	256	**4** ms
HeartBeat	416	**6.5** ms
Height Advertisement	320	**5** ms
QueryHeight	256	**4** ms

Table 4:- Signalling delays for DAM message types using 64000 bits/s links

It should be noted that the figures in table 4 are only true for when recovery is taking place. Variable packet lengths are supported and the size of the *HeartBeat* message is larger when it is an initial *HeartBeat* message carrying the source-routed list.

- IP packet forwarding delay (to lower or higher layers) was calculated to be **0.00667ms** this is a forwarding rate of 150000 packets per second.

From the time that link failure was notified localised healing took **10.5533 ms** to complete. Similar results were observed for other single link failures in this scenario, recoveries taking between **10.5267 ms** for recoveries in which no signalling was required to reverse a link to **10.5533 ms** for the non-trivial recovery described. The time taken to reform a DAG is negligible and in situations where this happens it is often the case that only one iteration of the partial reversal method is required. In terms of signalling cost, reforming the DAG may only involve a single message being sent from the affected node to its immediate neighbours and since the height calculation is only done at the affected node, this has nearly no effect on the recovery time. In the example shown in figure 25, the most time consuming process is the message exchange required to obtain a new label back on to the MP2P LSP. It should be noted that the recovery time of 10.5533 ms seconds recorded for the non-trivial recovery accounts for two time slots of signalling delay; this is because node 5 and node 1 are independently and simultaneously obtaining new labels as a result of the failure. It is also worth noting that link failure in this example affects five upstream branches in this particular MP2P LSP; an equivalent P2P LSP arrangement would require the restoration of at least five LSPs. The following timing diagram shows the delays associated with the directed label request/reply when no signalling is required to reform the DAG. The times shown in figure 27 were observed in OPNET by running and stopping the simulation on an event-by-event basis.

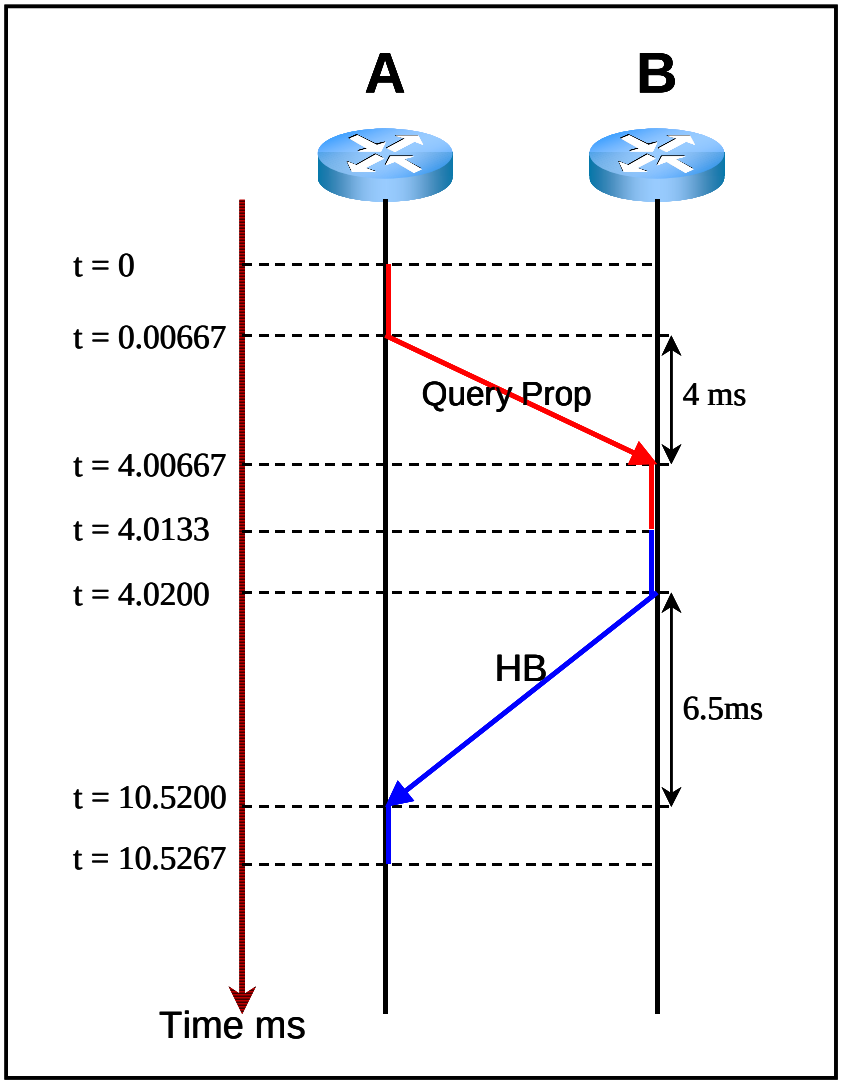


Figure 27:- Timing diagram showing directed label request/reply when no signalling is required to reform the DAG

Figure 27 shows in detail the time components making up the resultant 10.5267ms that was recorded when no signalling was required to reform the DAG in order to perform localised healing. Signalling delays account for 10.5ms of the recovery time, 4ms for the *QueryPropogate* message and 6.5ms for the *HeartBeat* message. The remaining time (**0.0267ms**) is made up of four periods of IP packet forwarding delay, each delay accounting for **0.00667ms.** Figure 27 shows that each message requires two periods of IP packet forwarding delay, one when it is being sent and one when it is being received.

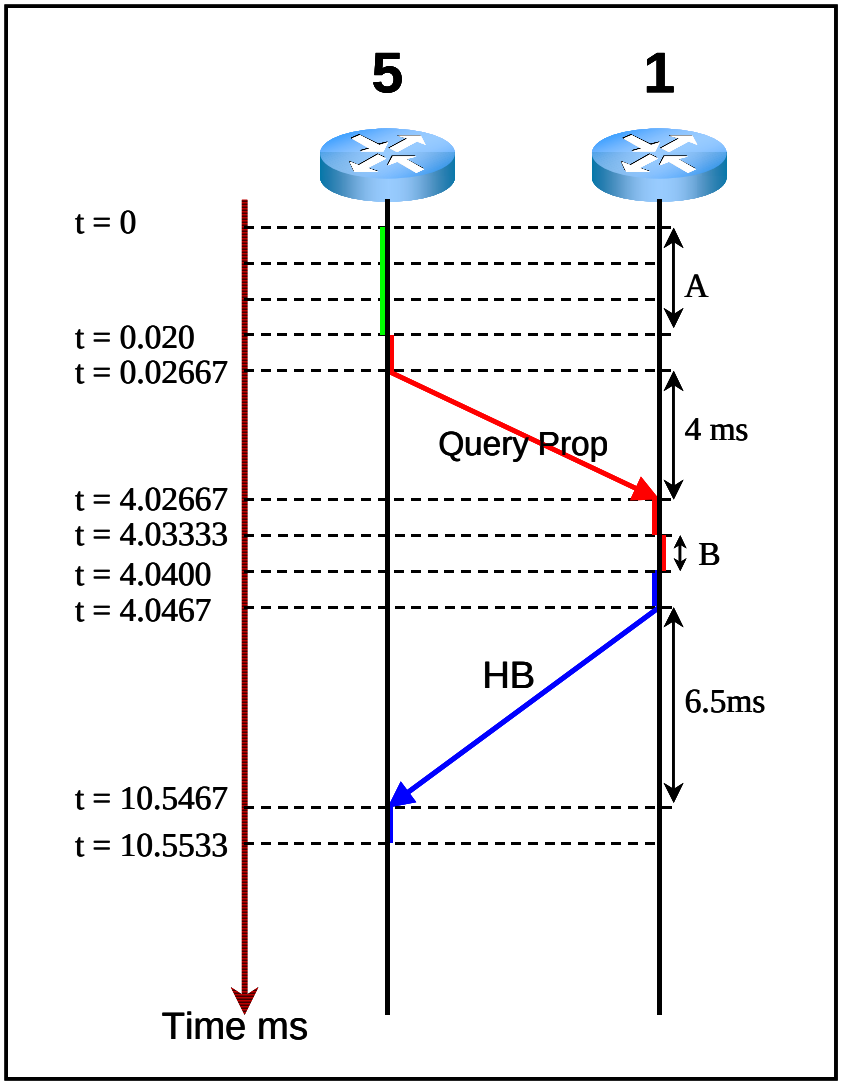


Figure 28:- Timing diagram showing directed label request/reply for non-trivial recovery shown in figures 25 & 26

Figure 28 shows the time constituents that make up the non-trivial recovery shown in figures 25 and 26. The time recorded for the recovery shown in figure 25 is **10.5533ms**. Signalling delays account for **10.5ms** and **0.0267ms** is the IP packet forwarding delay associated with the *QueryPropogate* message and *HeartBeat* message sent between node 5 and node 1. Together these times make up **10.5267ms** of the recorded recovery time. The remaining recovery time (**0.0267ms**) is comprised of further four units of IP packet forwarding delay, these are shown in figure 28 and are labelled **A** and **B**. Three units of IP packet forwarding delay are marked in green and labelled **A** in figure 28, these three delays at node 5 are the result of three *Height Advertisement* messages sent by node 5 to neighbouring nodes 4, 9 and 11. A single unit of IP

packet forwarding delay is marked in red and labelled **B** in figure 28, this delay takes place at node 1 and is the result of node 1 sending a *QueryPropogate* message to node 3. Node 1 sends node 3 the *QueryPropogate* message since at this point it has not received the label binding from node 3. The *HeartBeat* message from node 1 to node 5 is sent before node 1 receives the *HeartBeat* message from node 3 this is because the link between node 1 and 3 is active in the DAG and node 1 anticipates a reply.

This experiment shows the scale of recovery signalling to be similar to that for single link failures as the reaction to failure is localised around the failure point. Given the fact that the reaction is localised, the time taken for recovery is independent of network size but dependant on the topology in the vicinity of the failure. As has been discussed earlier in this thesis (Section 2.3) this is not usually the case in many P2P arrangements, as recovery requires signalling back to the ingress or egress to switch to an alternative path. This notion is re-enforced by further simulations described in Section 6.3. It is also the case that one failure may require multiple LSPs to be repaired.

Other researchers have shown that the number of re-established LSPs has an approximately linear relationship with recovery time [MAR05]. In the study by Margaria *et al* [MAR05] the working and backup paths are identical for all 14 LSPs being recovered. The linearity displayed in the study by Margaria *et al* [MAR05] is probably attributed to the uniform amount of processing and signalling required in recovering each individual LSP. The impact of recovery times for multiple LSPs is of particular interest as in general this has been overlooked as a major benefit of MP2P LSPs. It has in fact been overshadowed by label space reduction that is the focus of other studies [SAI00] [BHA02] [BHA03] & [APP03] into MP2P LSPs.

6.3 Re - Establishing Multiple MP2P LSPs

This section presents studies on the recovery of multiple MP2P LSPs. Two network topologies have been used for this: a “25 node 55 link” network and a “50 node 116 link” network.

6.3.1 Recovery Study for 25 Node Toronto Metropolitan Network

The following network topology was used to investigate recovery times for multiple overlaid MP2P LSPs. This network has 25 nodes and 55 links and is known as the Toronto Metropolitan Network [XIO99].

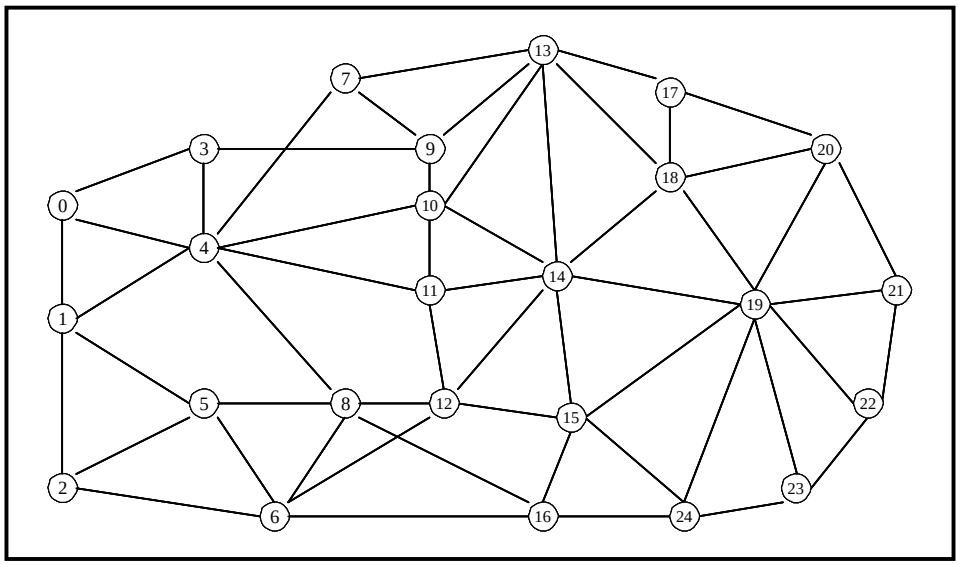


Figure 29:- Toronto Metropolitan Network [XIO99]

In order to test the effects of a single link failure on multiple MP2P LSPs, the link weights in the above network are adjusted such that all MP2P LSPs include the link between nodes 4 and 11, and, in all cases, node 11 is the downstream neighbour to node 4 before this link is failed. Experiments were conducted based upon ten different scenarios; the number of MP2P LSPs present in each scenario was different and ranged from 1 to 10. Nodes 15 through to 24 were used as the egress points in this experiment. In the simulation with only 1 MP2P LSP, only node 15 was used as an egress, nodes were incrementally added for each simulation to act egress points so that for 10 MP2P LSPs nodes 15 to 24 were simultaneously acting as egress points. The time (in milliseconds) to recover all MP2P LSPs was recorded for each scenario. This was

taken to be from the time the link failure was identified (where hardware notification of failure exists this is instant) to the time the last MP2P LSP was re-established through the receipt of the relevant label binding. The results for this set of experiments are shown in figure 30. Confidence intervals are not necessary since there are no random variables in the simulations; in fact the structured reaction of DAM to failure is deterministic.

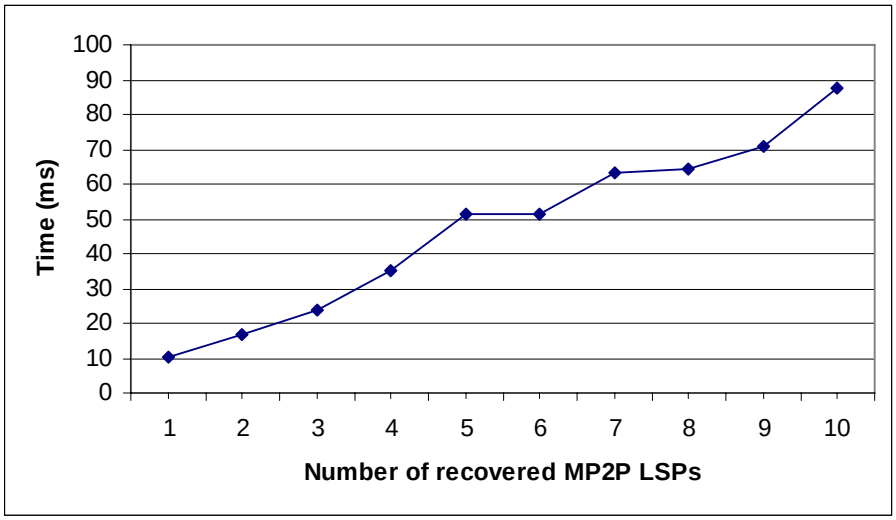


Figure 30:- Graph showing recovery times (ms) for multiple MP2P LSPs in 25 node Toronto Metropolitan Network(fig. 29) with single link failure between node 4 and 11

The results shown in figure 30 illustrate the ability of DAM to support multiple MP2P LSPs in this particular scenario. Recovery times range from 10.5267ms for a single MP2P LSP to 87.5267ms for 10 MP2P LSPs. The results also illustrate the effectiveness of the localised response to failure; the recovery times for multiple MP2P LSP re-establishments are fast and reasonable given the time taken to heal a single MP2P LSP. The results recorded for the re-establishment of five and six MP2P LSPs are identical and indicate that recovery time is not solely related to the number of LSPs requiring re-establishment. A closer inspection of the manner in which recoveries take place can explain the results obtained, particularly the recovery time recorded for recovering six MP2P LSPs. The way that the recovery for six MP2P LSPs takes place is through five directed label requests to node 8 and a single directed label request to node 7. Examining the recovery for five MP2P LSPs in the same manner reveals that five *QueryPropagate* messages are sent to node 8. In this set of simulations the recovery of five and six MP2P LSPs are the only instances in this set of recoveries where the number of *QueryPropagate* messages sent to a single neighbour are identical, this number being five. This is in fact the reason why the times recorded in both these simulations for recovering all MP2P

LSPs are identical. Figure 31 shows recovery times in relation to the number of directed label requests sent to a single neighbour from the node upstream of the point of failure.

