Aggregation of Traffic Classes in Multi-Protocol Label Switching Networks

by

William Michael Vallat
B.Sc., University of Victoria, 2004

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Aggregation of Traffic Classes in Multi-Protocol Label Switching Networks

by

William Michael Vallat
B.Sc., University of Victoria, 2004

Supervisory Committee

Dr. S. Ganti, Supervisor (Department of Computer Science)

Dr. G.C. Shoja, Departmental Member (Department of Computer Science)

Dr. U. Stege, Departmental Member (Department of Computer Science)

Dr. L. Cai, External Examiner (Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. S. Ganti, Supervisor (Department of Computer Science)

Dr. G.C. Shoja, Departmental Member (Department of Computer Science)

Dr. U. Stege, Departmental Member (Department of Computer Science)

Dr. L. Cai, External Examiner (Department of Electrical and Computer Engineering)

ABSTRACT

As Multi-Protocol Label Switched (MPLS) networks increase in usage and size, the number of traffic engineered tunnels or Label Switched Paths (LSPs) which must be established has an impact on network state maintenance, administration and scalability. The ability to signal and meet Quality-of-Service (QoS) requirements in such networks has been addressed through the addition of Differentiated Services (Diff-Serv) mappings and other traffic engineering mechanisms. However, for the purpose of path computation, route advertisements, signaling and admission control, multiple traffic classes carried together are still treated as a single class. This work explores extensions to MPLS which allow for the accommodation of up to eight distinct traffic classes per label switched path. Through an examination of simulation results, a comparison between existing methods and the proposed additions is made that shows scenarios in which such traffic class aggregation or "bundling" provides a significant reduction in the number of paths which must be maintained in the network.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

(C)SPF (Constrained) Shortest Path First. An algorithm (and its extension) for computing the shortest path from a source to destination using Dijkstra's algorithm.

CT Class Type. A set of traffic on a link which is governed by a set of bandwidth constraints.

DSCP Differentiated Services Code Point. A field in the IP header which indicates the per-hop behaviour to be applied.

EXP Experimental. The 3-bit field in an MPLS label stack entry reserved for future use.

E-LSP EXP-Inferred LSP. A label switched path which differentiates treatment of class types by the value of the EXP field in the MPLS label stack entry.

FEC Forwarding Equivalence Class. A set of packets which may be handled as if they are equivalent for the purpose of forwarding.

LDP Label Distribution Protocol. A method by which label bindings are allocated and distributed to peers in an MPLS network.

LER Label Edge Router. A label switch router located at the edge (ingress/egress) of an MPLS network.

LSP Label Switched Path. The path which will be followed by a labeled packet in an MPLS network.

LSR	Label Switch Router. A node that resides inside the MPLS domain boundary and forwards MPLS packets using only the label.
MPLS	Multi-protocol Label Switching. A technology using label switching in conjunction with network layer routing over various link level technologies.
PHB	Per Hop Behaviour. The forwarding treatment experienced by a packet at each network node in a Differentiated Services domain.
QoS	Quality-of-Service. Applying and ensuring specific, quantifiable performance levels on a network
RDM	Russian Doll Model. A method of partitioning and sharing link bandwidth among multiple classes of traffic.
RSVP	Resource Reservation Protocol. A protocol which supports the reservation of resources across a network.
TE	Traffic Engineering. Finding a path in the network which meets certain constraints imposed by the type of traffic being carried
TED	Traffic Engineering Database. A database containing link state attributes and resources in a network for the purpose of traffic engineering.

Chapter 1

Introduction

1.1 Description

There are two basic architectures for communication networks [30]. The first has been exemplified in the original telephone network, and is called a circuit switched network. Here a connection request is made, for example picking up a telephone and dialing a number, and an end-to-end path is setup within the network between the caller and receiver. An important characteristic of this approach is that for the duration of the call, the resources (such as bandwidth or circuit) which have been allocated to that connection are dedicated. No other connections may use the resources allocated to the call. A positive attribute of this is that the existing connection has been guaranteed resources and will not be interrupted under ordinary circumstances. However, if a connection is not using its allocated resources to total capacity, network resources are being wasted, and consequently fewer calls overall can be sustained by the network.

The second architecture, called the packet switched network, alleviates the resource wastage of circuit switching by giving no single call guaranteed network resources. This approach is used in the Internet, and a common analogy is the postal service. Here a "conversation" or sequence of data to be transmitted is segmented into smaller pieces, with source and destination data tags appended to each segment. These segments are termed "packets". Each packet is sent into the network and multiplexed with all other packets from other communications going on in the network. The route which a packet follows to reach its destination is unknown, as it is handled at each intermediate router, or "hop", by a dynamic shortest path first algorithm. This type of network is termed connectionless. In a connectionless network, resources can be used up to total capacity because every connection is vying for its required resources against all

others. Consequently packets may be discarded at a router if there are insufficient network resources for it, thereby requiring the sender to retransmit. This method allows for greater sharing and network resource utilization, but at the cost of losing any form of guaranteed service. The unpredictable nature of how packets traverse the network is inherently "best effort".

Multi-protocol Label Switching, or MPLS [29], is an emerging technology which can combine the best of both circuit- and packet-switching. When a packet enters an MPLS network, its existing source/destination data tag is consulted and then an additional data tag, called a "label", is added to it based on table entries in the network edge, or ingress router. This label is a unique identifier in the network and exists as part of the packet until it leaves the MPLS domain. It is the only piece of information consulted as the packet moves through the network. Each router in an MPLS network is preconfigured either manually by a network operator, or via a routing algorithm, to determine a fixed path through the network for a given label. For example, if a label value "A" is added to a packet incoming to the edge of an MPLS network, it may follow a predetermined path through a high speed route to its destination. If a different label value "B" is applied at the ingress edge, it may follow an entirely different, perhaps less desirable path with more delay through the network. The importance of applying these labels is that now the network administrators have direct control over and awareness of the path which a certain type of packet will follow through the network. These paths are called label switched paths (LSPs) [29].

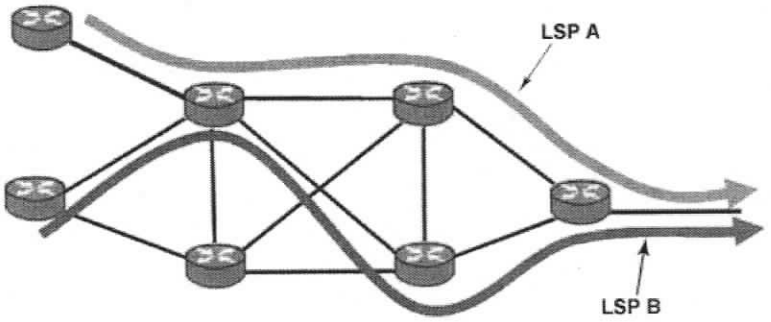


Figure 1.1 : Established label switched paths (LSPs) in an MPLS network

The label switched paths are created via a label distribution protocol (LDP) [2], which sends control messages through the MPLS network to tell routers how to direct packets when they see given labels. The most common is the Resource Reservation Protocol with Traffic Engineering extensions, RSVP-TE [4]. Traffic Engineering (TE) is the term used to describe the aforementioned control of the path through which a data packet travels and the quality of treatment it will receive. Because these paths are now known in advance, the resources from end-to-end can be quantified, and guarantees on availability can be made. This is the positive feature of circuit-switching integrated into MPLS. Additionally, if we do not consider each call, but instead a "class of calls" or class-of-service, we can aggregate these into a single path to maximize network utility. This is the positive feature of packet-switching integrated into MPLS.