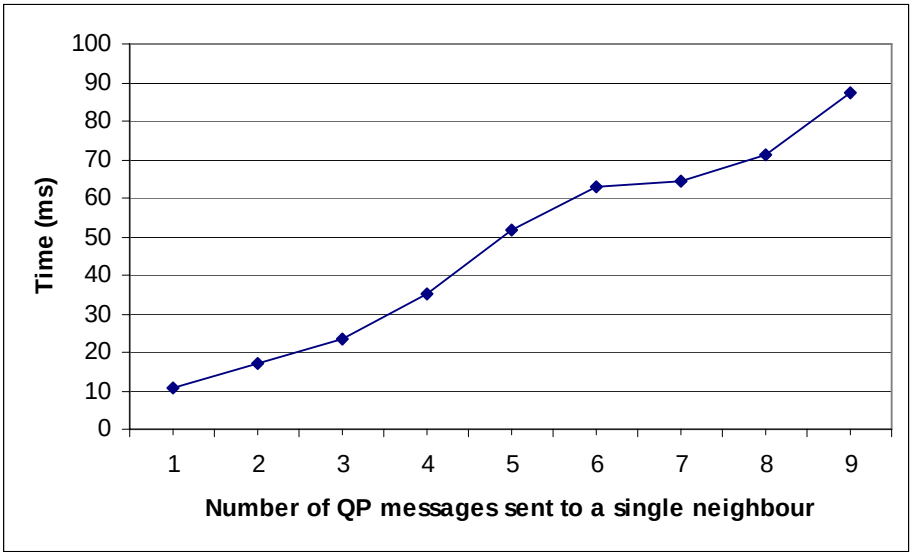


Figure 31:- Graph showing recovery of multiple MP2P LSPs via a single neighbour

All the *QueryPropagate* messages sent in figure 31 relate to directed label requests sent by node 4 to node 8 during the process of failure recovery. The recovery of each individual MP2P LSP in this set of simulations is in the order of 10ms.

Recovery times were investigated using the same method at two other links in the Toronto Metropolitan network shown in figure 29, the link between nodes 8 and 12, and the link between nodes 11 and 14. In the first case the link weights in the network were adjusted such that all MP2P LSPs include the link between nodes 8 and 12, and, in all cases, node 12 is the downstream neighbour to node 8 before this link is failed. In the second case all MP2P LSPs include the link between nodes 11 and 14, and, in all cases, node 14 is the downstream neighbour to node 11 before the link is failed. In both cases experiments were conducted based upon ten different scenarios; the number of MP2P LSPs present in each scenario was different and ranged from 1 to 10. Nodes 15 through to 24 were used as the egress points in both sets of experiments.

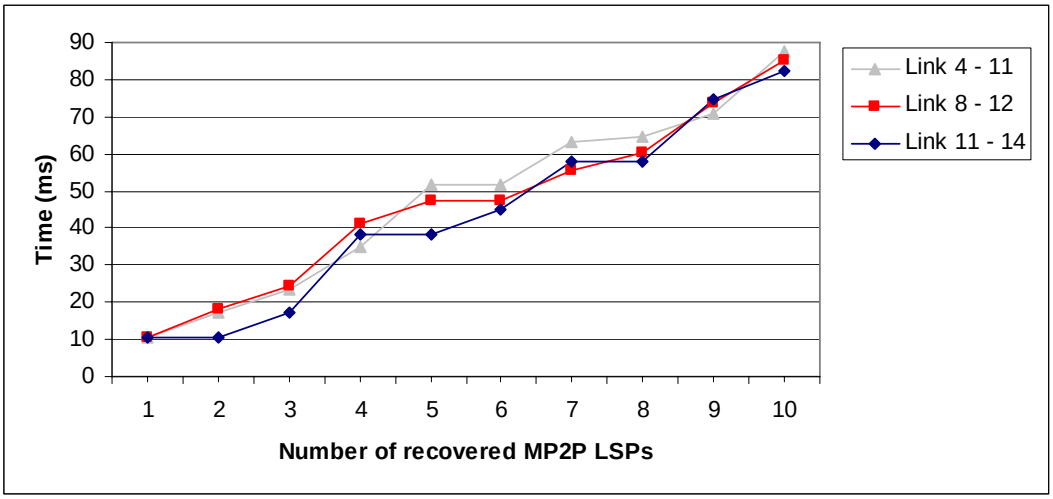


Figure 32:- Graph showing recovery times (ms) for multiple MP2P LSPs in 25 node Toronto Metropolitan Network(fig. 29) with single link failure between nodes 8 and 12 (shown in red) and single link failure between nodes 11 and 14 (shown in blue)

The grey plot in figure 32 shows the results for the link failure between nodes 4 and 11 and is as shown in figure 30. The red plot shown in figure 32 shows recovery times for multiple MP2P LSPs in the Toronto Metropolitan network for a single link failure between nodes 8 and 12. The results for the recovery of 5 and 6 MP2P LSPs are identical; the reason for identical results in both cases is that the maximum number of MP2P LSPs being recovered via a single alternative downstream neighbour is 5.

The blue plot in figure 32 shows the results collected for the recovery of multiple MP2P LSPs when the link is failed between nodes 11 and 14. The results for the recovery of 1 and 2 MP2P LSPs are identical in this case the maximum number of MP2P LSPs being recovered via a single alternative downstream neighbour is 1. Similarly, the results for the recovery of 4 and 5 MP2P LSPs are also identical and the maximum number of MP2P LSPs being recovered via a single alternative downstream neighbour is 4. Finally, the recovery times for 7 and 8 MP2P LSPs are also identical; the maximum number of MP2P LSPs being recovered via a single alternative downstream neighbour is 7 in this case.

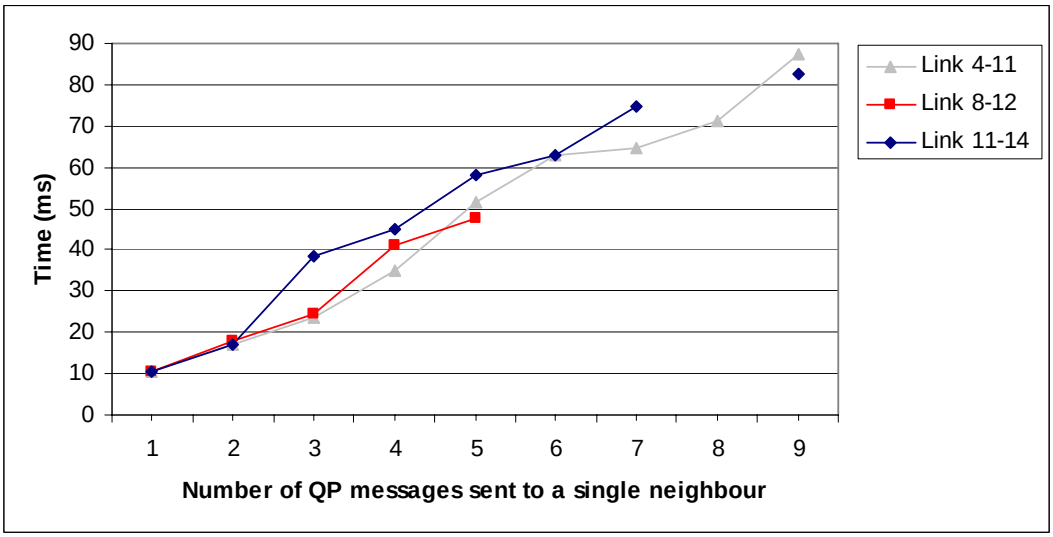


Figure 33:- Graph showing recovery of multiple MP2P LSPs via a single neighbour

Figure 33 shows the recovery of multiple MP2P LSPs via a single neighbour. The grey plot is as shown in figure 31 and are the results collected for node 4 sending *QueryPropagate* messages (directed label requests) to node 8 when the link failure occurs between nodes 4 and 11. The red plots shows recovery times for multiple MP2P LSPs being recovered via node 16 as a result of node 8 sending it *QueryPropagate* messages when the link between nodes 8 and 12 is failed. The blue plot is not contiguous since there is no result for the recovery of 8 MP2P LSPs, the times recorded relate to the recovery of MP2P LSPs via node 12. The directed label requests are sent to node 12 from node 11, link failure occurs between nodes 11 and 14 for this case. The variation in recovery times for multiple MP2P LSPs is attributed to the fact that each individual MP2P LSP is healed using its own specific DAG and as a result signalling overheads are not uniform for the recovery of all MP2P LSPs.

Section 6.1 shows that the signalling required for the recovery of a single MP2P LSP is dependant not only on the degree of the node upstream of the failure but also the local topology, the amount of signalling impacts directly on the recovery time to heal a MP2P LSP after a failure. Based on the results collected in this section (6.3.1) it would appear that node degree and localised topology are not the only factors in determining the recovery time for multiple MP2P LSPs. The results shown in figure 32 indicate that the total number of MP2P LSPs being recovered via a single neighbour appears to also be a factor this is investigated further in Section 6.3.2.

6.3.2 Recovery Study for 50 Node Network

The set of simulations described in section 6.3.1 was repeated for the network topology shown below in figure 34.

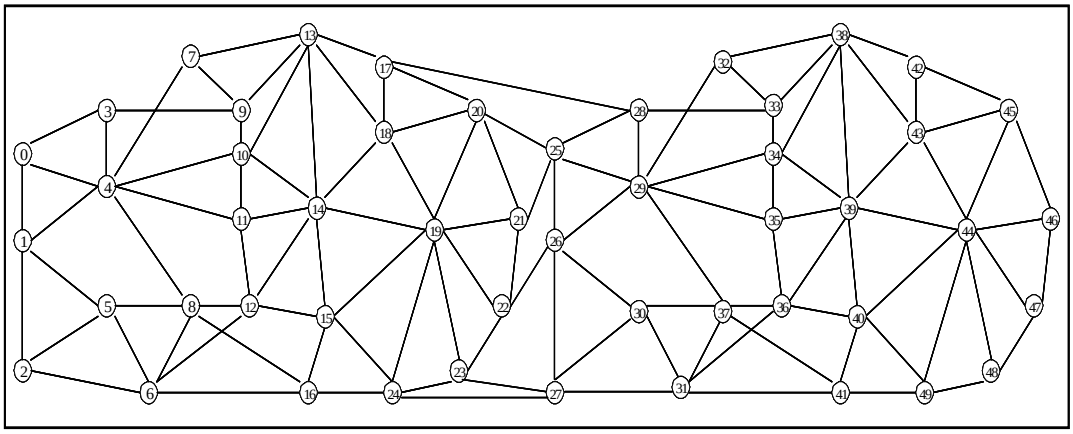


Figure 34:- Example Network 50 Nodes - 116 Links, based on two joined Toronto Metropolitan Networks

As in the previous set of simulations, the link weights in the above network are adjusted such that all MP2P LSPs include the link between nodes 4 and 11, and in all cases node 11 is the downstream neighbour to node 4 before this link is failed. Nodes 25 through to 49 were used as the egress points in this experiment.

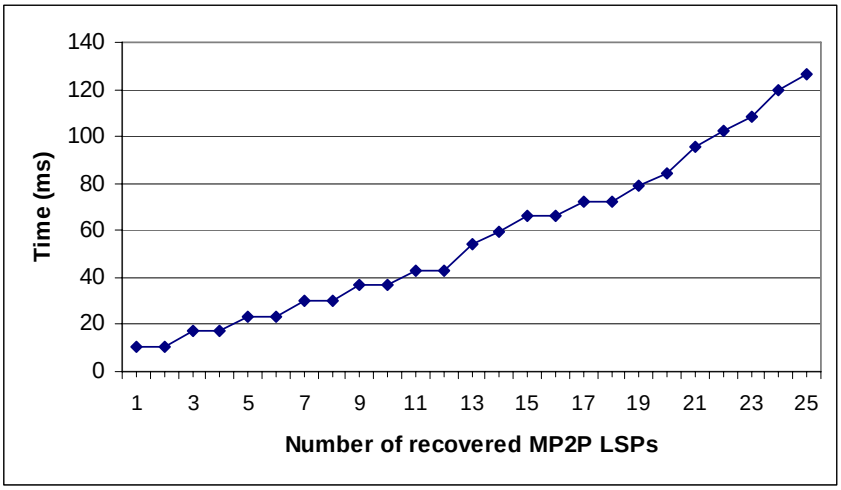


Figure 35:- Graph showing recovery times (ms) for multiple MP2P LSPs in 50 node 116 link network (fig. 34) with single link failure between node 4 and 11

Figure 35 shows the recovery times for 1 to 25 MP2P LSPs in the 50 node network shown in figure 34 with recovery times ranging from 10.5267ms for a single MP2P LSP to 126.5267ms for 25 MP2P LSPs. This result clearly shows the effectiveness of the localised structured reaction to failure, the recovery times are not affected at all by the scale of the network. In the previous set of simulations run on the Toronto metropolitan 25 node network it was noted that recovery times for n and n + 1 MP2P LSPs was identical in some cases. This was due to the maximum number of trees being recovered through a single neighbour being identical; this property is further investigated for the results displayed in figure 35 for the 50 node network topology. The number of *QueryPropagate* messages sent to a single neighbour was examined. This number is also the number MP2P LSPs being recovered via a single neighbour.

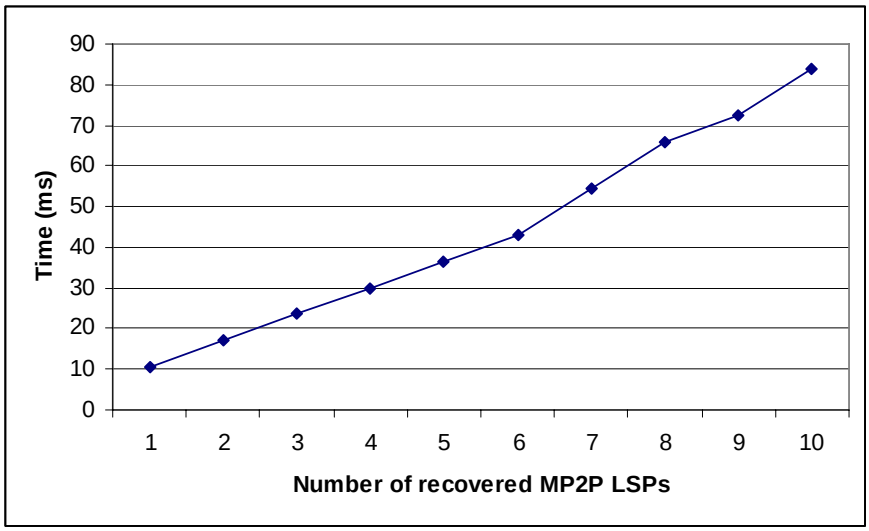


Figure 36:- Graph showing recovery times of multiple MP2P LSPs via node 7 in 50 node network shown in figure 34

All the *QueryPropagate* messages sent in figure 36 relate to directed label requests sent by node 4 to node 7 during the process of failure recovery. Out of a possible 25 MP2P LSPs 10 are recovered through directed label requests to node 7. Recovery times range from 10.5667ms for one MP2P LSP to 84.0667ms for ten MP2P LSPs.

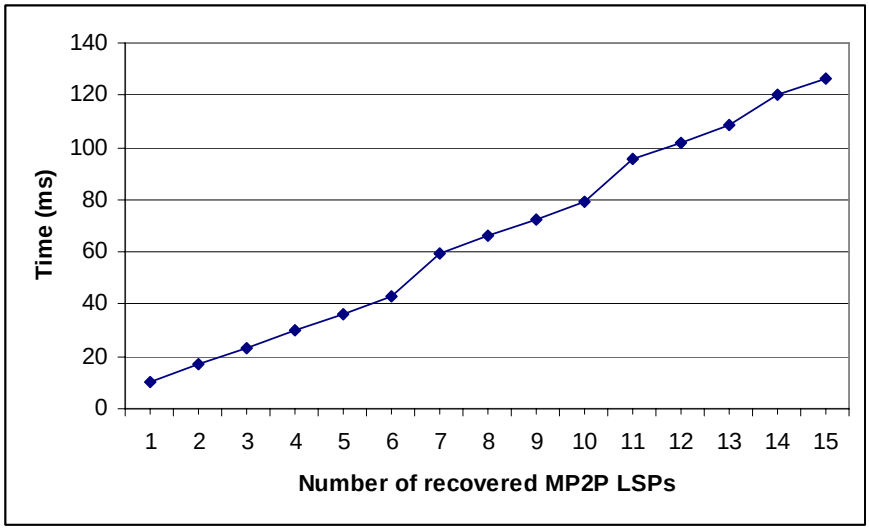


Figure 37:- Graph showing recovery times of multiple MP2P LSPs via node 8 in 50 node network shown in figure 34

Figure 37 shows the recovery times for the MP2P LSPs being recovered using node 8 as a downstream alternative to node 11. Node 4 sends *QueryPropagate* messages to node 8 in order to recover the MP2P LSPs. In figures 36 and 37 the average time taken to recover each MP2P LSP is 8.4ms, these results reinforce the conclusions drawn from the previous set of simulations, that recovery time is not only related to localised topology but also to the number of MP2P LSPs being recovered by directed label requests to a single node.

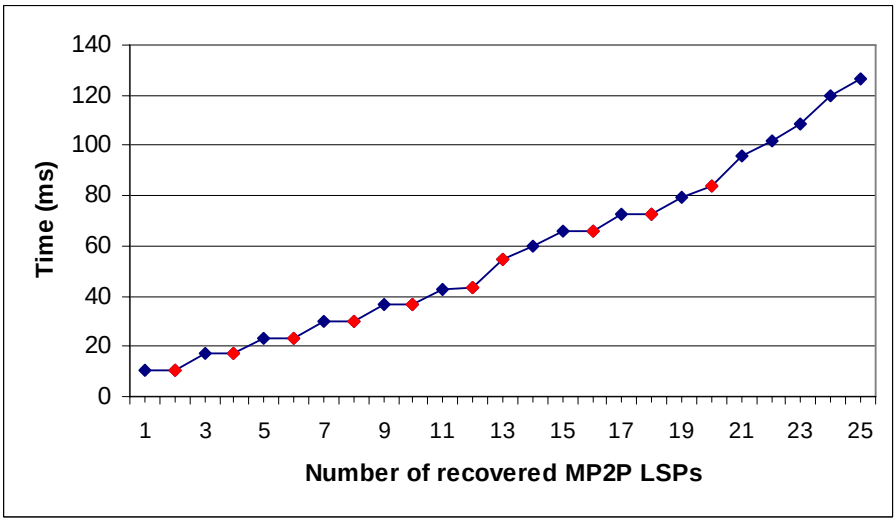


Figure 38:- Graph showing recovery times (ms) for multiple MP2P LSPs, recombining figures 36 and 37

In figure 38 the blue points correspond to the times recorded for recover via node 8 as shown in figure 37. The red points in the graph correspond to recovery times shown in figure 36 for LSP recovery via node 7.

6.3.3 Recovery Time Study for all Links in Toronto Metropolitan Network

The set of simulations conducted in this experiment was run for the 25-node network shown in figure 29. The link costs for all 55 links were set to one in both directions. All 25 nodes were set to act as egress and ingress points. This meant that the network was supporting 25 MP2P LSPs and that all links in the network were utilized for this purpose. This experiment was conducted 55 times with the network set up as described. Each time the simulation was run a different link was failed and the time to recover all MP2P LSPs using the failed link was recorded. Recovery times were recorded for each link in the network until all possible link failures had been simulated. The results recorded were then rounded to the nearest 10ms and the frequency of recovery was plotted.

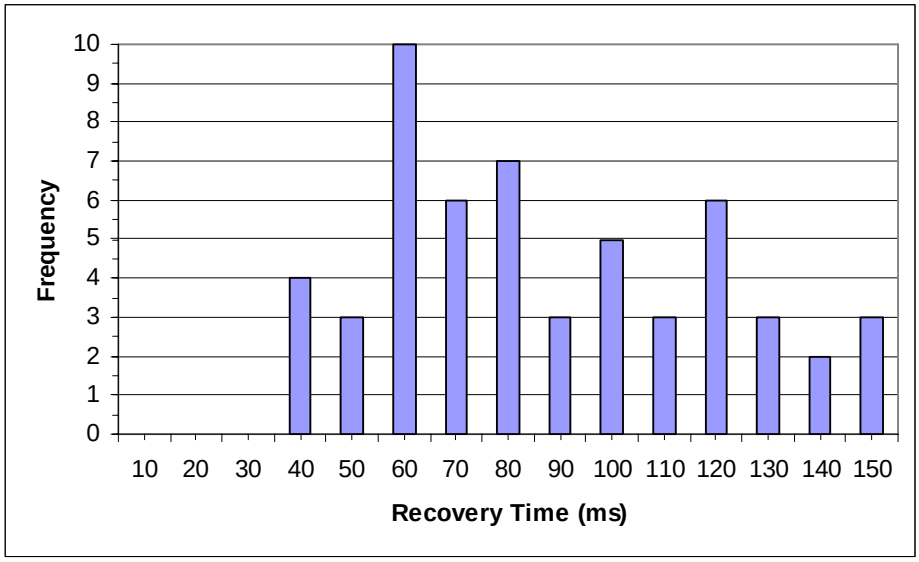


Figure 39:- Graph showing frequency of recovery times

Figure 39 shows the recovery time histogram recorded for the 55 simulations conducted in this experiment. The average time taken to recover all MP2P LSPs using a link was calculated to be 87.1 milliseconds the standard deviation for the results collected was calculated to be 31.89 milliseconds. In this experiment 73% of the recoveries took place between 60 and 120

milliseconds; a range approximately equal to the standard deviation either side of the average recovery time.

6.4 Label Space Reduction

The network topologies shown in figures 22, 23, 29 and 34 have been used for the work described in this section. In addition to these networks, the following network has also been simulated.

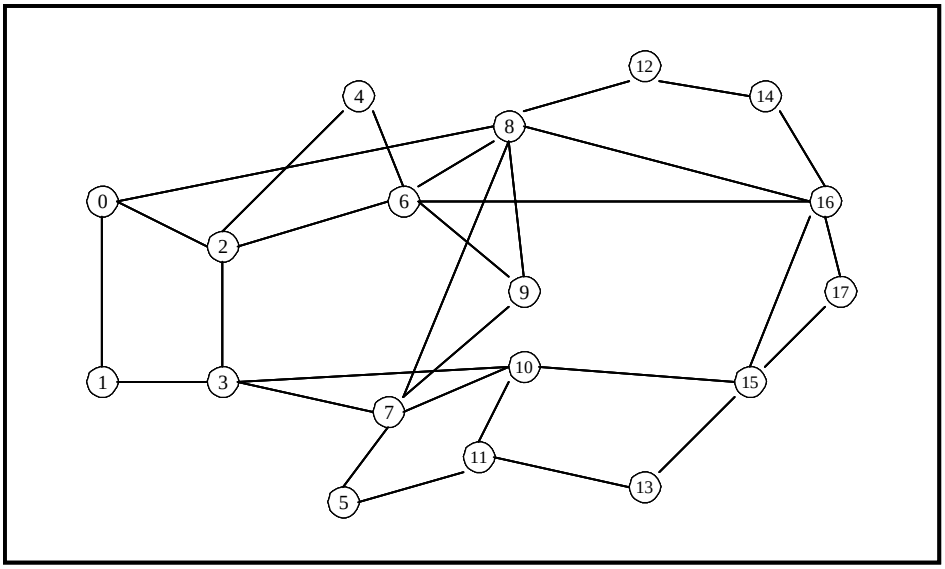


Figure 40:- Bhatnagar Network [BHA03]

Figure 40 shows the test network used by Bhatnagar *et al* [BHA03]. This network comprises 18 nodes and 30 links.

The networks were simulated and the MP2P LSPs were formed, the number of labels required to support these MP2P LSPs was noted. This figure was then contrasted with the number of P2P LSPs and labels required to provide the equivalent level of connectivity in each network scenario that was examined. The following table displays results for all five network scenarios.

Network topology	Number of MP2P LSPs supported	Number of equivalent P2P LSPs	Total number of MP2P LSP Labels	Total number of P2P LSP labels required for equivalent connectivity	Percentage reduction of number labels resulting from MP2P LSP use
New Jersey LATA Network (figure 22)	11	110	45	192	76.57%
NSF Network (figure 23)	14	182	94	391	75.96%
Bhatnagar Network (figure 40)	18	306	165	724	77.21%
Toronto Metropolitan Network (figure 29)	10	240	130	658	80.25%
50 Node 116 Link Network (figure 34)	25	1225	682	4962	86.26%

Table 5:- Label reduction using DAM

Table 5 shows the effectiveness of DAM in reducing the number of labels used in the network. The merging algorithm presented by Bhatnagar *et al* [BHA03] (using the network in figure 40) is able to reduce the number of labels used by 60% for 306 P2P LSPs, a similar scenario using DAM produces a 77.21% reduction in the number of labels used. In the results presented by Bhatnagar *et al* [BHA03] the maximum reduction in the number of labels used is 70% this result is achieved by applying the merging algorithm to 1545 P2P connections in the same network. The smallest reduction in labels using DAM is 75.96% this has been recorded for the NSF Network with an equivalent 182 P2P LSPs. The biggest reduction using DAM is 86.26%; recorded for the “50 node 116 link” network with an equivalent 1225 P2P LSPs.

6.5 Recovery Time Study using DAM-PP

The results presented here relate to simulations that have been carried out using DAM with Partial Pre-Planning (DAM-PP). In this section the recovery times have been studied for the effects of a single link failure on multiple MP2P LSPs. The first set of simulations has been conducted as in section 6.3.1. This was carried out using the Toronto Metropolitan network shown in figure 29 with the link weights in the network adjusted such that all MP2P LSPs include the link between nodes 4 and 11, and, in all cases, node 11 is the downstream neighbour to node 4 before this link is failed. Conducting the simulations in this way ensures that the same node, in this case node 4, initiates all recoveries. As in section 6.3.1 this experiment was conducted with ten different scenarios; the number of MP2P LSPs present in each scenario was different and ranged from 1 to 10. Nodes 15 through to 24 were used as the egress points in this experiment. The time to recover all MP2P LSPs was recorded for each scenario. The results for this experiment are shown in figure 41.

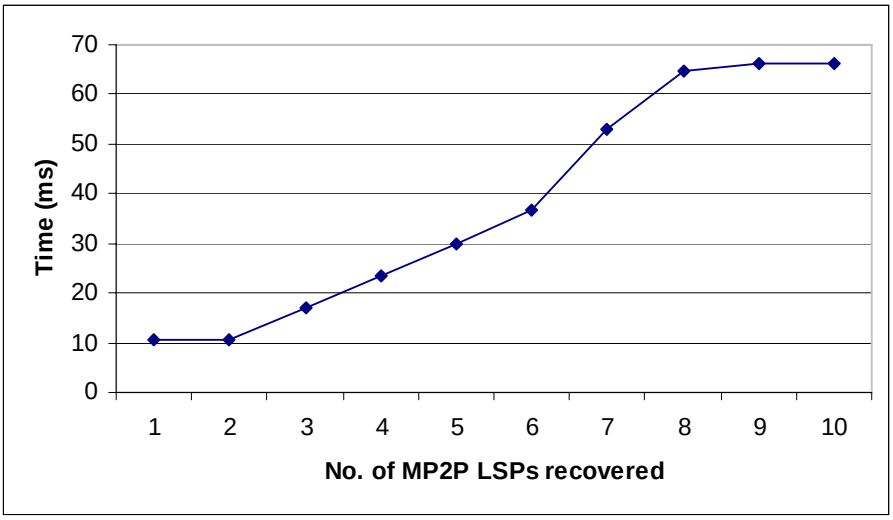


Figure 41:- Graph showing recovery times (ms) using DAM-PP for multiple MP2P LSPs in 25 node Toronto Metropolitan Network(fig. 29) with single link failure between node 4 and 11

The results recorded for this set of simulations show faster recovery time in comparison with the results obtained for the standard DAM implementation. Recovery times in figure 41 range from 10.5267ms to 66.0333ms making DAM-PP about 20ms faster for this particular scenario. For the same set of simulations, DAM achieved recovery times between 10.5267ms and 87.5267ms.

A comparison of the manner in which these recoveries are made reveals that in this experiment using DAM-PP, the maximum number of MP2P LSPs healed via a single node (pre-planned downstream alternative) is 8 this compares to 9 for DAM using the *QueryPropagate* message. This along with the extra signalling of the DAM Height protocol explains why DAM-PP is able to achieve a faster set of recovery times for this specific scenario. The fact that DAM-PP uses a Shortest Path First calculation to select the pre-planned alternative downstream node means that the rerouting around the failure is done via the preferred node in terms of cost. This benefit can be offset by the fact that a single link failure may require multiple MP2P LSPs to be recovered via the same pre-planned alternative downstream node leading to lengthened recovery times. This negative characteristic is not associated with standard DAM as the recovery mechanism supports a DAG, a view of the network that is not based on least cost routing. This is shown clearly in the second set of simulations presented in this section.

The second set of simulations using DAM-PP has been conducted in the same manner as in section 6.3.2 using the "50 node 116 link" network shown in figure 34. As in the previous set of simulations the link weights in the network are adjusted such that all MP2P LSPs include the link between nodes 4 and 11, and in all cases node 11 is the downstream neighbour to node 4 before this link is failed. Nodes 25 through to 49 were used as the egress points giving a maximum of 25 MP2P LSPs in this experiment. The following graphs (figures 42 & 43) show the results recorded.

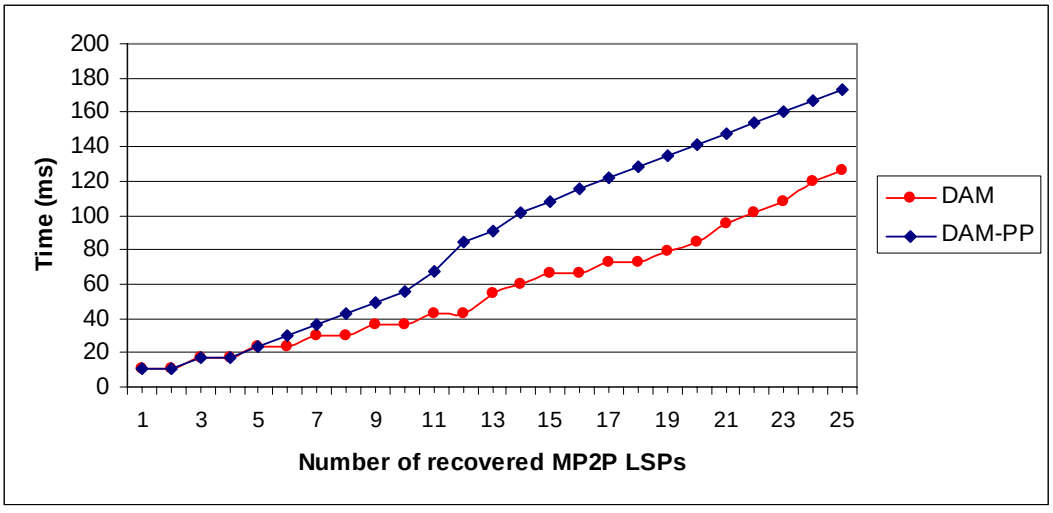


Figure 42:- Graph showing recovery times (ms) for multiple MP2P LSPs using DAM-PP in 50 node 116 link network (fig. 34) with single link failure between node 4 and 11

Figure 42 shows the recovery times using DAM-PP (shown in blue) for 1 to 25 MP2P LSPs in the 50 node network shown in figure 34 with recovery times ranging from 10.5267ms for a single MP2P LSP to 173.5267ms for 25 MP2P LSPs. In this scenario, standard DAM (shown in red in fig 42) performed considerably better achieving recovery times of 10.5267ms for a single MP2P LSP to 126.5267ms of 25 MP2P LSPs almost 50ms faster than for DAM-PP. A comparison of the manner in which these recoveries are made reveals that in this experiment using DAM-PP the maximum number of MP2P LSPs healed via a single node (pre-planned downstream alternative) is 23. This compares to just 15 for DAM using the *QueryPropagate* message.

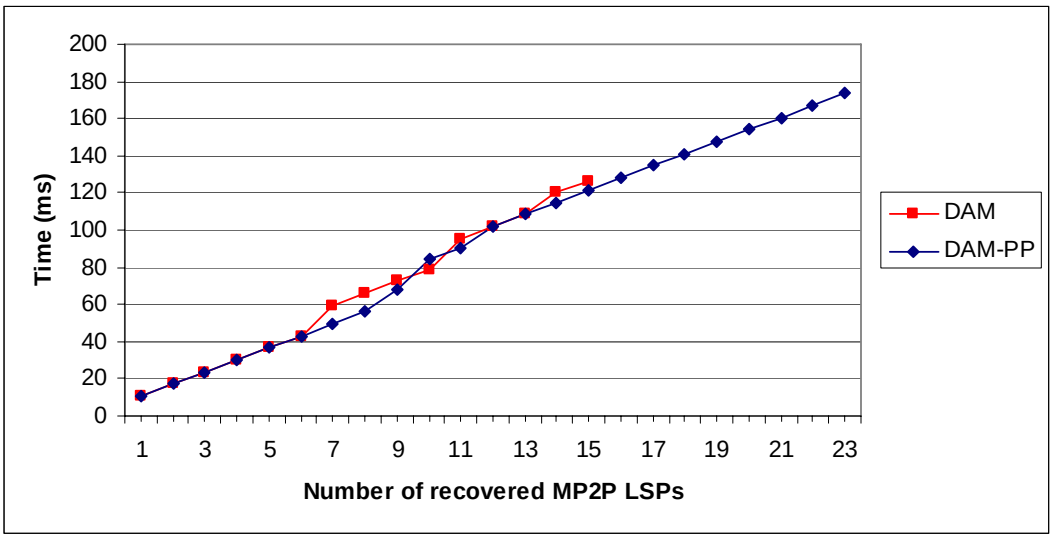


Figure 43:- Graph showing recovery times of multiple MP2P LSPs via node 3 in 50 node network shown in figure 34

In figure 43 the recovery times shown in blue relate to the re-establishment of MP2P LSPs with DAM-PP via the same pre-planned downstream alternative to node 4. In this case the alternative pre-planned neighbour is node 3. Node 3 has been chosen by DAM-PP using a shortest path first calculation. Out of a possible 25 MP2P LSPs, 23 are recovered through node 3, recovery times range from 10.5267ms for one MP2P LSP to 173.5267ms for 25 MP2P LSPs. The results shown in red are for standard DAM and are those given in figure 37 for node 8, chosen by examining the DAG. The number of messages required by standard DAM for recovery is dependant on the degree of the node upstream of the failure and localised topology for this reason the results shown for node 8 are not as regular as those shown for node 3 using DAM-PP. The underlying reason for this irregularity in standard DAM recovery times is that each of the MP2P LSPs is healed using its own DAG that is specific for each particular tree. For each

individual MP2P LSP healed using DAM there are specific signalling overheads that result in the recovery times achieved, these are defined by whether or not the DAG must be reformed, and local topology. Using multiple DAGs to recover multiple MP2P LSPs may lead to recovery of these LSPs via more than one alternative downstream neighbour. The benefit that is derived from maximising the number of alternative downstream neighbours used to heal a set of MP2P LSPs is that the maximum number of LSPs healed via a specific neighbour is reduced and this can help to reduce the total recovery time.