There is implemented into RSVP-TE, the capability for signaling routers along a desired path and how to treat a certain class-of-service, that is, a certain label. The path computation itself is done using a Constrained Shortest Path First algorithm (CSPF) [9], because if we wish to route a class-of-service through the network along a given path, this path must satisfy the resource requirements or constraints along each intermediate link from end-to-end. Being able to satisfy the requirements of a class-of-service along all links through the network is termed Quality-of-Service routing [10]. This can be expressed in many ways, such as gold, silver, and bronze service, where users who receive gold service will have their data follow a more desirable path through the network with little delay and high throughput, whereas on the other end of the spectrum bronze users will have their data transmitted along a less expedient path. This differentiation of users, or regions, or fee categories into service classes is the basis of attempts to implement Quality-of-Service in the Internet. It has historically met with limited success, and MPLS is now a significant driving force in this direction.

The mechanisms exist today in MPLS networks to divide traffic into such service classes and route them along paths which satisfy the requirements of each class. For example, a path for high priority interactive video data may be configured, and another path for low quality bulk data transfer may also be configured. Each path will have a

separate associated label entry at each router, and will be separately maintained and configured by network operators. In large MPLS networks, there may be hundreds or thousands of such label switched paths.

1.2 Objectives

Within the MPLS label, there are two fields considered for Quality-of-Service routing. The first is the label itself, which serves as a unique identifier of the route to be followed. This is the common approach in use today [11], necessitating the need for a separate path setup for each class-of-service, even if they may be transmitted on the same physical links. The second field is the experimental or EXP field [29], which is three bits made available for protocol growth. The following research is aimed at using these EXP bits to reduce the number of label switched paths which must be administered in the network. This can be done by using a single label to, rather than identify a single class-of-service, identify an aggregated bundle of multiple classes of service, which can then be differentiated by the value of the EXP bits. With three bits available, there is the possibility of "bundling" up to eight classes of service into a single label identifier. This makes finding a single path through the network which satisfies the requirements of all of the bundled/aggregated classes of service more difficult, however if such a path exists then the number of label switched paths which must be maintained in the network is reduced, which could be a significant gain in terms of scalability and administration.

There are three main issues such a bundling of classes of service requires us to consider. The first is that if this process is to be automated, which would be highly desirable, an algorithm is needed to compute a feasible path satisfying all of the requirements of each class-of-service cumulatively. This restricts the probability of finding such a path, particularly if we are bundling many high-demand classes of service. Therefore, how traffic is bundled together is as important as finding a path. If six classes of service are aggregated and receive differentiated treatment via the EXP bits, there may not be a single path which satisfies their cumulative requirements. Therefore it must be separated, but whether to search for two separate paths satisfying three classes each, or

attempt another division into smaller aggregates is an example of the problem which must be addressed.

The second consideration is that once a path has been computed for the aggregated service classes, it is necessary to reserve the resources along the path traffic will take. There must be a signaling mechanism in the network which indicates to the intermediate routers along the path to be aware of this new incoming class aggregate, and to inform them how to differentiate and handle each based on their EXP values.

Finally, once the path has been established from end to end with all intermediate routers aware of how to handle the traffic along the new label switched path, the updated state information of the network (now with reduced resource availability along the established path) must be advertised to the rest of the network. In this way future path computations for new connections will accurately take into account the current network state.

1.3 Approach

There exists a constrained shortest path first algorithm in use in MPLS networks, which is an extension to Dijkstra's shortest path algorithm [30]. If we consider the network topology as a graph, then each edge/link which does not satisfy a certain requirement, for example minimum delay, is pruned from the graph. This is repeated for each constraint prior to the shortest path first algorithm being run. This functionality is available in traffic engineering extensions to Open Shortest Path First, OSPF-TE [16], a link state and path computation protocol which includes the means to advertise the network state, and is therefore the candidate for use in this work. The meaning of "shortest path" has been extended to "least cost", which includes a cost function returning the cost of alternative paths [23]. The goal of approaching the path computation portion of this work from a cost-based perspective is that cost can be dynamically updated to reflect resource usage, whereas using only the least number of hops as criteria for the optimal path can lead to network imbalance and congestion.

The current label distribution protocol, RSVP-TE, has been extended to allow for the signaling of multiple aggregated classes of service along a single path. The protocol was designed to be extensible, with the ability to add optional data objects into the signaling messages. Therefore, a new data object has been added to the protocol which informs routers how to differentiate the classes of service by their EXP bits or the label, that is, the new method and existing method should not be mutually exclusive.

The proposed extensions have been implemented in the OMNeT++ discrete event simulator [27]. OMNeT++ is a popular network simulator which provides existing MPLS functionality in an open source, extensible environment. The simulation results presented in chapter four were derived from simulations run with the purpose of evaluating the overall efficiency of the class bundling approach to Quality-of-Service routing in MPLS networks. A comparison with the traditional method of establishing a single label switched path for each class-of-service individually is the core objective of the simulation.

1.4 Contributions

An overall reduction of the number of paths which must be established in an MPLS network contributes to the scalability of modern networks and IP/MPLS backbone switches, which may have hundreds or thousands of paths that must be maintained. Network state maintenance is an important benefit of reducing the number of paths which must exist. It is easier to troubleshoot, administer, protect and restore fewer paths. Additionally, the ability to carry multiple types of traffic from the same customer over a single LSP has implications for service level agreements between customers and service providers. Customer traffic over a single path can be monitored more stringently for robust availability and adherence to desired service levels for each class-of-service.

The implementation and results of the simulation as presented in chapters 3 and 4 contribute to the knowledge base of MPLS networks in relation to Quality-of-Service

routing and algorithms for aggregating multiple classes of service into a single label switched path. The comparisons made between methods of bundling traffic, as well as a contrast with the traditional approach, have not previously been conducted despite some theoretical discussion in this regard in the MPLS community. It is hoped that this work provides evidence of the results of a real deployment and may be considered a precursor to such developments in the field.

Chapter 2

Background and Motivation

2.1 MPLS Overview

Multi-protocol Label Switching (MPLS) is a technology which evolved in the mid/late 1990s with the initial goal of bringing the switching speed of Layer 2 of the Open Systems Interconnection (OSI) Reference Model [30] to Layer 3. In the OSI model, of which only a subset is most used in practice today, the functions of protocols are divided into a series of layers. Layer 2 resides directly above the physical layer and is called the data link layer. Here the procedural and functional means to transfer data over a physical medium is addressed. Examples of layer 2 technology include ATM , Frame Relay and Ethernet [30]. Layer 3 is termed the network layer, providing the means for performing routing, segmentation, error and flow control, with routers operating at this level. The Internet Protocol (IP) is by far the most widely known example of a layer 3 protocol. MPLS is generally considered to reside between layers 2 and 3 of the OSI model and is thus often referred to as "layer 2.5".