The results collected for DAM-PP show the impact of using least-cost routing as part of a recovery scheme in that the majority of recoveries may as a result take place via a single neighbouring node bringing a negative impact to recovery times. Signalling and processing can be reduced when it more evenly and widely spread amongst viable alternative downstream neighbours, using least-cost routing in choosing such neighbours can be a restriction to achieving this. The benefit of the type of recovery executed by DAM-PP is that the recovery path follows the least cost path. The same is not true for standard DAM since the DAG is used to execute the recovery.

6.6 Simulations using DAM-LB

DAM with Load Balancing provides extensions to the standard implementation of DAM by taking advantage of the DAG to provide a novel method for carrying out load balancing within the network. The simulations conducted here have been carried out using the Toronto Metropolitan network shown in figure 29. All 25 nodes in the network have been made to act as egress / ingress nodes resulting in a total of 25 MP2P LSPs present in the network. The cost of all links in both directions has been set to one. DAM-LB (section 5.2) works by setting a load balancing threshold for this initial implementation this threshold is calculated by :- Z / (degree of the network – 1) where Z is the number of MP2P LSP being supported and Z is not less than the number of nodes in the network. The degree of the Toronto Metropolitan network is 4.4 ((55 ÷ 25) ÷ 2), the load balancing threshold is calculated to be 7.35 but is rounded up to the next whole number and thus set to 8. This scenario was simulated and the numbers of MP2P LSPs flowing in each direction of a link were separately noted. This was done for all 55 links in the network.

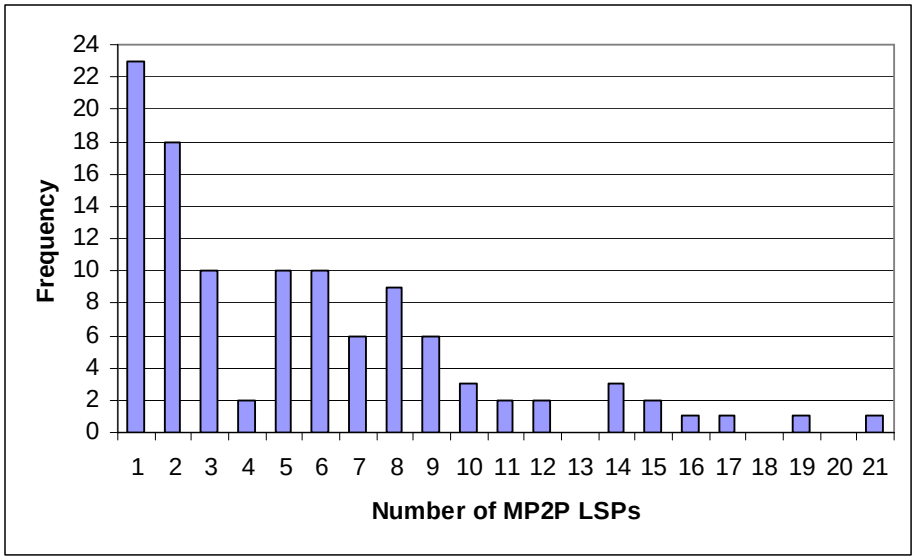


Figure 44:- Graph showing frequency of MP2P LSP flows in one direction on a link using DAM-LB

Figure 44 shows a histogram of the number of MP2P LSP flows in one direction on a link; in the above graph 110 measurements have been noted corresponding to 55 links in both directions. Using DAM-LB in this scenario resulted in 97 out of 110 cases there were 10 or less MP2P LSPs flowing in one direction on a link. The average number of MP2P LSPs flowing in a

single direction on a link was calculated to be 5.42 MP2P LSPs and the standard deviation is calculated to be 4.44 MP2P LSPs. The same measurements were taken for the same scenario using the standard implementation of DAM.

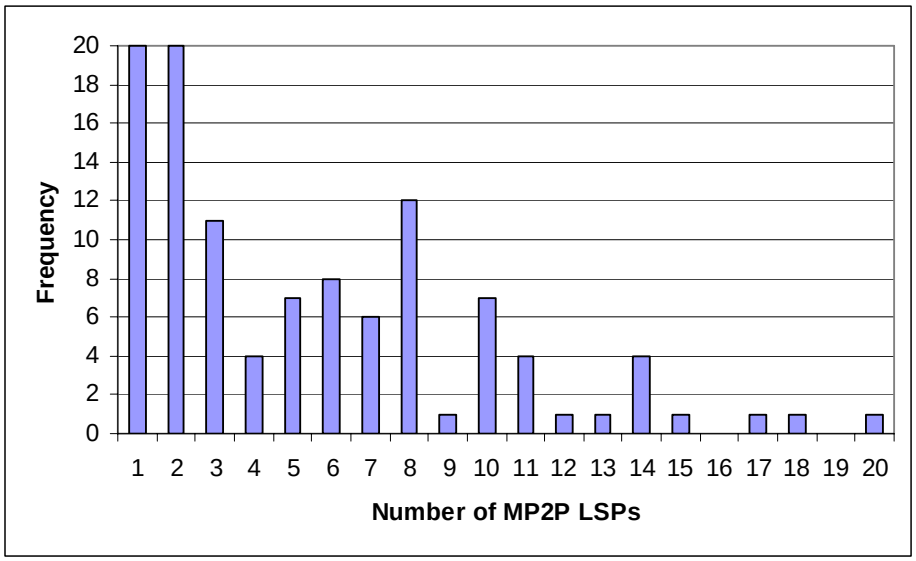


Figure 45:- Graph showing frequency of MP2P LSP flows in one direction on a link using DAM

The results recorded for link usage using DAM are not dissimilar from the results recorded for DAM-LB. Using the standard implementation of DAM in this scenario meant that in 96 out of 110 cases there were 10 or less MP2P LSPs flowing in one direction on a link. The average number of MP2P LSPs flowing in a single direction on a link was calculated to be 5.49 MP2P LSPs and the standard deviation is calculated to be 4.34 MP2P LSPs.

The results recorded for DAM-LB and DAM are very similar; this, however, does not mean DAM-LB is not performing as intended. In this scenario, all link costs are equal and all nodes are acting as egress points this leaves very little scope for load balancing to have any affect on the average number of flows carried in a link.

As a further investigation of DAM-LB the experiment described in section 6.3.3 was repeated using DAM-LB. This was done in order to study the impact on recovery times that the redistribution of flows had in this network scenario. This experiment was conducted 55 times with the network set up as described using equal link costs and all nodes acting as egress / ingress points. Each time the simulation was run a different link was failed and the time to recover all MP2P LSPs using the failed link was recorded. Recovery times were recorded for

each link in the network until all possible link failures had been simulated. The results recorded were then rounded to the nearest 10ms and the frequency of recovery was plotted.

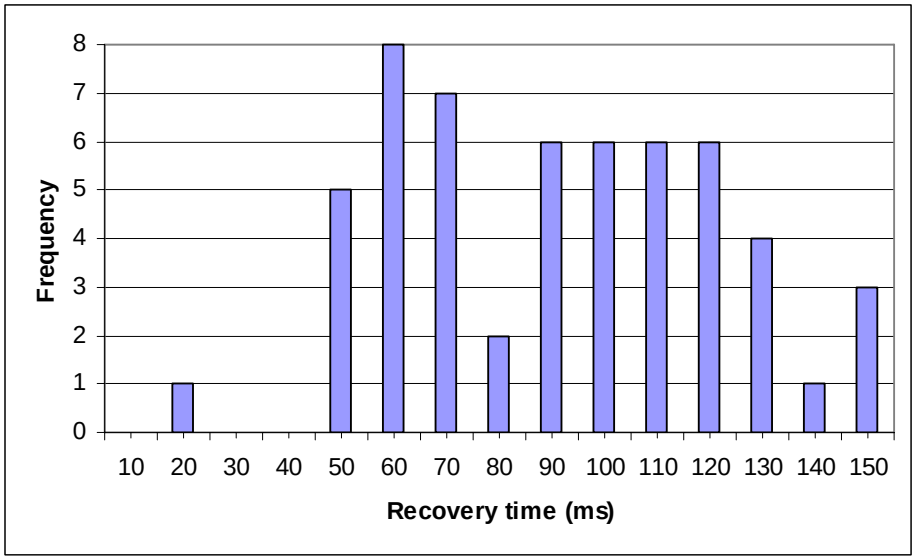


Figure 46:- Graph showing frequency of recovery times for DAM-LB

Figure 46 shows the recovery time histogram for the 55 simulations conducted in this experiment. The average recovery time recorded for DAM-LB in this scenario is 91.45 milliseconds and the standard deviation was calculated to be 31.08 milliseconds again these results are similar to those recorded for DAM. In the time range between 60 and 120 milliseconds 74.5% of recoveries took place; this is only marginally better than the result recorded for DAM.

Chapter 7

7 Discussion & Future Work

The boundaries between exclusively private networking and shared networks are becoming increasingly blurred; this observation is increasingly apparent as private companies and individuals continue to subscribe to services that are provided using the shared backbone of service providers. The challenge that existed until recently was the trade-off between the flexibility provided by IP and the speed and QoS guarantees of technologies such as ATM and Frame Relay. The key point for consideration here is the fact that the flexibility provided by IP at Layer 3 is due in part to IP being connectionless. However, this leads to problematic support for services developed with the connection-orientated approach of Layer 2 technologies such as ATM and Frame Relay. More recently better integration of these two layers has been enabled through the use of MPLS acting as a shim layer between Layer 2 and Layer 3. MPLS has allowed service providers to offer customers a wider range of services than before by allowing them more flexibility in the way they manage their resources. Nowhere is this more prevalent than in the gradual decline of leased line services in favour of MPLS VPNs [CIS04]. This is no coincidence as cost savings to the customer ultimately dictate this transition [SPR04], in fact Cisco give real examples where cost savings are in the region of 61% [CIS04]. MPLS VPNs allow a service provider to manage their resources in a more cost effective way [BOA02] [CIS04a] [SPR04] and, as a result, these savings can be passed on to the customer. The work presented in this thesis is of particular importance to this research area as additional tools in the hands of service providers can translate into financial benefits for both providers and customers. This thesis has presented novel techniques for the management of an MPLS network.

Chapter 1 has introduced the main theme of this work, that is MP2P LSPs within the context of MPLS and recommendations made by the IETF in [RFC3031]. The introduction goes on to identify the benefits that can be achieved through the use of MP2P LSPs highlighting some of these benefits as motivating factors for this work. The work and merits of other research efforts in this field are also introduced as they add impetus to the work described here in that the differences in approach help to highlight the novel contributions as well as providing support for the motivating factors that fuelled this thesis. Having highlighted the motivation for this thesis, “DAM” is introduced.

Chapter 2 entitled Research Landscape builds on the introduction given in chapter 1 by examining the relevant elements in greater depth. Traditional IP routing is discussed as well as routing protocols such as OSPF and BGP that are used in determining the path that packets follow through a network. The manner in which datagrams are processed as they traverse a network is also discussed. In conventional IP routing the forwarding decision is made at each intermediate node on a hop-by-hop basis. This forwarding decision is made on the basis of a longest prefix match that can be cumbersome in that each packet is dealt with individually. Furthermore, packets may take different routes to the same destination; IP delivers them by a best effort service. Solutions such as ATM and Frame Relay have been used to reduce the processing overhead within the provider core, enabling high-speed data transfer by simplifying the forwarding process. However these Layer 2 solutions are purely connection-orientated and do not afford the service provider the flexibility associated with conventional IP forwarding.

A solution that is able to provide the benefits of fast forwarding whilst providing greater flexibility than purely connection-orientated technologies is MPLS. A description of the basic features of MPLS and its architecture has been given along with what it has to offer over conventional routing. MPLS is a framework that decouples route selection from the forwarding mechanism and, by doing so, allows those employing it to tailor their own traffic engineering solutions. In MPLS, the manner in which packets can be assigned to a FEC results in much of the flexibility it is able to offer those that deploy it. Unlike conventional routing, this assignment is not based solely on the contents of the packet header but can be done using other information such as the packets entry or exit point to the MPLS domain. Furthermore, examination of the packet header, if necessary, need only happen once at the entry point to the MPLS domain. MPLS provides a framework for fast forwarding based on short fixed-length labels that are only locally significant; as a result there is no need to have a global addressing scheme. The MPLS architecture allows aggregation to take place with respect to FECs. This is a powerful feature that can be leveraged to aid in traffic engineering. The MPLS framework presents a TE opportunity in that it has to offer a rich set of features. Within this framework and central to the work described in this thesis is the concept of MP2P LSPs that was first described in the draft versions of [RFC3031].

MP2P LSPs offer a more scalable solution within an MPLS domain as they can reduce the amount of label space required to support full connectivity. This is particularly important as when the magnitude of the MPLS domain increases its management can present a very real challenge. The importance of this challenge is highlighted by the example given by Bhatnagar *et*

al [BHA03] where 50000 * 49999 P2P LSPs would be required to support a VPN with 50000 CE devices. The use of MP2P LSPs over P2P LSPs in such an example not only presents a dramatic reduction in the number of LSPs required to support all the CE devices, the very nature of MP2P LSPs provides an architecture in which multiple service levels are more easily implemented and managed. The result is a provider core with a volume of LSPs that is more manageable whilst providing an architecture that can still support multiple service levels. This is particularly important in the context of VPNs.

In the discussion of VPNs the most important solutions have been discussed and even though standards are currently under ongoing development their importance to service providers cannot be overstated, as interest in this area is continually growing reflecting the shift from leased line to “MPLS VPN” service provision [CIS04][SPR04]. What such discussions highlight is that regardless of at which layer the forwarding decision is made in such VPNs, all the discussed solutions use MPLS as the generic tunnelling mechanism with which to forward data. It therefore follows that effective traffic engineering using MP2P LSPs over P2P LSPs not only alleviates the problems associated with a fully meshed arrangement (which are still applicable to solutions limiting the number of LSPs in the core through the use of pseudo wires) it ultimately translates into more profitable service provider networks that are able to operate more effectively. This is evidenced by the cost savings and management benefits brought to the service provider by MPLS VPNs [CIS04][SPR04].

Whilst maximising the utilisation of their resources, service providers need to be able to shield a customer from service degradation. Setting aside resources to meet a specific customer need can help in achieving this, but in the event of link or node failure within the provider core, timely and cost effective restoration becomes the key issue for MPLS networks. In this thesis MPLS recovery schemes have been discussed most of which centre around protection switching. Protection switching requires the use of pre-established backup paths. This impacts directly on how costly such solutions are with some strategies setting aside resources exclusively for this purpose. Another drawback of protection switching is the distance failure notification messages must travel to get to the point of repair. Even so, this type of restoration is popular because recovery times can be very fast. The alternative to protection switching is re-routing but it remains largely unappealing because it tends to rely on the re-convergence of higher layer protocols and, as a result, recovery times are generally longer. What makes re-routing attractive is its ability to provide simple cost-effective recovery where resources are only committed when a failure occurs. In order to make re-routing a practical alternative to protection switching for

MPLS networks, recovery times must be reduced. This entails employing methods that are not reliant on the re-convergence of higher layer routing protocols, Ahn *et al* [AHN02] present such a scheme.

Four schemes for the creation of MP2P LSPs are presented in Chapter 2. Of greatest relevance to this thesis are those proposed by Saito *et al* [SAI00] and Bhatnagar *et al* [BHA03]. Both of these solutions rely on the appropriate selection of P2P LSPs as input to their algorithms. The scheme proposed by Saito *et al* [SAI00] formulates the MP2P tree creation problem as a 0 –1 integer programming problem. This is acknowledged to be in general NP complete. Bhatnagar *et al* [BHA03] state that it cannot be used frequently because of the processing overheads associated with its complexity. The scheme proposed by Bhatnagar *et al* [BHA02][BHA03] is described as a simple polynomial time heuristic. This scheme is comprised of a simple path-merging algorithm coupled with a path selection heuristic for the merging of paths into subsequent trees. As a result of the path selection heuristic Bhatnagar *et al* [BHA03] show that the path-merging algorithm can lead to sub-optimal results. Neither scheme fully addresses the issue of resilience. In fact, only the scheme by Saito *et al* [SAI00] makes reference to this by proposing the use of complete MP2P LSPs to act as back up trees. Chapter 2 also makes the case that MP2P LSPs are an important TE opportunity and that there is an absence of research in this area in that no scheme to-date has proposed the dynamic creation of MP2P LSPs or that provides a dedicated resilience scheme. In response, this thesis presents the **D**ynamically **A**daptive **M**ultipoint to Point (**DAM**) scheme and its associated protocol for the creation and maintenance of MP2P LSPs.