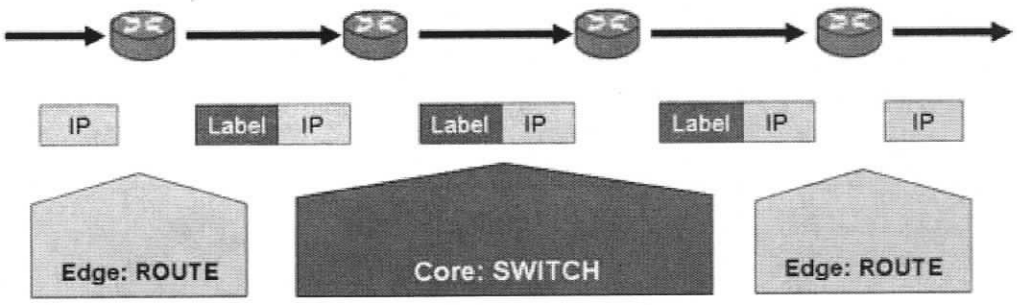


Figure 2.1 : MPLS operation at domain edge and core

In an MPLS network, packets are forwarded (switched) along a Label Switched Path (LSP) by Label Switch Routers (LSRs) [29] which make forwarding decisions based

on the value of a label added to the incoming packets as they arrive at an LSR residing at the edge of an MPLS network. Such an ingress/egress LSR is commonly referred to as an ingress/egress Label Edge Router (LER). Where IP routers must perform more complex calculations for routing decisions at each hop in the path from source to destination, an intermediate LSR in a path through an MPLS network is required only to consult the label of an arriving packet, perform a simple operation (typically a "swap" which removes the existing label and inserts a new one), and forward the packet on to the next LSR. There is a gain in switching speed with this methodology, however the ever increasing speeds of layer 3 routers make this no longer a primary justification for MPLS. More important today, MPLS brings with it benefits such as:

- *Tunneling mechanisms for virtual private networks*: MPLS packets have one or more labels added to a payload, which may be an IP packet or any of a number of other kinds of payload packet. To intermediate nodes in an MPLS network the route and payload are opaque, and these LSRs perform operations on the prepended label(s) only. This "tunneling" insulates the original packet and creates the illusion of a tunnel through which the wrapped packet travels across the intermediary network.

- *The ability to create circuits across multiple layer 2 transport mediums*: Due to the generality provided by MPLS labels, there is the opportunity to carry these packets over different link layer technologies. For example, MPLS can make use of existing ATM network infrastructure, as its labeled flows can be mapped to ATM virtual circuit identifiers. Also, Generalized MPLS (GMPLS) [24] is emerging when there is a need to interconnect routers, SONET switches, and other optical communication devices.

- *The ability to establish a fixed path with specific performance characteristics:* The feature of MPLS which is the focus of this work, and any endeavor in traffic engineering/management is to provide Quality-of-Service (QoS). QoS is defined here as applying and ensuring specific, quantifiable performance levels on a

network. MPLS adopts portions of previous methods such as the IntServ and DiffServ models (described later), and the complex QoS signaling of ATM, with the aim of generally simplifying and providing a unified approach.

An MPLS header which contains one or more labels, collectively referred to as the label stack. Each label stack entry is 32 bits and formatted as follows:

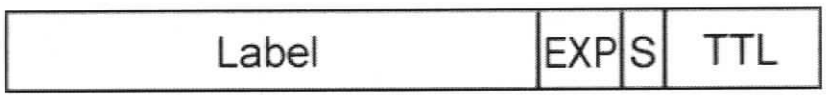


Figure 2.2 : Contents of an MPLS label stack entry

An MPLS packet encapsulates other higher layer protocols and payload data, and the header may contain multiple label stack entries, with S=1 indicating the bottom entry of the label stack:

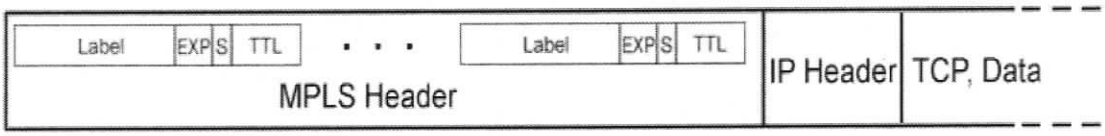


Figure 2.3 : Example MPLS packet and label stack

At the time the MPLS label stack entry was initially designed, the protocol was given "room to grow" by incorporating three experimental bits with no pre-defined usage. The individual field sizes and meanings are shown below.

Table 2.1 : MPLS label stack entry field sizes and meanings

Field Name	# of Bits	Meaning
Label	20	Label value
EXP	3	Experimental for future use
S	1	Bottom of stack (boolean)
TTL	8	Time to live

When a labeled packet arrives at an LSR, the topmost label of the label stack is examined. Based on a table lookup which determines the operation to perform according to the contents of the label field, an LSR may initiate a swap, push, or pop on the label stack.

In a push operation a new label is added to the top of the stack, serving to encapsulate the packet in another layer of MPLS. This "nesting" allows for the hierarchical routing of MPLS packets. This is notably used in virtual private networks mentioned previously.

When performing a pop operation, the label is removed from the packet. If another label exists below the one removed from the stack, the packet has not yet reached its destination and is forwarded on. Otherwise it is at or directly before the end of its label switched path. The egress LER does not always perform the final pop operation: In some situations a Penultimate Hop Pop (PHP) [29] is performed by the LSR adjacent to the egress LER prior to forwarding, to offload some of the work which must be done at the egress point when the packet leaves the MPLS network.

For a swap operation the topmost label is removed and a new label is applied. During transit the contents of the packet below the MPLS label stack are not considered. Most commonly a path through an MPLS network will begin with a push operation at the ingress node, a sequence of swap operations along intermediate nodes, and a final pop operation at the egress node. This allows for protocol independent forwarding which does not require a protocol specific routing table.

A Forwarding Equivalence Class (FEC) [29] is a set of packets which may be handled as if they are equivalent for the purpose of forwarding and consequently they can all be bound to a single label. The figure below shows IP traffic arriving at an MPLS domain, where it is divided into FECs at the ingress LER by pushing label L1 onto packets IP1 and IP3, and label L2 onto packet IP2. At each intermediate LSR the label values L1 and L2 are swapped for new ones, however IP1 and IP3 will share the same label value throughout as they are part of the same FEC. At the egress node LSR4,

packets IP1 and IP3 are sent on to different destinations following a pop operation, as determined by the egress LER. The mapping of table values for FEC to next hop label forwarding entry dictates the operation(s) to be performed on packets at each LSR.

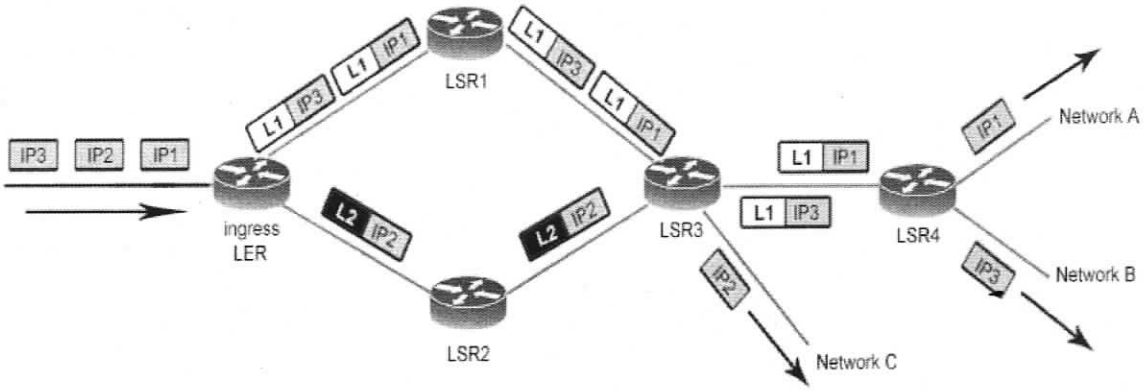


Figure 2.4 : Treatment of IP packets in an MPLS domain

MPLS Traffic Engineering or MPLS-TE [1] uses native traffic engineering mechanisms to minimize congestion and improve performance while accommodating differing service levels (bandwidth requirements, drop priority, delay, jitter, uptime, etc). Through the extension of traditional IP protocols, constraint-based and explicit routing allow an ingress LER to compute a path which meets the requirements of the traffic flow within an MPLS network.