Chapter 3 describes the principles that make up DAM. LRR features as part of DAM's structured reaction to failure. LRR encompasses a set of algorithms that were originally designed for use in adhoc networks. DAM leverages the partial reversal method [GAF81], a highly adaptive LRR algorithm in the provision of its dedicated resilience scheme. In Chapter 3 two versions of DAM are presented; these are the distributed version and the edge orientated versions of DAM. Common to both versions are their design goals that require that MP2P LSPs to be created and maintained dynamically whilst providing a dedicated resilience scheme that can lead to fast localised healing in a structured and reliable manner.

The first version of DAM to be described in this thesis is the distributed version, it is founded on the fact that an MP2P LSP can be considered to be an inverted tree with the egress node acting as the root. This notion of an inverted tree led to a survey of the methods employed in

multicast routing and as a result RPF was chosen as the basis with which to form MP2P LSPs. The distributed version of DAM uses the RPF algorithm in its creation of the MP2P LSPs but forms sink-based trees as opposed to source based trees for which the algorithm was originally intended. The Reverse Path Forwarding algorithm maintains a reverse path table in which a preferred interface is marked; in DAM this table is maintained by conducting SPF calculations based on the OSPF link state database. In the edge-orientated version of DAM the MP2P tree calculation is pushed to the edge of the domain. This is done by using an adapted version of Dijkstra's algorithm and encoding the tree into signalling used in the initialisation phase of the protocol. As a result, processing in the core of the network is reduced. Chapter 3 also describes this adapted version of the Dijkstra's algorithm that is used in calculating the reverse shortest path first tree. MP2P creation and maintenance is discussed for both versions of DAM along with how a new node may join an existing tree. Common to both versions of DAM is that each MP2P LSP has a corresponding DAG that is used to maintain and provide resilience for the MP2P LSP. The heights in this DAG are used as a guide for the joining process as well as for the healing of the tree. Chapter 4 goes on to give a specification describing the details necessary for an implementation of the DAM protocol. The difference between the two versions of DAM described in chapter 3 is in the formation of the MP2P LSPs; however, since both use the OSPF link state database in the calculation of the tree the actual resulting topology is identical. The strategies employed in both versions of DAM for the maintenance of the MP2P LSPs rely on the use of a DAG and the partial reversal method. The details given in chapter 4 are specifically for an implementation of the edge-orientated version of DAM. Nevertheless. This specification is easily adapted for an implementation of the distributed version. Implementation of the edge-orientated version was chosen over the distributed version because of its simplicity in the formation of the MP2P LSPs; the fact that the calculation is conducted at the network edge was also deemed to be advantageous. The methodology undertaken in moving from a protocol specification to a working prototype in OPNET is also described in chapter 4.

This thesis has also presented two further extensions to the work presented in chapter 4, these are described in depth in chapter 5. The first that has been presented in chapter 5 is DAM with Partial Pre-Planning. This extension to the standard specification in chapter 4 was designed in order to examine an alternative way of implementing localised healing using DAM. The motivation for designing an alternative scheme for localised healing in this context was to aid in assessment of the strategy adopted by the original specification of DAM. The localised recovery strategy employed by DAM-PP uses pre-planned alternative neighbours whereby each node participating in a tree calculates an alternative neighbour to be used should the current

downstream neighbour become unusable. This strategy is considered to be partial pre-planning since a complete alternative backup branch on to the tree is not calculated. This alternative downstream neighbour is calculated by artificially inflating the link cost to the current downstream neighbour and then locally conducting a reverse Dijkstra calculation based on the amended link costs.

The second extension to DAM presented in chapter 5 is a novel method for performing load balancing for multiple overlaid trees called DAM with Load Balancing (DAM-LB). In DAM-LB the recovery strategy is the same as for the standard specification described in chapter 4. During the creation of MP2P LSPs, sections of the LSP may be re-routed where possible so that overall link utilisation decreases. This method takes advantage of the acyclic nature of the DAG that is essentially a graph of multipaths to a particular destination. The fact that this graph is acyclic allows sections of an MP2P LSP to be re-routed locally without other measures being put in place to avoid loops. DAM-LB uses signalling to perform this local re-routing. This involves making directed label requests along the DAG on the basis of decisions made locally with respect to specific trees. Nodes receiving such label requests can choose to respond only if they have resources to accommodate the additional traffic.

In this thesis, chapter 6 presents both simulation and analysis of the work presented in the previous chapters. In section 6.1 an analysis of message exchange is presented. This analysis examines the message exchange made between nodes in order to perform restoration in response to a single link failure. The network used for this analysis was the New Jersey LATA Network this network has 11 nodes and 23 links. In this particular example the results obtained for DAM show the consistency in the number of messages required to perform restoration following a link failure. This feature of DAM was observed with other network topologies. Similar results were found by analysis of the NSF and Toronto Metropolitan networks (the NSF network has lower node connectivity and the Toronto Metropolitan greater node connectivity in comparison to the New Jersey LATA Network). In the analysis in section 6.1 the average number of messages required to re–establish connectivity using DAM is 2.5, the standard deviation was calculated to be 0.9 for this set of results. In the message exchange analysis for the New Jersey LATA Network using DAM it is important to also examine the number of messages required to reconstitute the DAG, the average for this is 0.3 and the standard deviation is calculated to be 0.5. In most cases zero messages are required to reconstitute the DAG. In the cases where messages are required, for this to happen only one message is required. The main factor contributing to this is network topology. In the message exchange analysis for the NSF

network using DAM the average number of messages required to reconstitute the DAG is 0.89 and the standard deviation is calculated to be 0.99. The less well-connected a node is, the more likely it is to have only a single outgoing link in the DAG. In contrast, a node that has many connected links has greater potential to have multiple outgoing links in the DAG. The average number of messages required to obtain a label binding using DAM for both the New Jersey LATA and NSF networks is 2.2 and the standard deviation is calculated to be 0.6. In both examples all but one case require two messages to obtain the label binding. This consistency is attributed to the fact that in most cases the partial reversal method is not employed in order to reconstitute the DAG and since all nodes are members of the tree, no intermediate nodes are used in order to obtain this label binding. Naturally the use of intermediate nodes to obtain a label binding requires more messages to be used. Section 6.1 concludes with a discussion and example on the signalling impact of node degree and localised topology near the point of failure. The message exchange analysis demonstrates how the locally structured recovery mechanism employed by DAM is able to limit the signalling required to recover from a failure and that the number of messages required to do this is similar for the majority of single link failures in the examples presented. The same cannot be said for reactively signalling a new branch or for 1:1 protection switching, both presented in section 6.1. For this example, signalling a new backup branch to recover the affected section of the tree required on average 6.5 messages and the standard deviation was calculated to be 1.1 messages for the 10 distinct link failures examined. In the same example the use of 1:1 protection switching required on average 2.9 messages with a standard deviation of 3.3 messages. The use of both these methods means that the number of messages will on average grow as the size of the network increases. A point that has been identified by others [HUA02] is the delay that can be experienced in protection switching when signalling the failure to the point of repair. This observation accounts for the relatively high standard deviation calculated for 1:1 protection switching in this example. The variation shown here reflects the various distances failure notifications must travel in this example, it follows that as the size of the network grows so does this variation.

In this thesis, section 6.2 presents simulation results for single link failures using the OPNET models for the standard implementation of DAM described in chapter 4. The simulations were run in the example network shown in section 6.2 and recovery times recorded for all possible single link failures. In this section attention has been given to the case in which the single link failure requires an upstream neighbour of the affected node to become its downstream neighbour. This process requires the use of the partial reversal method to reconstitute the DAG followed by directed label requests from the affected node and its neighbour. This type of

recovery has been described in section 6.2 as “non-trivial” because it requires a greater amount of signalling to take place for the recovery to complete. The simplest type of recovery only requires two messages, a directed label request and its reply or at most three messages when a height advertisement is sent prior to the label request and reply. In this section the “non-trivial” recovery has been described in detail and the recovery time noted. The simulations presented in this section show that DAM works effectively in the test network and that recovery times were similar for all types of single link failures in this network. Each message exchanged between the nodes during the restoration was observed in the simulator and confirmed the correct functioning of the protocol. The recovery times recorded for this test network ranged between 10.5267 and 10.5533 milliseconds. Recovery times observed at the affected node for the “non-trivial” type of recovery was 10.5533 milliseconds; this is only marginally longer than the 10.5267 milliseconds recorded for the simplest type of recovery. The messages employed by DAM for recovery are very small the *QueryPropagate* message is 256 bits and the *HeartBeat* message is 416 bits. These sizes mean that signalling delays account for 10.5 ms, approximately 99% of the recovery times recorded as experiments were conducted using point-to-point links between nodes with a slow data rate of 64000 bits/s. The remainder of the recovery times is made up of IP packet forwarding delay, each delay accounting for 0.00667 ms per packet handled. Detailed timing diagrams are given for both the simplest and “non-trivial” recovery examples (figures 27 and 28). More realistic link speeds of 1Gigabits/s would result in signalling delays in the region of 0.3 microseconds. Propagation delay in optical fibre is in the region of 0.5 ms over a distance of 100 kilometres. Realistic IP packet forwarding delay would be negligible. Currently typical delays are in the order of nanoseconds per packet. It is therefore probable that simulations based on a more realistic scenario with a network spanning hundreds of kilometres, using link speeds of a least 1 Gigabits/s with propagation delays modelled would yield much faster results.

Sections 6.1 and 6.2 examine the recovery of a single MP2P LSP after a link failure and in doing so have shown the functionality of DAM in performing successful restoration. The impact of node degree and local topology on restoration has also been considered. The recovery of a single MP2P LSP is only a test case. DAM has been designed to create and maintain multiple MP2P LSPs. Section 6.3 presents a series of simulations that examine the recovery of multiple MP2P LSPs. The first set of simulations is presented in section 6.3.1; the simulations were run on a “25 node 55 link” network. In this set of simulations ten separate experiments were conducted and in each case the number of MP2P LSPs was increased by one so that results could be collected for one MP2P LSP through to ten MP2P LSPs. In order for the results to

reflect the growing number of LSPs in each simulation the experiments were set up so that all MP2P LSPs used the link being failed in the same direction (see section 6.3.1 “in all cases node X is the downstream neighbour to node Y before this link is failed”). Recovery times ranged from 10.5267ms for a single MP2P LSP to 87.5267ms for 10 MP2P LSPs. This set of results shows that the time taken to recover from a link failure is dependant on the number of MP2P LSPs that must be recovered as a result of the failure. The recovery times recorded for each set of ten simulations conducted show the effectiveness of DAM in supporting multiple LSPs. The advantages of the localised structured reaction to failure are also shown in that recovery times are not affected at all by the scale of the network. The variation in recovery times for multiple MP2P LSPs observed in section 6.3.1 for the three distinct link failures is attributed to the fact that each individual MP2P LSP is healed using its own specific DAG and as a result signalling overheads are not uniform for the recovery of all MP2P LSPs. This set of experiments has also produced results that indicate the recovery time to be related to the maximum number of MP2P LSPs being recovered via a single neighbour. This observation is illustrated more conclusively in section 6.3.2. In section 6.3.2 an experiment was devised in the same manner as in section 6.3.1 however simulations were run on a “50 node 116 link” network with the number of MP2P LSPs simultaneously using the same link growing incrementally from 1 to 25 with each separate simulation run. The results from this series of simulations again show the advantages of the localised structured reaction employed by DAM in response to failure; the scale of the network does not affect recovery times. This observation is made further clearer when comparing recovery times recorded in sections 6.2, 6.3.1 and 6.3.2, recovery times for the restoration of a single MP2P LSP are similar for the three different topologies used. The results in section 6.3.2 illustrate that recovery times with DAM are related to the maximum number of MP2P LSPs being recovered through a single neighbour in addition to degree of the node upstream of the failure and local topology.

Sections 6.3.1 and 6.3.2 have examined the failure of a particular link in order to show the affects this has on the recovery of multiple MP2P LSPs. Section 6.3.3 then examines recovery times for all links in the network. In section 6.3.3, a “25 node 55 link” network was used, all link costs were set to one and all nodes were made to act as egress / ingress nodes. The network configuration used in section 6.3.3 results in all links being used and total of 25 MP2P LSPs supported by the network. Each time the simulation was run a different link was failed and the time to recover all MP2P LSPs using the failed link was recorded. Recovery times were recorded for each of the 55 links in the network until all possible link failures had been simulated. Unlike in the previous experiments in section 6.3.1 and section 6.3.2 the failure of a

link meant recoveries were executed at both ends of the link failure. This was true in all cases and is due to the overlaid MP2P LSPs occupying all possible link directions in this scenario. The average time taken to recover all MP2P LSPs using a link was calculated to be 87.1 milliseconds the standard deviation for the results collected was calculated to be 31.89 milliseconds. In this experiment, 73% of the recoveries take place between 60 and 120 milliseconds a range approximately equal to the standard deviation either side of the average recovery time. Recovery times outside of the range 60 to 120 milliseconds can be attributed to specific link utilisation and in some cases the position of a link in the network topology. For example, links closer to the edge of the network are connected to nodes that are less well connected than others in the network. In such cases recoveries may require more signalling as described in section 6.2. The range of recovery times observed in this set of simulations is comparable to the results that were observed for the specific single link failures described in sections 6.3.1 and 6.3.2; moreover it shows the consistency of DAM in its ability to make fast recoveries in response to a failure.

In this thesis section 6.4 examines the label space reduction achieved by DAM in five different network topologies. Each network was simulated and the MP2P LSPs were formed using DAM. The number of labels required to support these MP2P LSPs was then noted. A comparison was made between this figure and the number of labels required to achieve the same level of connectivity with P2P LSPs. The label space reduction achieved using DAM ranged from 76% to 86% depending on the topology and the level of connectivity in each scenario. In the result obtained for the Bhatnagar network a direct comparison could be made with the work presented by Bhatnagar *et al* [BHA03]. Here the merging algorithm presented by Bhatnagar *et al* [BHA03] is able to reduce the number of labels used by 60% for 306 P2P LSPs, a similar scenario using DAM produces a 77.21% reduction in the number of labels used. In the scheme of by Bhatnagar *et al* [BHA03] the maximum label reduction is 70% this figure is cited for the application of the merging algorithm to 1545 P2P connections. The results obtained for DAM show that good label space reduction can be achieved in the networks presented in section 6.4. Moreover the scale of this reduction is similar to that achieved by Bhatnagar *et al* [BHA03] and in the case where a direct comparison could be made, DAM achieved greater label space reduction.

This thesis then presented extensions to the standard specification of DAM; in section 6.5 simulations have been carried out using DAM with Partial Pre-Planning (DAM-PP). These simulations examine recovery times for DAM-PP. The experiments that are presented in this

section are set up in the same manner as those described in section 6.3.1 and section 6.3.2. In the first set of simulations, conducted on the “25 node 55 link” network, the results collected with DAM-PP show faster recovery than those obtained for the same network scenario using standard DAM. These results were closely scrutinised and it was found that in this specific scenario, the maximum number of MP2P LSPs recovered via a single neighbour was less for DAM-PP than for DAM and this was the major factor in DAM-PP achieving faster restoration times in this example. DAM-PP also requires less signalling than DAM as it does not employ the DAM Height protocol this, however, does not have a significant impact on the performance. An important point for consideration is the fact that DAM-PP uses a shortest path first calculation in the selection of the pre-planned alternative downstream node that is used to re-route when there is a failure. Whilst this ensures least cost recovery is performed, this can lead to slower recovery times as the re-establishment of MP2P LSPs may focus upon a particular node near the point of failure. The second set of simulations shows this phenomenon. The recovery times observed for DAM-PP are thus impaired and are significantly slower than those achieved with DAM (see figure 42) for the same example. In this scenario the maximum number of MP2P LSPs healed via a single node is 23 for DAM-PP this compares to just 15 for DAM further illustrating this point. These results show the disadvantage of using least cost routing as part of a recovery scheme and the direct effect this can have on recovery times.