Constraint-based routing is optional in MPLS networks. When left unspecified, the Label Distribution Protocol (LDP) serves to establish label bindings to packets without path or service constraints, and to notify peer LSRs of such mappings so that correct swap operations by intermediate nodes occur. For distributing labels and bindings to FECs where resource requirements must be met, there exist two primary protocols: Constraint-based LDP (CR-LDP) [15] and the Resource Reservation Protocol with Traffic Engineering extensions (RSVP-TE) [4]. RSVP-TE is an extension to the IP-based RSVP protocol, while CR-LDP is derived directly from LDP. Both methods developed simultaneously and bear many similarities, however RSVP-TE has emerged as the

dominant protocol for MPLS-TE and development on CR-LDP has ceased [3]. As such CR-LDP will not be mentioned further here.

2.2 RSVP-TE

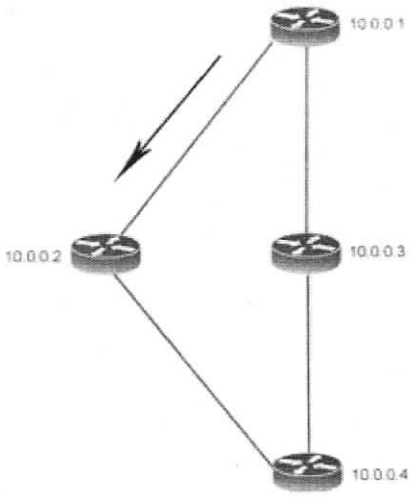
The Resource Reservation Protocol RSVP [8] was designed with the original intention of supporting real-time applications over the Internet. Key applications which could benefit from network-wide deployment of RSVP never materialized. Instead RSVP-TE, an extension for traffic engineering, has been widely accepted by network providers to support MPLS applications. The RSVP-TE protocol is an addition to the RSVP protocol for establishing label switched paths in MPLS networks. It supports the instantiation of explicitly routed LSPs with or without resource reservations. RSVP-TE also supports fast rerouting of LSPs, pre-emption, and loop detection.

Networks which support both MPLS and RSVP-TE can map labels to traffic flows. Once a label switched path is established, the traffic through the path is defined by the label applied at the ingress (edge) node of the LSP. At each intermediate hop along the path from source to destination, the existing label is swapped with a new one which will indicate treatment at the next downstream node. The set of packets that are assigned the same label value by a specific node are said to belong to the same FEC, which serves to define the RSVP flow. When traffic is mapped onto an LSP in this manner, we call the established path an LSP tunnel. The extensions to RSVP for traffic engineering include new Session, Sender and Filter Spec objects which support the LSP tunnel feature. The Session object defines a traffic engineered tunnel and contains a tunnel ID as part of the Session object. The Sender and Filter Spec objects carry an LSP ID. The Sender (or Filter Spec) object, together with the Session object, uniquely identify an LSP tunnel.

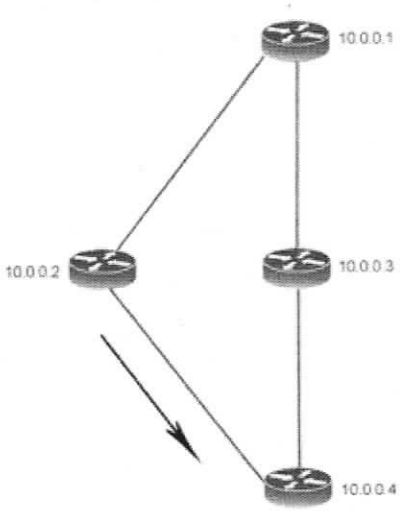
An advantage of using RSVP-TE to create LSP tunnels is that it allows for the reservation of resources such as bandwidth along a path. LSPs can be instantiated without reservation of resources for a variety of uses including recovery policies or best effort

traffic. However, the ability to use RSVP reservation mechanisms to ensure bandwidth requirements along LSP tunnels is our primary concern.

The following simple example shows the interaction between LSRs to establish an LSP via the RSVP-TE protocol. An explicit path has been computed by the LSR with address 10.0.0.1 and labels are allocated at each LSR seeing the LABEL_REQUEST object. The upstream return message reserves resources along the path and distributes label bindings to peers. As RSVP maintains a "soft state", this information must periodically be refreshed.



1) RSVP Path Message, LABEL_REQUEST,
ERO = {10.0.0.1, 10.0.0.2, 10.0.0.4}



2) RSVP Path Message, LABEL_REQUEST,
ERO = {10.0.0.2, 10.0.0.4}

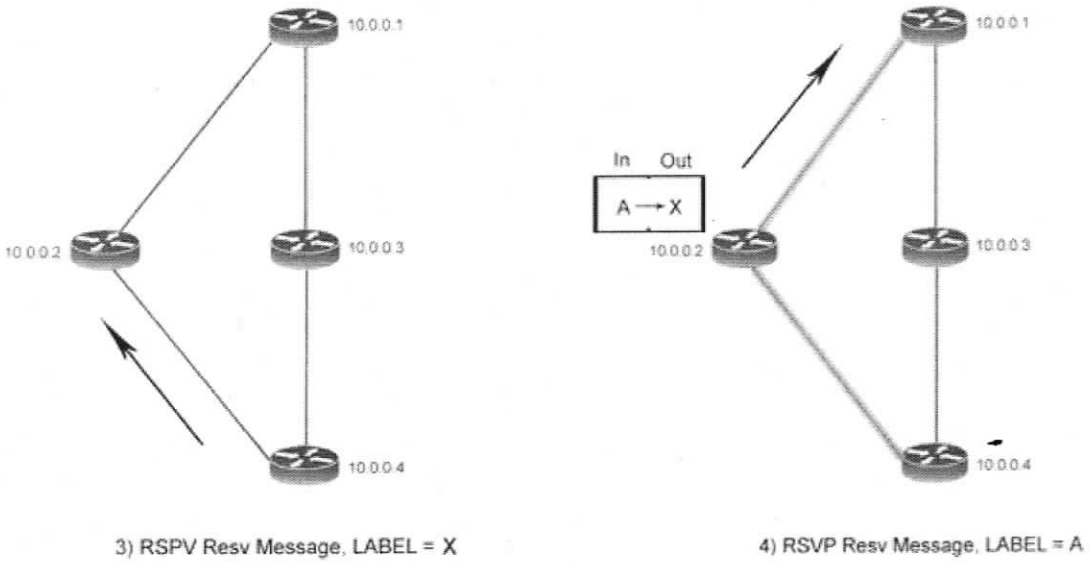


Figure 2.5 : MPLS label bindings and explicit routing via RSVP-TE

Following the operations shown above, LSR addressed 10.0.0.2 will swap an incoming label 'A' with 'X' and transmit to 10.0.0.4

A basic RSVP reservation request message consists of two objects which define the requirements and classification of data packets, respectively. The Flow Spec specifies the level of service which is desired. The Filter Spec defines the set of packets to receive the Quality-of-Service level defined by the Flow Spec. Together the Flow Spec and Filter Spec comprise the Flow Descriptor. A request to bind labels to a specific LSP tunnel is initiated by the LER at the ingress point to the network via the RSVP Path message. To make this possible the RSVP Path message is augmented with a LABEL_REQUEST object. Labels are allocated downstream and propagated upstream via the RSVP Resv message. The RSVP Resv message is thus also augmented with a new LABEL object.