In the final part of chapter 6 (section 6.6) a simulation study of DAM with Load Balancing is presented. The results that are presented have been obtained using the network scenario described in section 6.3.3 (“25 node 55 link” network supporting 25 MP2P LSPs). The first set of results in this section examines link usage by recording the number of MP2P LSPs occupying a link in a single direction. In total 110 results were recorded corresponding to the 55 bi-directional links in the network. Using DAM-LB the average number of MP2P LSPs occupying one direction of a link is 5.42 and the standard deviation is calculated to be 4.44 MP2P LSPs. The results obtained using DAM in the same network scenario are very similar, the average number of MP2P LSPs occupying one direction of a link is 5.49 and the standard deviation is calculated to be 4.34 MP2P LSPs. In this particular example all link costs in the network were set to one and since all nodes in the network were acting as egress / ingress nodes the resulting MP2P LSPs leave little scope for load balancing to have an affect. The change in link utilisation with DAM-LB is very small for this particular example but not insignificant; the reduction in the average link utilisation is only 1.3% and there is slightly more spread in the results recorded. This is an important result as the average link utilisation did not grow but was reduced slightly by DAM-LB despite there being little scope for load balancing to have an affect. Moreover it is

important that the scheme does not act over zealously when faced with such a network scenario as this may bring adverse affects by re-routing flows that need not have been re-routed. Section 6.6 also presents an investigation into the recovery times that are achieved when using DAM-LB in the same network scenario as in section 6.3.3. Each of the 55 links was failed in turn in a separate simulation and the recovery times were recorded. The average recovery time recorded for DAM-LB in this scenario is 91.45 milliseconds and the standard deviation was calculated to be 31.08 milliseconds again these results are similar to those recorded for DAM. In the time range between 60 and 120 milliseconds 74.5% of recoveries took place; this is only marginally better than the result recorded for DAM.

The analysis and simulation of DAM that has been presented in this thesis show DAM to be effective in the examples that have been discussed. DAM offers a very practical solution with respect to the formation and maintenance of MP2P LSPs; it is believed that DAM is the first complete protocol to have been designed and implemented for this purpose and to-date the only one to be in existence. In terms of practical solutions, the only other standard to be published is the work in [YAS06]. This IETF draft [YAS06] specifies extensions to RSVP-TE for the support of MP2P LSPs. However, these extensions are limited to signalling that facilitates the merging of P2P LSPs. The selection of candidate P2P LSPs and appropriate merge points is not discussed. [SAI00] [BHA02] [BHA03] and [APP03] are schemes that have been discussed in section 2.7; these are all centralised in their calculation of MP2P LSPs. Moreover these calculations are conducted offline and can be slow in arriving at a solution. In the case of [APP03] this calculation was conducted using a commercial linear programming software package CPLEX [ILOG1]. DAM is a decentralised online protocol for the formation and maintenance of MP2P LSPs. The focus of the schemes in [SAI00] [BHA02] [BHA03] and [APP03] has been label space reduction. In the examples shown in section 6.4 DAM has been successful in reducing the label space and has outperformed the scheme by Bhatnagar *et al* [BHA03] in the Bhatnagar network.

One of the major challenges associated with the maintenance of LSPs is the issue of MPLS recovery. In this thesis the merits of the various approaches employed in MPLS recovery have been discussed in section 2.3. Re-routing at present remains an unattractive solution as most schemes rely on the re-convergence of high layer routing protocols leading to relatively slow recovery times. The most attractive solutions for MPLS recovery are based on some form of protection switching; this is because alternatives tend to be slow in comparison. Whilst protection switching is able to offer fast recovery it can be expensive in terms of the resources

required to support it. In addition to this there can be a great variation in recovery times as they can be impaired by failure notification having to travel far to reach the point of repair [HUA02]. DAM addresses all of these issues through its localised structured recovery scheme. DAM provides fast re-routing by supporting a DAG and removing the need to rely on the re-convergence of slow higher layer protocols before restoration can take place. DAM takes advantage of an algorithm originally designed for use in adhoc networks to ensure the fast maintenance of the DAG in response to failure. The structured approach of DAM in providing fast re-routing enables DAM to limit the amount of signalling required to perform a restoration. The amount of signalling used by DAM to perform restoration is unaffected by the scale of the network, the same cannot be said of protection switching (see section 2.3). This consistency of DAM in the delivery of restoration can be seen in the simulation and analysis in sections 6.1 to 6.3.3. This is an important feature as it allows service providers to specify tighter bounds of recovery times when offering services to customers. An alternative restoration strategy is implemented in DAM-PP. This strategy also alleviates the need for the reliance on higher layer routing protocols to re-converge prior to re-routing taking place. DAM-PP uses pre-planned alternative downstream nodes selected on the basis of a shortest path first calculation. This, however, can lead to restoration of multiple LSPs being focused on a particular neighbouring node and as a result can adversely affect restoration times. This consideration is also applicable to the approach adopted by [AHN02] where a series of shortest path first calculations are conducted in order to perform re-routing. In DAM this is not an issue as re-routing is performed through the use of the DAG that is not based on a shortest path first calculation. A consequence of using the DAG for this purpose is that whilst re-routing does not always take place via the most optimal neighbouring node, in the restoration of multiple MP2P LSPs re-routing can be spread more widely amongst all viable neighbours (see sections 6.3.2 and 6.5 for examples of this) leading to faster restoration times.

From a practical perspective DAM is simple to implement and it is believed that existing MPLS hardware could have a firmware or software update so that DAM could be put into service in existing networks. The operational complexity of the DAM is scoped by its most complex task this is the reverse Dijkstra calculation. Dijktra's shortest path first calculation is widely used and known to be practical for even large networks, its complexity is quoted to be $O(n^2)$ [JOH73][BAR98] for the original algorithm where n is the number of nodes. This is the same as for the adapted version used by DAM as replacing the standard cost with the reverse cost has no bearing on calculation complexity. From the perspective of the service provider whilst using DAM has benefits it may be perceived as restrictive in terms of the MP2P tree formations. A

service provider will almost certainly wish to implement P2P LSPs alongside those formed using DAM and it is probable that the label space may be partitioned to meet this need for both types of LSP. The decision as to whether or not to employ DAM may also be influenced by the service providers network topology. As the size of the network becomes smaller likewise the benefits of using DAM may become less. The reduction in label space using MP2P LSPs over P2P LSPs is decreased in smaller networks. Another consideration is that of node connectivity, a poorly connected node is more likely to require DAM to perform more signalling during a restoration since it is more likely to require link reversal. This may be more prevalent in smaller networks. These factors may well mean that a service provider considers DAM to be impractical for its particular needs.

One of the problems faced by DAM regardless of where it is deployed is that many failures over a given period may lead to sub-optimal MP2P LSPs as a consequence of the restoration strategy employed. To counter this, MP2P LSPs are periodically replaced with new MP2P LSPs. This periodic replacement is carried out by DAM in a manner that avoids network disruption. DAM does this by supporting simultaneously both the old tree and the replacement tree for a short period of time after the replacement tree has been formed. Details of this are given in section 3.3.

During the course of this work it was observed that the strategy employed by DAM for the purposes of restoration could be adapted for load balancing, the acyclic nature of the DAG lending itself to achieving this aim. It is believed that to date no scheme has been published that uses the multipath acyclic nature of the DAG to perform load balancing. DAM-LB was designed and implemented as a result, DAM-LB is believed to be the first of its kind in that it has been designed to provide load balancing for multiple overlaid MP2P LSPs. In DAM-LB the decision to reroute part of an MP2P LSP in order to redistribute flows is made locally and the load balancing is carried out in a distributed fashion. Nodes participating in DAM-LB do not have global knowledge of which or how many MP2P LSPs are flowing along a link. Though it would be possible for nodes to make calculations locally based on the link state database this would be impractical and require large amounts of processing. Moreover any calculation based on the link state database made after the MP2P LSPs have been formed could well be invalid as a result of the failure. Restoration of an LSP using DAM may not yet be known to the higher layer protocol populating the link state database. In order to address this issue DAM-LB adopts a method that allows neighbouring nodes to share information regarding their ability to support extra flows on a particular link. Since this scheme deals with multiple overlaid MP2P LSPs, this

shared information is relative to a particular MP2P LSP, link details are therefore masked in this process. DAM-LB uses what is called a “downstream utilisation index”. This is an integer relating to how many other MP2P LSPs share a particular MP2P LSP’s current downstream link. Knowledge of this “downstream utilisation index” allows a node to make a more informed decision as to whether or not it should agree to be used to re-route part of a MP2P LSP. The directed label requests used by DAM-LB for the re-distribution of flows to improve link utilisation are sent to specific nodes that are chosen because of their position in the DAG.

7.1 Future Work

Further development of DAM could include additional extensions to standard DAM in order to support a multiple CoS trees for a particular egress node. The standard implementation of DAM already has the ability to support multiple trees. In order to extend this to multiple CoS trees only the algorithm that calculates the trees, based on any required constraints, needs to be added.

Further experimental work is also desirable to test the current implementations of DAM under a multitude of different conditions. Simulations investigating both the resilience and scalability of DAM would provide more comprehensive results, particularly in more varied network topologies and with different levels of node participation within specific trees.

At present, a unique exchange of messages is used to recover each distinct MP2P LSP. Since recovery time is related to the maximum number of MP2P LSPs being recovered via a single neighbour it may be possible to reduce recovery times further by using a single composite message exchange to recover all MP2P LSPs being restored via a single neighbour.

An investigation into how evenly distributed the directed flows (arcs) are across all network links for multiple overlaid DAGs would be beneficial possibly leading to the development of new methods for DAG creation.

At present the DAG is formed through a process in which the calculation of heights is made on what height information has been received at that time and, as a last resort, on an IP address comparison (see section 3.2). As part of this ongoing research, work could examine if this process can be improved so that the DAGs that are formed aid in making the load balancing more globally optimal without losing the benefits that the DAG brings to restoration. Of

particular interest here is an area of graph theory that is concerned with edge-disjoint and vertex-disjoint trees. Such trees in the context of networks are trees that share the same ingress points to an egress point but do so without using any of the same intermediate links (edge – disjoint) or nodes (vertex – disjoint). Algorithms of this type have been proposed in [MED99] for pre-planed recovery in networks whereby back up paths follow a completely different path to the path that they are acting as a back up for. These back up paths are pre-calculated. In the context of DAM the formation of the DAG could perhaps benefit by producing DAGs that are partially arc disjoint (edge disjoint in a single direction). At present heights in the DAG are represented by the triple $\{\alpha, \beta, z\}$. This would need to be extended possibly to a 5-tuple so that the calculation of the standard triple could be conducted simultaneously to the calculation to achieve partially disjoint arcs. The reason for investigating this possibility is to consider whether a more globally balanced set of arcs can be selected for the DAGs formed in the network. The benefits that may come from doing this would be a better load balancing mechanism and possibly faster restoration, since it is hoped that this would produce more evenly spread restoration via the maximum number of viable neighbouring nodes. A consideration for this future work is that although DAM and DAM-LB are fast in terms of processing, as all calculations made are not very complicated, the benefits of alternatives would have to be offset against any extra calculation complexity.

Chapter 8

8 Conclusions

This thesis has presented a novel scheme for the formation and maintenance of MP2P LSPs. This scheme and its protocol is called DAM. The design goal that has driven the design of DAM is that MP2P LSPs are created online and maintained dynamically whist providing a dedicated resilience scheme that can lead to fast localised healing in a structured fashion.

MP2P LSPs present an important traffic engineering opportunity. This is particularly relevant in the realm of "MPLS VPNs" where a service provider is likely to benefit from a more manageable core network along with opportunities for the provision of multiple service levels. It is believed that DAM is the first complete protocol to have been designed and implemented for the purpose of MP2P LSP creation and maintenance; it has been shown that DAM is effective in doing this.

DAM has been shown to be successful in reducing the amount of label space and in its structured approach to restoration. DAM provides fast rerouting by supporting a DAG and removing the need to rely on the re-convergence of slow higher layer protocols before restoration can take place. The structured approach of DAM in providing fast rerouting enables DAM to limit the amount of signalling required to perform a restoration. The amount of signalling used by DAM to perform restoration is unaffected by the scale of the network, the same cannot be said of protection switching (see section 2.3). A consequence of using the DAG for restoration is that whilst rerouting does not always take place via the most optimal neighbouring node, in the restoration of multiple MP2P LSPs rerouting can be spread more widely amongst all viable neighbours leading to faster restoration times.

An alternative restoration strategy is implemented in DAM-PP. This strategy also alleviates the need for the reliance on higher layer routing protocols to re-converge in order for rerouting to take place. The performance of DAM-PP has been studied in comparison with DAM.

DAM-LB has been presented it is believed to be the first of its kind in that it has been designed to provide load balancing for multiple overlaid MP2P LSPs. DAM-LB presents a novel method

for performing load balancing, it is believed that to date no other scheme has been published that uses the multipath acyclic nature of the DAG to perform load balancing.

9 References

[AHN02] Gaeil Ahn, Jongsoo Jang and Woojik Chun, “An Efficient Rerouting Scheme for MPLS-Based Recovery and Its Performance Evaluation”, Telecommunication Systems 19:3,4, pages 481–495, Kluwer Academic Publishers 2002

[AMY01] R. M. Krishnaswamy and K. N. Sivarajan, “Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers”, IEEE/ACM Transactions on Networking, vol. 9, no. 2, April 2001

[AND04] Loa Andersson and Eric C. Rosen (Eds.), “Framework for Layer 2 Virtual Private Networks (L2VPNs)”, Network Working Group Internet Draft, IETF June 2004, draft-ietf-l2vpn-l2-framework-05.txt

[APP03] David Applegate and Mikkel Thorup, “Load optimal MPLS routing with N + M labels”, IEEE INFOCOM 2003

[BAR98] Michael Barbehenn, “A Note on the Complexity of Dijkstra’s Algorithm for Graphs with Weighted Vertices”, IEEE Transactions On Computers, Vol. 47, No. 2, February 1998

[BHA02] Sudeept Bhatnagar, Samrat Ganguly and Badri Nath, “Label Space Reduction in Multipoint-to-Point LSPs for Traffic Engineering”, 2nd European Conference on Universal Multiservice Networks, ECUMN2002

[BHA03] Sudeept Bhatnagar, Samrat Ganguly and Badri Nath, “Creating Multipoint-To-Point LSPs For Traffic Engineering”, Workshop on High Performance Switching and Routing 2003, HPSR, June 24-27, 2003

[BHA05] Sudeept Bhatnagar, Samrat Ganguly and Badri Nath, “Creating Multipoint-To-Point LSPs For Traffic Engineering”, IEEE Communications Magazine, January 2005

[BOA02] Bruce Boardman, “MPLS VPNs: The Real Deal”, Network Computing 6/10/2002, www.networkcomputing.com

[BRI00] Paul Brittain, Adrian Farrel, “MPLS VIRTUAL PRIVATE NETWORKS - A review of the implementation options for MPLS VPNs including the ongoing standardization work in the IETF MPLS Working Group”, Data Connection Limited, November 2000, http://www.dataconnnection.com

[CAL97] Jean Calvignac, Patrick Droz, Claude Basso and Doug Dykeman, “Dynamic IDentifier Assignment (DIDA) for Merged ATM Connections”, The ATM Forum Technical Committee, July 20-25, 1997 (Montreal, Canada), http://www.zurich.ibm.com/~dro/atm-97-0504.txt

[CIS04] Cisco Systems, Inc. “The Move To MPLS-Based VPNs: Exploring Service Options”, White Paper Cisco Systems, Inc. 2004. http://www.cisco.com/en/US/netsol/ns341/ns121/ns193/networking_solutions_white_papers_list.html

[CIS04a] Cisco Systems, Inc. “MPLS-BASED VPNS: WHAT’S POSSIBLE FOR ENTERPRISES”, White Paper Cisco Systems, Inc. 2004. http://www.cisco.com/application/pdf/en/us/guest/netsol/ns193/c654/cdccont_0900aecd800f911c.pdf

[COM00] Douglas E. Comer, “Internetworking with TCP/IP - Volume 1”, 4th Edition, Prentice Hall 2000

[COR01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, “Introduction to Algorithms”, 2nd Edition, MIT Press and McGraw-Hill, 2001, Section 24.1: The Bellman-Ford algorithm, pages 588–592

[CUR04] Curt Harler, “Network Wars: MPLS vs. VPN”, Processor Editorial Article, October 22, 2004, Vol.26 Issue 43, http://www.processor.com/editorial/article.asp?article=articles/P2643/32p43/32p43.asp&guid=

[DOY98] Jeff Doyle, “Routing TCP/IP”, Volume1, Cisco Press 1998

[DIJ59] E.W. Dijkstra, “A Note on Two Problems in Connexion with Graphs”, Numerische Mathematik, vol. 1, pp. 269-271, 1959

[FEN06] Bill Fenner, Mark Handley, Hugh Holbrook and Isidor Kouvelas, “Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised) ”, IETF Internet Draft, draft-ietf-pim-sm-v2-new-12.txt, IETF March 2006

[FER98] P. Ferguson, G. Huston, “What is a VPN?” Revision 1, April 1998, http://www.employees.org/~ferguson/vpn.pdf

[FIN04] Matthew Finlayson, Jon Harrison and Richard Sugarman, “VPN TECHNOLOGIES - A COMPARISON”, Data Connection Limited, 2004, http://www.dataconnnection.com

[FOR03] Behrouz A. Forouzan, “TCP/IP Protocol Suite”, Second Edition, McGraw Hill 2003

[GAF81] E. Gafni and D. Bertsekas, “Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology”, IEEE Personal Communications January 1981

[GAR79] Michael R. Garey, David S. Johnson, “Computers and intractability: a guide to the theory of NP-completeness”, San Francisco, W. H. Freeman, 1979.