The RSVP-TE protocol supports explicit routing by incorporating an Explicit Route object (ERO) [4] into Path messages. The ERO contains a list of all of the hops from source to destination which will constitute an explicitly routed path. This permits paths taken by label switched flows to be pre-determined, either by administrative

specification or automatic computation via a suitable algorithm which takes into account the current network state, for example Constrained Shortest Path First described below. A useful application of explicit routing is traffic engineering. Using explicitly routed LSPs, the node at the ingress edge of an MPLS network can direct the path through which a specific flow of data traverses the network from itself to an egress/destination node. The ability to do so allows for optimizing the utilization of network resources and improved traffic oriented performance characteristics.

During reservation setup, an RSVP request is passed to a local decision module, admission control. This module determines whether the node has sufficient resources available to supply the requested level of service. If the check fails, the RSVP program returns an error notification to the appropriate receiver(s). Otherwise the specified resources are allocated and the reservation request continues to propagate upstream to the originator. There are three styles of reservation specified in RSVP: Wildcard, Shared Explicit, and Fixed Filter [8]. For the remainder of this work it is expected that Fixed Filter style is being used, whereby a distinct reservation for packets from a particular sender is made, and no sharing with other senders' packets occurs.

2.3 MPLS DiffServ

The first model developed as an attempt to accommodate Quality-of-Service in traditional IP networks was Integrated Services (IntServ) [7], whereby applications directly requested QoS guarantees from the network. For this purpose the RSVP signaling protocol was used to distribute such requests to the nodes in the network. The huge numbers of traffic flows in IP networks made this approach very complex and unscalable, as state information had to be maintained for each flow at every hop. To remedy this deficiency, another model was designed called Differentiated Services (DiffServ) [26]. The problem is instead approached in the DiffServ model by partitioning traffic into classes and allocating resources within the network on a per-class basis. To avoid the need for a signaling protocol, the class is marked directly on the IP header in a

6-bit DiffServ Code Point (DSCP) field. The DSCP field is part of the original Type-of-Service field in the IP header. The meaning of this field was altered to become a 6-bit DSCP field and a 2-bit congestion notification field. The DSCP value indicates QoS behaviour of a packet at a particular node in the network. This is called the per-hop behavior (PHB) [26] and is expressed as scheduling, queueing, resources and drop preference that a packet receives. DiffServ provides differential treatment to traffic which enforces QoS for different traffic flows relative to one another. Unlike IntServ, the DiffServ model does not require a signaling protocol or state maintenance, and is thus a scalable solution. The drawback which exists with DiffServ is that if the path traffic follows in the network does not have adequate resources, there can be no guarantee of meeting QoS requirements.

More recently, work has been done to create mechanisms for MPLS support of DiffServ [22]. As LSRs in an MPLS network make their forwarding decisions based solely on the MPLS header, any per-hop behaviour which should be applied must be inferred from information contained therein. Fortunately there exist the three EXP bits which can be used to carry DiffServ information. However, another difficulty arises: the original DSCP field in the IP header can accommodate up to 64 values with the six bits available. These must then somehow be mapped into the three bits of the EXP field, with a maximum of 8 values. Two solutions to this problem have been developed and are in use today.

If a network is to support less than eight PHBs, the first solution is a simple one of mapping particular DSCPs to equivalent EXP values. These values are configured either administratively or according to the DSCP value of the IP packets to be carried in the LSP, rather than signaled when the LSP is established. Such LSPs are called EXP-Inferred LSPs (E-LSPs), and can carry packets with up to eight distinct PHBs in a single LSP. It is an important distinction to note that all such E-LSP traffic is still treated as a single class type. Essentially, the current solution is designed to support per-class routing, the case where different classes of traffic for a single user are still sent over different LSPs.

The second solution applies to scenarios in which support for more than eight PHBs is required. Here the three EXP bits alone cannot contain the required information to allow for differentiation between all possible PHBs. This leaves use of the label field itself. During forwarding, the label indicates where to forward the packet and what scheduling behavior (QoS class) to apply, and the EXP bits designate the drop priority to assign to the packet. The PHB is thus determined from the information contained in both the label and EXP bits. This information needs to be conveyed when the LSP is signaled because the label is now intrinsically a part of the PHB. These LSPs are called Label-Inferred LSPs (L-LSPs). L-LSPs can carry packets from a single PHB, or from several PHBs that have the same scheduling requirements but differ in their drop priorities.

2.4 Traffic Engineering

The goal of traffic engineering is to find a path in the network that meets certain constraints imposed by the type of traffic being carried. In order to calculate the path to a destination, these constraints must be taken into account. Examples of typical constraints could be the amount of bandwidth required, the amount of jitter or end-to-end delay, and the number of hops (intermediate nodes) which exist in a path. Many additional constraints could also be applied to a traffic request such as path diversity or exclusion, administrative cost, pre-emption, etc.

The computation of a path which satisfies the given constraints requires that information about resources and performance characteristics of links in the network be maintained, and distributed to the node(s) responsible for path calculation. That is, relevant link metrics must be advertised through the network. This is achieved by adding TE extensions to the link state protocol OSPF which allows for advertising not just the up/down state of links, but the administrative attributes and bandwidth available for reservation by LSPs. This information about resource availability and usage is contained in a Traffic Engineering Database (TED) [18] which can be queried by routers seeking

resource availability information for the purpose of path computation. Based on periodic OSPF TE updates, each LER collects the information about link bandwidth and any other properties that are propagated and stores them in the TED. The traditional link state database and the new traffic engineering database are used to determine the optimal end-to-end path through the network over which to signal and establish an LSP. As is true with the link state database, the traffic engineering database has a consistent view from all the routers in the same area and reflects the changes in resource usage. This means that links advertised in the direction of an LSP will indicate the updated resource information upon path set up.

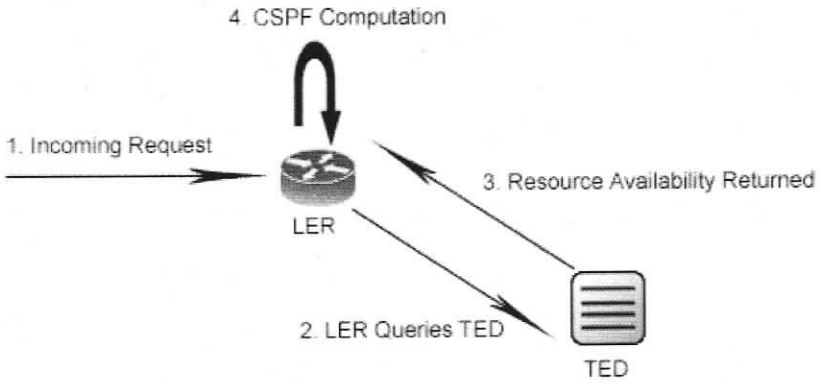


Figure 2.6 : Interaction between a label edge router and traffic engineering database

With the network resources and administrative attributes contained in the TED available, a modified version of Dijkstra's shortest path first (SPF) algorithm called Constrained SPF (CSPF) may be used by the ingress LER to calculate a feasible path compliant with the given constraints. Finding a path through the network subject to multiple constraints is known as the Multi-Constrained Path problem (MCP), also termed "QoS routing", and is generally made tractable through the use of heuristics [20], [31]. One such method occurs in CSPF, which operates in the same manner as SPF, except it first prunes from the topology all links which do not satisfy the constraints. For example, if the constraint is bandwidth, the algorithm will first prune from the topology all links with insufficient bandwidth. In [23] is described a method of including a cost function which can be applied to the CSPF algorithm for the inclusion of those constraints which cannot

be subject to pruning, for example source-destination fixed delay or load balancing across the entire network. This cost function can then be used to determine a least cost path on the remaining topology after pruning occurs. This is the method which has been selected for implementation in the simulation program (see sections 2.7, 3.6).

2.5 DiffServ-TE

For situations in which it is necessary to have service guarantees per class (for example, maintaining good quality voice traffic in the presence of other lower priority data traffic) a combination of DiffServ and traffic engineering methodologies were proposed in [21], introducing the concept of a Class Type (CT). A CT is defined as a set of traffic on a link which is governed by a set of bandwidth constraints. Class type is applicable to constraint based routing, admission control, and link bandwidth allocation. Up to eight CTs are required to be supported. LSPs which are traffic-engineered to guarantee bandwidth for a particular CT are referred to as DiffServ-TE LSPs. Introducing class types implies that we must now keep track of how much bandwidth (and perhaps other attributes) is available for each class of traffic. Bandwidth is partitioned amongst traffic classes according to a bandwidth model, of which the primary candidates are described in section 3.3.

Additionally, there must be an extension to the path computation module whereby CSPF is enhanced to take into account bandwidth for a given CT. The available bandwidth per-CT must be advertised for each link and retained in the TED. This means that in order for CPSF to perform a meaningful calculation, the CT chosen for an LSP must correspond to one of the preconfigured TE-classes in the network.

Once the path is calculated, it must be signaled and have resources reserved at each hop. The necessary extensions to RSVP-TE are defined in [22]. Essentially there has been the addition of a CT object in the RSVP path message which describes from which class the bandwidth is being requested. For interoperability, the absence of a CT object

implies a specific class type value of zero. A CT value is necessary to perform the calculation of available resources.

DiffServ provides for the scheduling of each type of traffic. Therefore, the combination of DiffServ and per-class traffic engineering allows for strict service guarantees. In accordance with the current model of the Internet Engineering Task Force, a DiffServ-TE LSP can only carry traffic *from one class type*, regardless of whether it is an EXP-Inferred or Label-Inferred LSP (E- or L-LSP). In the case where E-LSPs have PHB designated in the EXP bits of the label stack entry, these multiple classes are treated as a single class for path computation, route advertisements, signaling and admission control.

2.6 Motivation for Multi-Class E-LSPs

As mentioned previously, current DiffServ-TE solutions are restricted to L-LSPs or E-LSPs which carry only a single class-of-service. Work has been proposed [13] to extend these solutions to include carrying multiple classes of service per E-LSP, using the 3 EXP bits of the MPLS label stack entry to indicate class-of-service. This has been developed in such a way as to be an optional extension and to coexist with existing practices, as it is understood that not all MPLS networks would need support for multiple classes of traffic per LSP.

There exist many reasons for MPLS network providers to carry multiple classes of service on a single LSP. The key points can be summarized as follows:

- *The ability to carry multiple classes of traffic belonging to a single customer on the same LSP*: By way of example, consider an MPLS service provider required to honour a strict Service Level Agreement (SLA) which specifies a single level of path protection for all aggregate traffic from a single customer, in which there exist multiple classes of service. In such a situation sending each class-of-service on a different LSP complicates the ability to offer stringent, measurable

availability. Another example is bandwidth borrowing between classes carried on the same traffic trunk; that is, if a customer's high priority traffic is using less than its allocated resources, other best effort traffic from that same customer may receive preferential treatment. Additionally, when issues related to customer traffic arise, it is easier for the network operator to examine a single LSP instead of multiple LSPs.

- *Overall reduction of the number of LSPs in the network*: If more than one class-of-service can be carried over a single LSP, the total number of LSPs in the network will be reduced by a factor of at least two for each such multi-class E-LSP established, greatly increasing scalability. The benefits of reducing the number of LSPs in an MPLS network include less RSVP state maintenance, faster restoration times when links must be re-signaled after going down, more direct application of path protection, and simplification of maintenance and administration.

2.7 Required Extensions

When attempting to aggregate or "bundle" multiple classes of service into a single E-LSP, extensions must be made to the signaling protocol, RSVP-TE, for the inclusion of multiple traffic requirements/constraints per class. To accomplish this, a new ELSP object is defined, and must be present in both the PATH and RESV messages. E-LSPs that do not wish to signal multiple traffic requests on a per-class basis do not include this object. An example of how such an ELSP object may be formatted appears in [12]. In the simulation program, it has been implemented as an array extension of the existing RSVP-TE message constructs.

Additionally, more robust path computation and admission control is required, here focusing on OSPF-TE and the CSPF algorithm. The inclusion of multiple classes of service, each with its own set of constraints, further complicates the MCP or QoS routing

problem, as it must now be applied iteratively for each class-of-service. The basic MCP problem can be defined as follows [31]:

Let $G\ (N, E)$ denote a network topology, where N is the set of nodes and E is the set of links. Each link $u \rightarrow v \in E$ is specified by a link weight vector with as components m additive QoS link weights $w_i\ (u \rightarrow v) \geq 0$ for all $1 \leq i \leq m$. Given m constraints L_i, where $1 \leq i \leq m$, the problem is to find a path P from a source A to a destination node B such that

$$w_i(P) = \sum_{(u \rightarrow v) \in P} w_i(u \rightarrow v) \leq L_i$$

When considering the requirements of multiple distinct classes of service, a connection request may be considered a matrix R consisting of constraint vectors, where entry r_{ij} is the i'th constraint of class j. For each link, there is an equivalent matrix W of link weight vectors. Entry w_{ij} is the i'th cost for class j. In the simplest case, the path cost matrix C is the sum of all matrices W along path P. The inequality in the formulation above is thus extended such that this new matrix C have as entries $c_{ij} \leq r_{ij}$ for $0 \leq j \leq n$ for all $1 \leq i \leq m$, where n is the number of classes of service forming the aggregate connection request, and m remains the number of additive QoS link weights.

A path which satisfies each of the constraints is referred to as a feasible path. For a feasible path to exist in the case of multi-class E-LSPs, there must exist a path which is feasible for all of the classes together. There may be multiple paths through the network which satisfy the given constraints. According to the definition above, any of these solutions is viable. However, it is desirable for us to select the path with the least cost. "Cost" in this work refers to the smallest value retrieved from the cost function, whose implementation within the simulation is described in chapter 3 and completely in [23]. Where no feasible path exists to accommodate all traffic classes together, separation and different combinations of classes must occur, or else the entire multi-class request will be refused.

As the goal of this work is primarily to explore the validity and efficiency of aggregating/bundling multiple classes of service in practice, including algorithms for the combination of traffic classes, the above formulation serves only as a guide to help define the problem space. Further implementation details for path computation and admission are addressed in section 3.6.

Chapter 3

Implementation and Configuration

3.1 Simulation Environment

The simulation environment selected for implementing the extensions to MPLS was the INET framework for the OMNeT++ discrete event simulation system [27]. OMNeT++ is a component-based, modular and open-architecture simulation environment, whose primary, but certainly not only application is communication networks. The INET framework [14] is an open-source simulation package written in C++ for OMNeT++ which contains models for several Internet protocols beyond TCP/IP, including Ethernet ARP, PPP, RTP, and most importantly MPLS with LDP and RSVP-TE signaling.

The INET framework was selected because of the existing modules with MPLS and RSVP-TE functionality which could be extended as needed. The flexibility granted by the OMNeT++ Network Definition (NED) language allowed for the rapid creation of various network topologies and link characteristics, while INET supplied inbuilt LSR signaling and routing facilities.