[GON01] S. Gonzlez-Valenzuela, V.C.M. Leung and S. T. Vuong, “Multipoint-to-Point Routing With QoS Guarantees Using Mobile Agents”, Mobile Agents for Telecommunication Applications: Third International Workshop, MATA 2001, Montreal, Canada, August 14-16, 2001. Proceedings, LNCS 2164 - Springer-Verlag Berlin Heidelberg 2001

[GON02] S. Gonzlez-Valenzuela and V.C.M. Leung, “QoS Routing for MPLS Networks Employing Mobile Agents”, IEEE Network, Volume: 16 Issue: 3, May/Jun 2002, pages 16 - 21

[HAR03] Jon Harrison, “VPN TECHNOLOGIES - A COMPARISON”, Data Connection Limited, February 2003, http://www.dataconnnection.com

[HAR04] Ed Harrison, Adrian Farrel and Ben Miller, “PROTECTION AND RESTORATION IN MPLS NETWORKS - An examination of the methods for protecting MPLS LSPs against failures of network resources - Version 2”, Data Connection Limited, 2004, http://www.dataconnnection.com

[HUA02] Changcheng Huang, Vishal Sharma, Ken Owens, and Srinivas Makam, “Building Reliable MPLS Networks Using a Path Protection Mechanism”, IEEE Communications Magazine, March 2002

[HUN02] Lemma Hundessa and Jordi Domingo-Pascual, “Reliable and Fast Rerouting Mechanism for a Protected Label Switched Path”, IEEE Global Telecommunications Conference, GLOBECOM November 2002

[IBM01] Adolfo Rodriguez, John Gatrell, John Karas, Roland Peschke, “TCP/IP Tutorial and Technical Overview” IBM REDBOOKS, August 2001

[ILOG1] ILOG CPLEX, http://www.ilog.com/products/cplex/

[JOH73] Donald B . Johnson, “A Note on Dijkstra's Shortest Path Algorithm”, Journal of the Association for Computing Machinery, Vol. 20, No. 3, July 1973, pages 385-388

[KAR98] T. Karygiannis, “Network Security Testing Using Mobile Agents”, The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology, London, March 1998, http://csrc.nist.gov/mobilesecurity/Publications/Agents_PAAM98.pdf

[KNI04] Paul Knight and Chris Lewis, “Layer 2 and 3 Virtual Private Networks: Taxonomy, Technology, and Standardization Efforts”, IEEE Communications Magazine, June 2004

[KOD01] Murali Kodialam and T. V. Lakshman, “Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information”, IEEE INFOCOM 2001

[KOM03] K. Kompella, M. Leelanivas, Q. Vohra, J. Achirica, R. Bonica, D. Cooper, C. Liljenstolpe, E. Metz, H. Ould-Brahim, C. Sargor, H. Shah,V. Srinivasan and Z. Zhang, IETF Internet Draft – “Layer 2 VPNs Over Tunnels”, draft-kompella-ppvpn-l2vpn-03.txt, IETF April 2003

[LAK06] Umesh Lakshman and Lancy Lobo, “MPLS Configuration on Cisco IOS Software”, Cisco Press 2006, Chapter 9: MPLS Traffic Engineering, pages 375 - 417

[MAR03] Jose L Marzo, Eusebi Calle, Caterina Scoglio and Tricha Anjali “Adding QoS Protection in Order to Enhance MPLS QoS Routing”, IEEE International Conference on Communications, ICC May 2003

[MAR03a] Jose L Marzo, Eusebi Calle, Caterina Scoglio and Tricha Anjali “QoS Online Routing and MPLS Multilevel Protection: A Survey”, IEEE Communications Magazine, October 2003

[MAR05] Cyril Margaria, Guillaume Juillot, and Achim Autenrieth, “Performance Evaluation of a GMPLS Prototype”, IV Workshop in G/MPLS Networks, Girona, April 2005

[MED99] M. Medard, S.G. Finn, R.A. Barry, R.G. Gallager, “Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graph”, IEEE/ACM Transactions on Networking, Vol. 7, No. 5, pp. 641-652, October 1999

[MUR95] K. Murakami and H. Kim, “Joint optmisation of capacity and flow assignment for self-healing ATM networks”, in Proc. IEEE ICC'95, pp. 216-220

[NEO04] Constantinos Neophytou and Chris Phillips, “A Scheme for the Dynamic Formation of Robust Multipoint to Point LSPs” IEEE CCNC 2004, IEEE Consumer Communications and Networking Conference, Las Vegas January 2004

[PAR97] Vincent D. Park and M. Scott Corson, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks”, Proceedings of INFOCOM 97, IEEE 1997

[PAR01] V. Park and S. Corson, “Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification”, MANET Working Group Internet Draft, IETF July 2001, draft-ietf-manet-tora-spec-04.txt

[PER00] Charles E.Perkins, “Ad Hoc Networking”, Addison-Wesley December 2000, Chapter 8 - Link Reversal Routing

[QIA03] Dahai Xu, Yizhi Xiong and Chunming Qiao, “Novel Algorithms for Shared Segment Protection”, IEEE Journal On Selected Areas In Communications, Vol. 21, No. 8, October 2003

[RAH91] Moe Rahnema, “Frame Relaying and the Fast Packet Switching-Concepts and Issues”, IEEE Network Magazine, July 1991

[RFC793] J. Postel, RFC 793 “Transmission Control Protocol”, IETF, September 1981

[RFC1075] D. Waitzman, C. Partridge and S. Deering RFC 1075 “Distance Vector Multicast Routing Protocol”, IETF November 1988

[RFC1584] J. Moy, RFC 1584 “Multicast Extensions to OSPF ”, IETF March 1994

[RFC1771] Y. Rekhter, T. Li, RFC 1771 “BGP-4”, IETF March 1995

[RFC1812] F. Baker, RFC 1812 “Requirements for IPv4 Routers”, IETF June 1995

[RFC2328] J. Moy, RFC 2328 “OSPF Version 2”, IETF April 1998

[RFC2362] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley,V. Jacobson, C. Liu, P. Sharma, L. Wei, RFC 2362 “Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification”, IETF June 1998

[RFC2453] G. Malkin, RFC 2453 “RIP Version 2”, IETF November 1998

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, RFC 2475 “An Architecture for Differentiated Services”, IETF December 1998

[RFC3031] E. Rosen, A. Viswanathan and R. Callon, RFC 3031 “Multiprotocol Label Switching Architecture”, IETF January 2001

[RFC3036] L. Andersson, P. Doolan, N. Feldman, A. Fredette and B. Thomas, RFC 3036 “LDP Specification”, IETF January 2001

[RFC3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, RFC 3209 “RSVP-TE: Extensions to RSVP for LSP Tunnels”, IETF December 2001

[RFC3212] B. Jamoussi, Editor, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty and A. Malis, RFC 3212 “Constraint-Based LSP Setup using LDP”, IETF January 2002

[RFC3213] J. Ash, M. Girish, E. Gray, B. Jamoussi and G. Wright, RFC 3213 “Applicability Statement for CR-LDP”, IETF January 2002

[RFC3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, RFC 3272 “Overview and Principles of Internet Traffic Engineering”, IETF May 2002

[RFC3353] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul and F. Ansari, RFC 3353 “Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment”, IETF August 2002

[RFC3469] V. Sharma and F. Hellstrand, Editors, RFC 3469 “Framework for Multi-Protocol Label Switching (MPLS)-based Recovery”, IETF February 2003

[RFC3973] A. Adams, J. Nicholas and W. Siadak, RFC 3973 “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)”, IETF January 2005

[RFC4090] P. Pan, G. Swallow and A. Atlas, Editors, RFC 4090 “Fast Reroute Extensions to RSVP-TE for LSP Tunnels”, IETF May 2005

[RFC4271] Y. Rekhter, T. Li, S. Hares, RFC 4271 “A Border Gateway Protocol 4 (BGP-4)”, IETF January 2006

[RFC4420] A. Farrel, D. Papadimitriou, J.-P. Vasseur and A. Ayyangar, RFC 4420 “Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) Establishment Using Resource ReserVation Protocol-Traffic Engineering (RSVP-TE)”, IETF February 2006

[RFC4364] Eric C. Rosen and Yakov Rekhter, RFC 4364 “BGP/MPLS IP Virtual Private Networks (VPNs)”, IETF February 2006

[RFC4447] L. Martini, Ed., E. Rosen, N. El-Aawar, T. Smith and G. Heron, RFC 4447 “Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)”, IETF April 2006

[RFC4577] E. Rosen, P. Psenak and P. Pillay-Esnault, RFC 4577 “OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)”, IETF June 2006

[SAI00] Hiroyuki Saito, Yasuhiro Miyao and Makiko Yoshida, “Traffic Engineering using Multiple Multipoint-to-Point LSPs”, IEEE INFOCOM 2000

[SEM01] Chuck Semeria, “RFC 2547bis: BGP/MPLS VPN Fundamentals” Juniper Networks, Inc., 2001, www.juniper.net

[SIE02] Gerard Sierksma, “Linear and Integer Programming - Theory and Practice”, Second Edition, Publisher: Marcel Dekker Inc. 2002

[SPR04] John Polivka, “Sprint extends MPLS VPN benefits globally - Network-based unified services available in over 100 countries”, Sprint 27th October 2004, http://www.sprintworldwide.com/english/about/success/mplsvpn.pdf

[XIO99] Y. Xiong and L. G. Mason, “Restoration strategies and spare capacity requirements in self-healing ATM networks”, IEEE/ACM Transactions on Networking, vol. 7, no. 1, February 1999, pp. 98-110

[YAS06] Seisho Yasukawa, “Supporting Multipoint-to-Point Label Switched Paths in Multiprotocol Label Switching Traffic Engineering”, IETF Internet Draft, draft-yasukawa-mpls-mp2p-rsvpte-00.txt, IETF February 2006

10 Publications

[1] Constantinos Neophytou and Chris Phillips, “Self-Healing Multipoint-to-Point Label Switched Paths”, IEEE International Conference on Information and Communication Technologies: from Theory to Applications, Damascus, April 2006, (Invited Paper)

[2] Constantinos Neophytou and Chris Phillips, “Robust Multipoint to Point LSPs using the Partial Reversal Method”, IV Workshop in G/MPLS Networks, Girona, April 2005

[3] Chris Phillips, Song Dong, Liang Gao, Constantinos Neophytou , “MPLS: Challenges and Opportunities” , MPLS Networks Workshop Universitat de Girona, 2004

[4] Constantinos Neophytou and Chris Phillips, “A Scheme for the Dynamic Formation of Robust Multipoint to Point LSPs” IEEE CCNC 2004, IEEE Consumer Communications and Networking Conference, Las Vegas January 2004, (Awarded IEEE COMSOC Student Travel Grant)

[5] Constantinos Neophytou and Chris Phillips, “Dynamically Adaptive Mulitipoint to Point LSPs”, Proceedings of the London Communications Symposium 2003

[6] Juan Jim Tan, Leonid Titkov, Constantinos Neophytou, “Securing Multi-Agent Platform Communication”, Autonomous Agent Multi Agent Systems (AAMAS) 2002, Workshop on Security of Mobile Multi Agent Systems (SEMAS), July 2002, Bologna, Italy

Appendix 1

The DAM-PP protocol specification was then decomposed into the following state transition table before being implemented as a process model within the OPNET modeller.

State	Event	Condition	Action	Final State
INIT	Begin simulation	Always	Carry out initial registrations Initialise relevant data structures Set interrupt for tree calculation if node is egress	IDLE
IDLE	Calculate tree interrupt	Timer expired	Reset timer	CalcTree
IDLE	Receive HeartBeat	Always		RecvHBeat
IDLE	Receive Qprop	Always		QueryProp
IDLE	Maintenance interrupt	MNT timer expired -180s	Mark tree status	DamJoin
IDLE	QProp interrupt	Timer expired and HB not received		DamJoin
IDLE	HeartBeat interrupt	Timer expired and is egress node	Resend HB to refresh tree	IDLE
IDLE	Tree Dead timeout	Expired timer (10 minutes)	Set Status in Tree database	IDLE
CalcTree	New Tree	Always	Send initial HB	IDLE
CalcTree	Replacement Tree	Tree timeout	Send initial HB, Set Purge timeout on old tree	IDLE
RecvHBeat	Initial Heartbeat message	Always	Replace Label, Send on revised HB to upstream neighbours, Schedule interrupt for PrPlan state to	IDLE

			select alternative downstream neighbour	
RecvHBeat	Normal Heartbeat	Always	Reset timers and send on revised heartbeat	IDLE
RecvHBeat	Heartbeat in reply to a QProp message	Always	Update tree DB and remove source of this HB from upstream neighbour list if necessary	IDLE
QueryProp	Node part of tree	Always	Send label binding to source of QP and add this node to list of upstream neighbours	IDLE
QueryProp	Node not part of tree	Always	Store details of QP message	DamJoin
DamJoin	MNT / QP process	Tree status is MNT	Get pre planned alternative downstream neighbour, Send QueryProp message and set QP interrupt for some future time	IDLE
DamJoin	MNT / QP process	Tree entry does not exist	Set QP interrupt for some future time – 10s	IDLE
PrPlan	Calculate alternative downstream neighbour	Always	Calculate cost offset for tree, perform reverse Dijkstra using inflated cost	IDLE

Appendix 2

The DAM-LB protocol specification was then decomposed into the following state transition table before being implemented as a process model within the OPNET modeller.

State	Event	Condition	Action	Final State
INIT	Begin simulation	Always	Carry out initial registrations Initialise relevant data structures Set interrupt for tree calculation if node is egress	IDLE
IDLE	Calculate tree interrupt	Timer expired	Reset timer	CalcTree
IDLE	Receive HeartBeat	Always		RecvHBeat
IDLE	Receive Qprop	Always		QueryProp
IDLE	Maintenance interrupt	MNT timer expired -180s	Mark tree status	DamJoin
IDLE	QProp interrupt	Timer expired and HB not received		DamJoin
IDLE	HeartBeat interrupt	Timer expired and is egress node	Resend HB to refresh tree	IDLE
IDLE	Tree Dead timeout	Expired timer (10 minutes)	Set Status in Tree database	IDLE
CalcTree	New Tree	Always	Send initial HB, Invoke DAM Height protocol	IDLE
CalcTree	Replacement Tree	Tree timeout	Send initial HB, Invoke DAM Height protocol, Set Purge timeout on old tree	IDLE

RecvHBeat	Initial Heartbeat message (HB type 1)	Always	Replace Label, Send on revised HB to upstream neighbours, Set "Send_LBQP" interrupt if load balancing is required	IDLE
RecvHBeat	Normal Heartbeat (HB type 0)	Always	Only respond to HB from active downstream neighbour - Reset timers and send on revised heartbeat	IDLE
RecvHBeat	Heartbeat in reply to a QProp message for healing or joining a tree (HB type 2)	Always	Update tree DB (reset downstream neighbours and active neighbour flag) and remove source of this HB from upstream neighbour list if necessary	IDLE
RecvHBeat	Heartbeat in reply to a QProp message for load balancing request (HB type 3)	Always	Update tree DB (set alternative downstream neighbour and active neighbour flag = 0)	IDLE
QueryProp	QP message type 0 for tree healing / joining Node part of tree	Always	Send label binding to source of QP and add this node to list of upstream neighbours	IDLE
QueryProp	QP message type 0 for tree healing / joining Node not part of tree	Always	Store details of QP message	DamJoin
QueryProp	QP message type 1 for Load balancing, Node can accept additional traffic	Always	Send label binding (HB type 3) to source of QP and add this node to list of upstream neighbours	IDLE

DamJoin	MNT / QP process	DAG exists and Tree status not MNT	Get relevant heights from DAM Height process, Send QueryProp message and set QP interrupt for some future time	IDLE
DamJoin	MNT / QP process	DAG does not exist	Set remote interrupt to DAM Height process in order to Query Height, set QP interrupt for some future time – 10s	IDLE
LB_QP	More than one downstream neighbour exists in the DAG	Always	Send QP type 1 to alternative downstream neighbour	IDLE

Appendix 3

Implementation & Validation

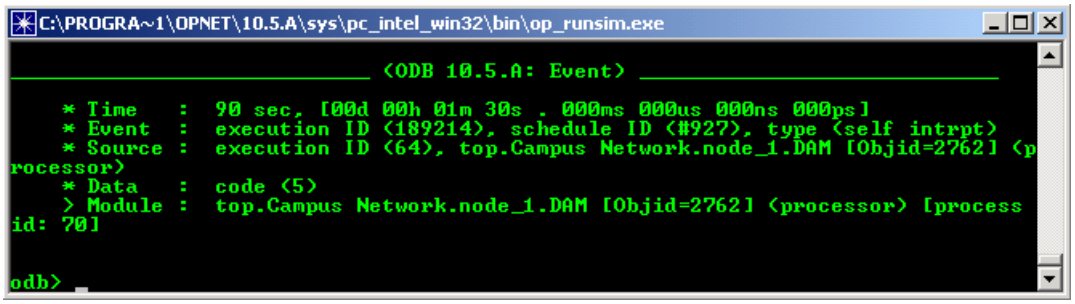
The implementation of the DAM simulator was undertaken in OPNET, all versions of DAM were implemented in a full protocol stack as shown in the architecture diagram shown in figure 16. The implementations of OSPF, IP and ARP that were used in the simulations are standard models provided by OPNET, the correct functionality of these models was confirmed before work began on the implementation of DAM process models. Moreover the standard models conform to all the relevant IETF RFCs. The various implementations of DAM interact with the RFC compliant standard models.