Specifically, the modules which were extended or rewritten are as follows:

RSVP and RSVPApp: Combined these modules have all the required RSVP-TE functionality. RSVP is responsible for all aspects of establishment and maintenance of the RSVP protocol, handling PATH, RESV, tear and error messages, creating and maintaining Traffic/Flow Spec objects and Path/Resv/Traffic Control State Blocks. This module also communicates with the TED to update link costs and capacities. RSVPApp is the interface between the MPLS switch and RSVP. Its primary purpose is to initiate

traffic control and signaling messages in the RSVP module and to reflect the results of the RSVP protocol to the MPLS switch in the form of FEC bindings.

TED: The Traffic Engineering Database maintains all link costs which are used in path computation by the OSPF module when performing path computations. The calculation of link costs is discussed in section 3.6. Additionally, the TED maintains all link capacities and metrics such as delay, amount of bandwidth available and allocated, and gathers statistics such as link and network utilization for final reporting.

OSPF: Path computation is handled entirely by this module. OSPF is invoked by RSVPApp prior to initiating PATH establishment to the RSVP module. RSVPApp invokes the OSPF module with a request containing the source, destination and traffic parameters of a single (or more importantly group) of incoming connection requests. OSPF then performs the modified Dijkstra computation described below, and if a feasible path exists returns an Explicit Route Object (ERO) which is then forwarded to RSVP to begin resource reservation in the network. OSPF queries the TED for link costs and resources when performing its computations.

The aforementioned modules encompass the core functionality which was extended to provide for multiple classes of service and metrics per link, as the INET framework did not initially accommodate any such differentiation.

It is the RSVPApp module which generates connection requests based on traffic parameters specified in an XML template file. This information is parsed and sent to the RSVP module. For this reason it was decided that RSVPApp would also be responsible for aggregating classes of service for E-LSP signaling, distinguished here as multi-class bundling and not the single class E-LSPs of chapter 2.

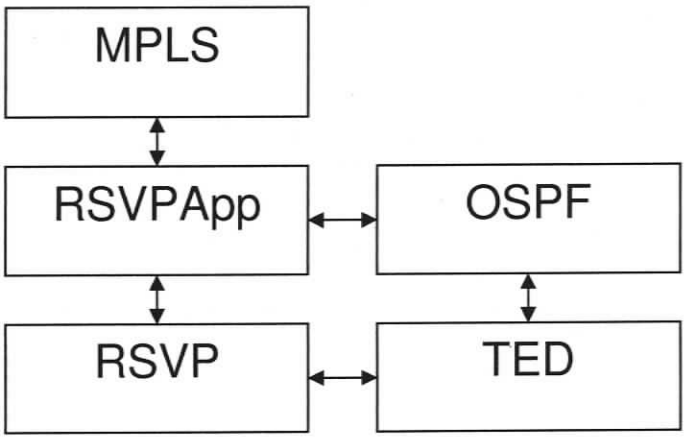


Figure 3.1 : Interaction of core simulation modules

3.2 Traffic Model

Incoming connection requests arrive at the LER in one of two ways: as a single request of the form *(req_bandwidth, max_delay, class_type)* in the case of L-LSP establishment, or as a bundle of multiple requests *(req_bandwidth0, max_delay0, class_type0) , ..., (req_bandwidthN, max_delayN, class_typeN)* where *N* is the number of classes being aggregated into a single path via E-LSP establishment. The values for *req_bandwidth*, *max_delay*, and *class_type* are derived from a template XML file by the RSVPApp module. When the L-LSP method is being used, requests arrive in cyclical order. For example, if we differentiate four classes of service then class 0 will arrive, followed by class 1, class 2, finally class 3, and the sequence repeats itself beginning with class 0. When the E-LSP method is being used, requests always arrive at the LER together, with each class representing a single entry in the group of requests.

Regardless of which method is being used, the values for bandwidth and delay are based only on class type and not LSP method. If the following set of values is specified in the traffic template file:

Table 3.1 : Sample traffic request template values

Bandwidth (kbps)	Delay(ms)	Class-of-service
128	10	0
64	20	1
32	30	2
16	40	3

Then with no bundling (L-LSP method) the single incoming connection requests will be of the form

Request 1: *(128,10,0)*
Request 2: *(64,20,1)*
Request 3: *(32,30,2)*
Request 4: *(16,40,3)*
Request 5: *(128,10,0)*
etc...

However when doing aggregation/bundling (E-LSP method) incoming connection requests arrive together as

Request 1: *(128,10,0), (64,20,1), (32,30,2), (16,40,3)*
Request 2: *(128,10,0), (64,20,1), (32,30,2), (16,40,3)*
etc...

If the number of connection requests for the L-LSP method is L, the number of connection requests for the E-LSP method is E, and the number of differentiated classes of service is C, the number of connection requests will be equal between the two methods if $L = EC$. For all runs conducted in this simulation, the total number of connection requests for the L-LSP method is set at 2000, and either 500 or 250 for the E-LSP

method, depending upon whether there has been a differentiation of 4 or 8 classes, respectively.

For both methods, the timing of incoming connections has been set to follow a uniform distribution between 1 and 5000 seconds, approximately one and a half hours total, or 24 connection requests every minute of simulation time.

For convenience the convention adopted is that lower numbered classes always have higher priority, and in addition more stringent requirements (higher bandwidth, lower delay tolerance) than higher numbered classes. In all simulations the requirements of connection requests decrease as class number increases. This could easily have been done in a reverse or arbitrary manner but for consistency that was never the case.

In addition to the connection requirements being specified in a template traffic file, a parameter was added to the simulation to allow for some variability of bandwidth between requests. If the bandwidth specified in the XML template file for a certain class-of-service is B and the variability parameter is P, when a connection is generated it is assigned a bandwidth following the normal distribution with mean B and standard deviation P.

3.3 Bandwidth Model

Three bandwidth models were implemented within the simulation program. These are termed the Capped, Shared, and Russian Doll Models. Each is a method of partitioning available bandwidth on a link among different classes of service. They differ only with regards to how the bandwidth is split, and whether or not borrowing is allowed between classes.

In the Capped bandwidth model, the available bandwidth on a link is divided into partitions, one for each class-of-service available in the network. This is also known as

the Maximum Allocation Model [19], but here without allowing overbooking of bandwidth (whereby bandwidth above the initial amount allocated to a class can be reserved if available). When a class-of-service has used all of the bandwidth it has been allocated on a link, no further reservations will be met regardless of available link bandwidth, as it has all been allocated to other classes.

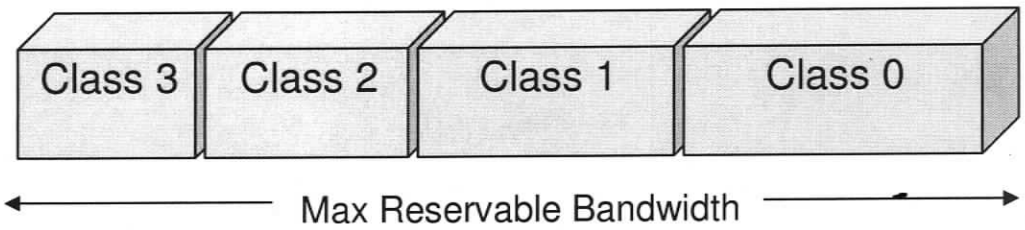


Figure 3.2 : Allocation of bandwidth to class types in the Capped bandwidth model

For the Capped bandwidth model a decision must be made as to how to divide the total link capacity among the classes of service. The approach implemented in the simulation was to use weighted partitioning of bandwidth, where the weight assigned to a class-of-service is calculated as the fraction of bandwidth such a class will request over the sum of bandwidth requested by all classes. Using some values from the simulation to illustrate:

Given link capacity 96000 kilobits and four classes of service requesting 128, 64, 32, and 16 kbps bandwidth (respectively, classes 0 to 3) on average, then the weight for class 0 is calculated as 128 / (128+64+32+16) = 0.533. Similarly 0.267, 0.133, 0.067 for the other classes 1 to 3. The partitioning of bandwidth for each class (0 to 3) is thus 51168, 25632, 12768, and 6432 kbits.