All DAM simulation models underwent the same method of validation that is described in this appendix. Validation was conducted at three different levels, these are:

- The interrupt / state transition level
- The data structure level
- The packet level

Combined these three forms of testing provided full functional level validation.

Validation at the interrupt /state transition level was necessary since the simulation models were implemented as state models. These state models were derived from the state transition tables that formed the first phase of development from specification to working simulator (examples of state transition tables can be found in appendix 1 & 2). The checks undertaken formed the first phase of validation and involved ensuring the correct transition from one state to the next given a particular interrupt. Validation at the interrupt /state transition level is also necessary to ensure there are no design flaws in the original state transition tables from which the state models are derived, a simulation model could become stuck in a state much like a while loop becoming infinite because an exit condition is not being satisfied. In order to conduct the checks on the simulator model state machines, simulations were stopped during transitions between states to confirm correct interrupt /state transitions were taking place. The following snapshot shows a simulation being stopped.

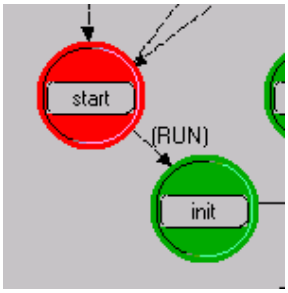


In the above snapshot the event is a self-interrupt and the code for this interrupt is the integer 5

In the header block of the simulation model the following state transition condition is defined:

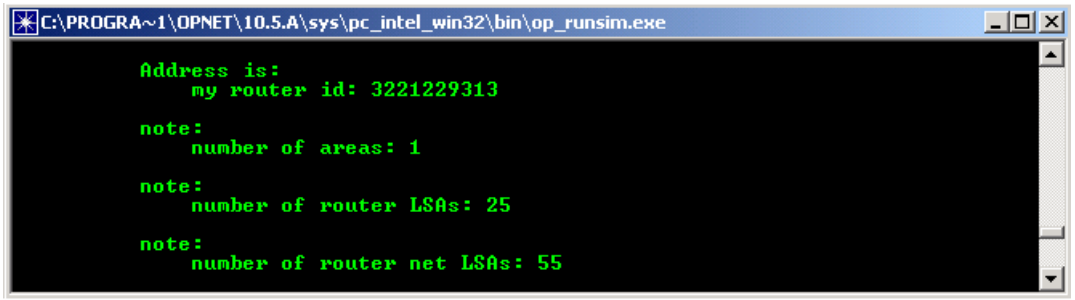
#define RUN (intrpt_type == OPC_INTRPT_SELF && intrpt_code == 5)

This corresponds to the following portion of the state model shown in figure 17

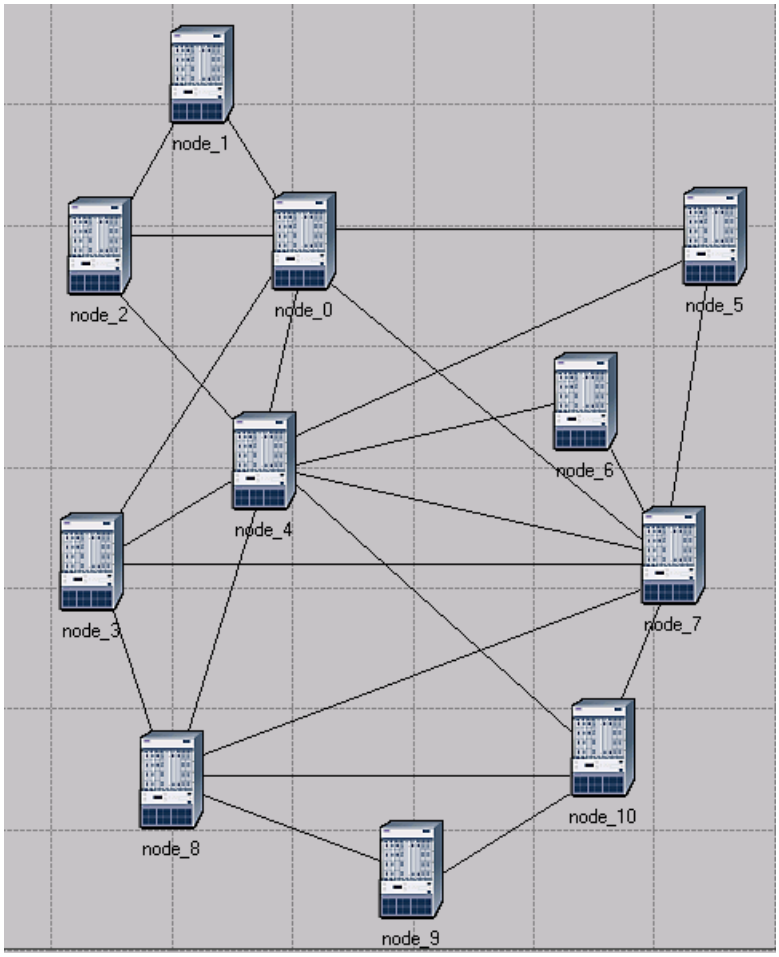


Validation was also necessary at the data structure level. This type of validation is straight forward, in that a break point is inserted in to the simulation model code and the contents of a particular data structure are printed to the screen.

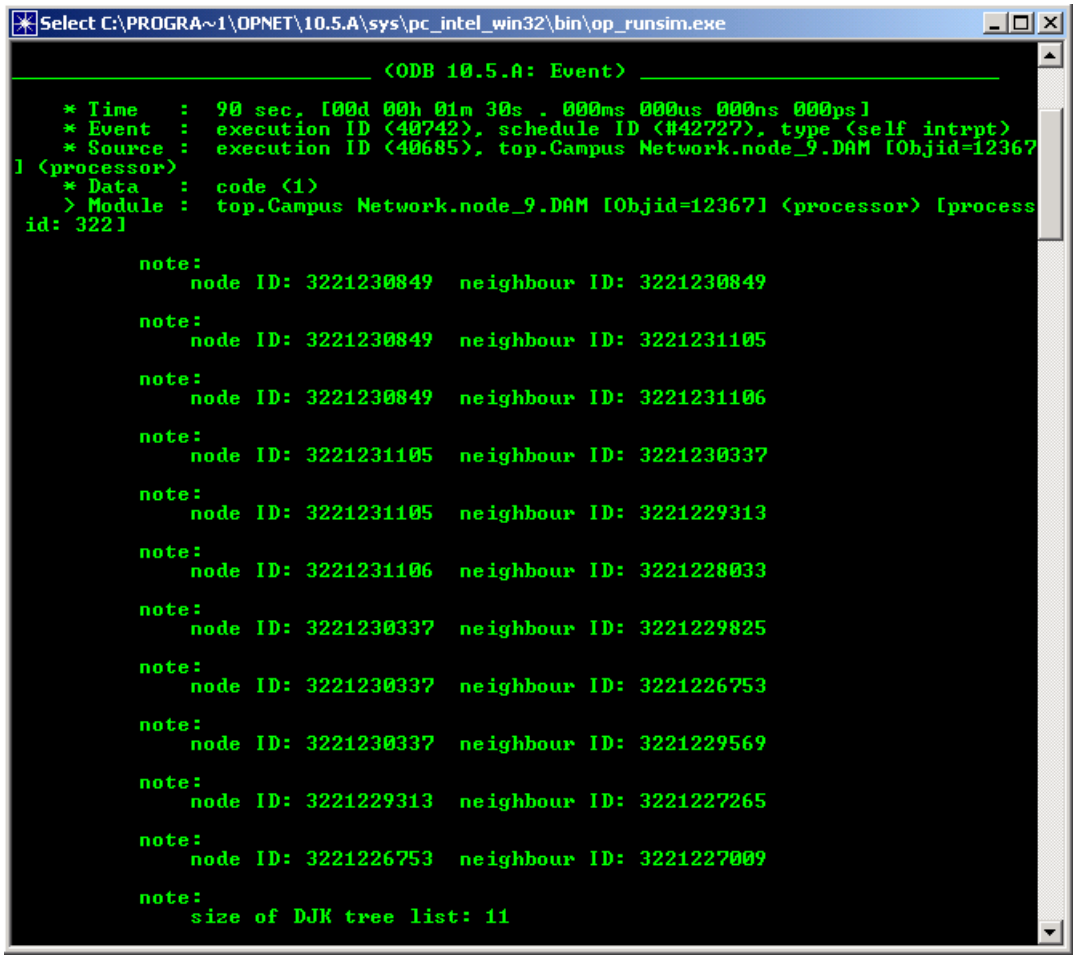
The following snapshot formed part of the validation to ensure that DAM had correct access to the OSPF link state database.



Date structure validation was also used to valid routines such as the reverse Dijkstra calculation. The following snapshot is of the New Jersey LATA Network (as shown in figure 22) implemented in OPNET Modeler:



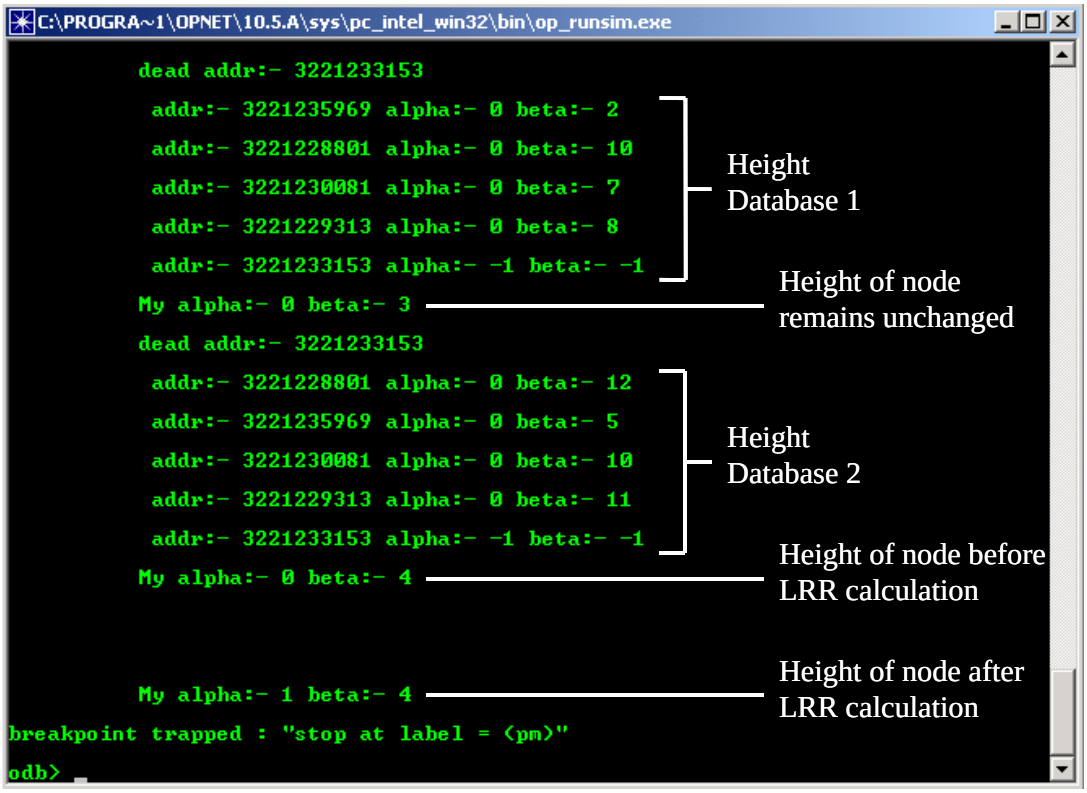
The simulator was stopped at the point when the reverse Dijkstra calculation was completed at node 9 and the following snapshot was taken of the tree database:



The node id 3221230849 is the router ID IP address as a 32 bit unsigned integer that is equal to C0001501 in hexadecimal and the equivalent of the dotted decimal IP address 192.0.21.1, this address is the router ID belonging to node 9 in the New Jersey LATA network shown in the previous snapshot. Replacing the unsigned integer addresses with the actual node number (eg 3221230849 -> 9) gives the following tree database:

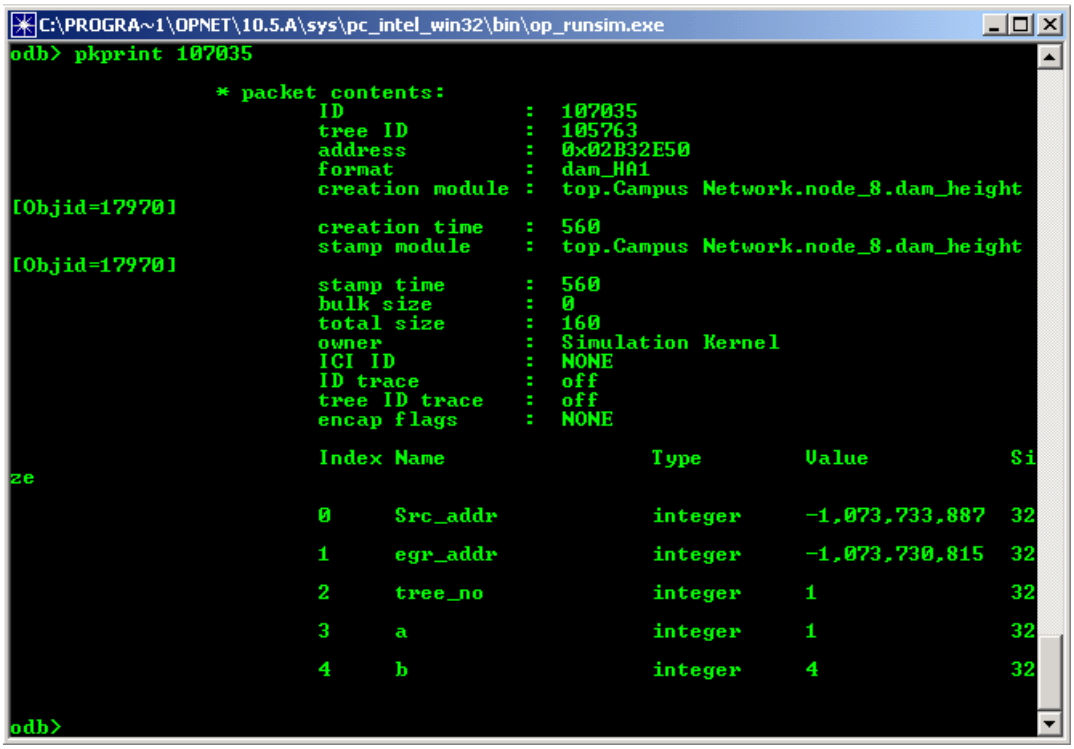
```
odule :  top.Campus Network.node_9.DAM [Objid=12367] (pro
2]

   note:
       node ID: 9  neighbour ID: 9

   note:
       node ID: 9  neighbour ID: 10

   note:
       node ID: 9  neighbour ID: 8

   note:
       node ID: 10  neighbour ID: 7

   note:
       node ID: 10  neighbour ID: 4

   note:
       node ID: 8  neighbour ID: 3

   note:
       node ID: 7  neighbour ID: 6

   note:
       node ID: 7  neighbour ID: 0

   note:
       node ID: 7  neighbour ID: 5

   note:
       node ID: 4  neighbour ID: 2

   note:
       node ID: 0  neighbour ID: 1

   note:
       size of DJK tree list: 11
```

The same method of validation, outputting data structures and the results of calculations was used throughout the validation process. The following snapshot is from the DAM Height protocol and shows a new height calculated after a node loses its last outgoing link



In the above snapshot in both Height Database 1 and 2 the height of the dead neighbour is set to alpha = -1 and beta = -1 signifing that the link to that neighbour has been lost. In Height Database 1 the link to neighbour 3221233153 that is lost is not the last outgoing link at this node so the height (alpha = 0 and beta = 3) remains unchanged. In Height Database 2 the link to neighbour 3221233153 that is lost is the last outgoing link at this node so the LRR calculation must take place, the height before this calculation is, alpha = 0 and beta = 4, and then it is set to, alpha = 1 and beta = 4. Note in the above snapshot {breakpoint trapped : "stop at label = (pm)"} this break point has been inserted into the simulation code to stop the simulation when the simulator is at the end of executing the partial reversal routine in the DAM height process model.

The final form of validation was done at the packet level this involved observing individual packets as they pass between processes in the same node, across links, and finally to the destination node and process. The following snapshot is of a *Height Advertisement* packet that is sent as a result of the link reversal in the previous snapshot:



This kind of validation allows the contents of an individual packet to be printed to the screen, the columns labelled “Index” and “Name” relate to the actual packet fields as they are in the packet, index fields 3 & 4 are the alpha and beta of the height in this *Height Advertisement*, index field 0 is the source address that makes up the third element of the height triple in this message. The index fields 1 & 2 make up the unique tree id for which the DAG relates to.