The Russian Doll model (RDM) [20] was also implemented. It functions in a similar manner to the Capped model, with the exception that higher priority classes can "borrow" bandwidth from any classes of lower priority if they have reached their initial cap. Whereas in the Capped model a class of traffic cannot make use of the bandwidth

left unused by another class, the RDM can do so provided it is making use of unused bandwidth from a class of lower priority.

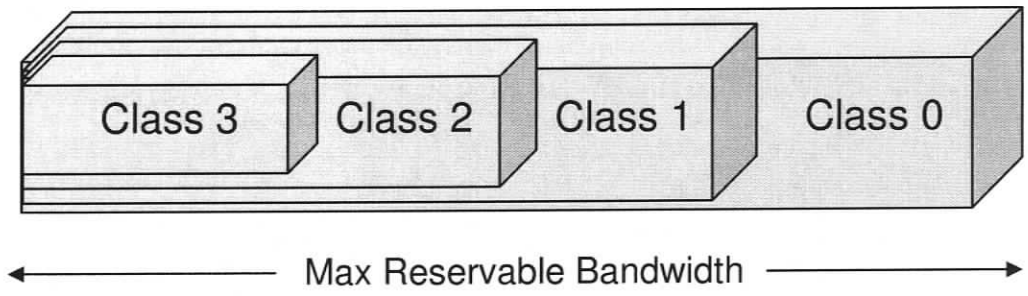


Figure 3.3 : Allocation of bandwidth to class types in the Russian Doll bandwidth model

The figure above shows how the RDM resembles its namesake the Russian toy; there is one big doll (highest priority class 0), and within is nested a smaller, and so on until the smallest doll (lowest priority class 3). The amount of available bandwidth per class is partitioned initially as the capped bandwidth model, but for example the available bandwidth per class can be as great as (using values from above):

Table 3.2 : Sample bandwidth availability per class-of-service using the RDM

Class-of-service	Bandwidth Available per COS	Total Available Bandwidth
0	51168 + 25632 + 12768 + 6432	96000
1	25632 + 12768 + 6432	44832
2	12768 + 6432	19200
3	6432	6432

Finally, the Shared bandwidth model is a global pool of bandwidth available to all classes of service, which could be considered no model at all. Each request, regardless of class-of-service, can reserve bandwidth on a link as long as it is available. There is no differentiation between traffic classes and no partitioning of link bandwidth is made.

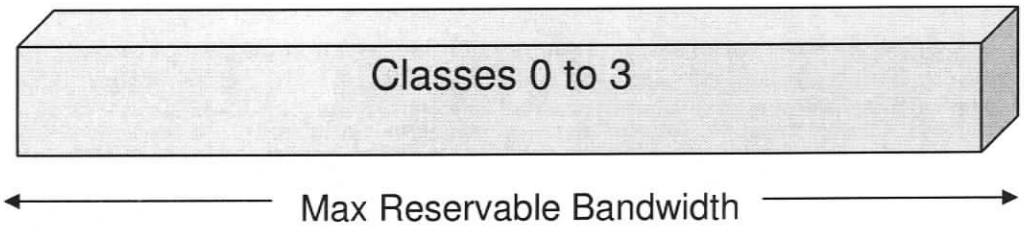


Figure 3.4 : Allocation of bandwidth to class types in the Shared bandwidth model

3.4 Bundling Algorithms

The progression of events which lead to the availability of a group of connection requests at the ingress LER, prepared for bundling, is as follows: Initially the network connectivity, link capacities and number of classes of service are specified for the simulation. Next when the simulation begins, timed events occur to trigger the creation of connection requests by the RSVPApp module at the LER. When a set of connections "arrive" in this manner, the traffic parameters for each class-of-service we are differentiating in the network are read. Any variability which has been specified is calculated, and then this group of connection requests is ready to attempt admission into the network. It is at the point when we're ready to do connection admission control (CAC) that the bundling algorithm in use attempts to establish a minimal number of E-LSPs in the network and route(s) for same. These are then handed over to the RSVP and TED modules for path reservation.

While aggregating traffic classes into as few E-LSPs as possible via bundling is of great importance in reducing the total number of LSPs in the network, the overhead of additional invocations of the path computation algorithm must also be considered in analyzing the merit of any method. What follows is a description with pseudocode of three algorithms adopted into the simulation program for achieving successful bundling, and two additional variations with particular applications. Some improvements, tradeoffs and other characteristics of these algorithms are expanded upon in the next chapter, alongside an analysis of the simulation results.

Three main methods of aggregating traffic classes into a single LSP were implemented; these are described as Additive, Subtractive, and Hybrid. As the names suggest, additive begins with a single connection request and adds successive classes to it, increasing the bundle size to the maximum the network will tolerate. Subtractive works in a reverse fashion, beginning with the maximum number of classes within the network as a single bundle and reducing its size iteratively until the request is acceptable. Hybrid bundling is a combination of additive and subtractive methods. Leveraging the notion of temporal locality, that is what has come before is likely to occur again, the hybrid method begins subtractively, and upon finding a successful bundle stores this information for immediate use when the next set of connection requests arrive. This new set of connection requests is divided exactly as the last successful admission into the network, with the subtractive algorithm being applied if it is unsuccessful, and the additive algorithm attempting to create a new bundle of those connections which remain (ie. were not successful previously). Finally, two variations for the additive and subtractive methods are included. In these variations a preprocessing step is added to check each connection request individually in the network prior to any attempt to do aggregation.

Let us assume we can invoke the modified OSPF algorithm via a function call `computePath( TrafficRequest[] TR )` which returns true if all of the traffic requests contained in the array TR can be successfully admitted into the network along the same LSP and false otherwise.

Additionally we have `sendToRSVPModule( TrafficRequest[] TR )` which initiates the RSVP module when the request set is admissible into the network, and `notifyBlockingRequest( TrafficRequest TR )` which notifies the LER that the single connection request passed to it is not admissible in the network and has been rejected. The pseudocode for the subtractive algorithms is presented below; to initiate an attempt to aggregate a group of connection requests into a single LSP, the RSVPApp module invokes the function `doSubtractive( TR )`.

```
Algorithm doSubtractive( TrafficRequest[] TR )
begin
     c ← sizeOf( TR )
     TrafficRequest[] tempTR

     for i ← c-1 to 0 do
     begin
          if computePath( TR ) is true then
               sendToRSVPModule( TR )
               break
          else if sizeOf( TR ) equals 1 then
               notifyBlockingRequest( TR[i] )
               break
          else
               tempTR[i] ← TR[i]
               TR[i] ← NULL
     end

     if tempTR is NULL then
          return
     else
          doSubtractive( tempTR )
end
```

The above algorithm attempts to admit into the network a bundle of maximum size. If this is unsuccessful, remove and store the lowest priority class, and try to admit the remaining bundle of size (max-1). Proceed in this way until a bundle is admitted, or a bundle of size 1 (singleton) is rejected. In either case the previously stored connection requests are gathered and the algorithm is repeated on them. This occurs until all connections have either been admitted or blocked as singleton requests.

This method has the undesirable bundling characteristic below, with ◎ and ⊘ signifying admissible and inadmissible connection requests, respectively:

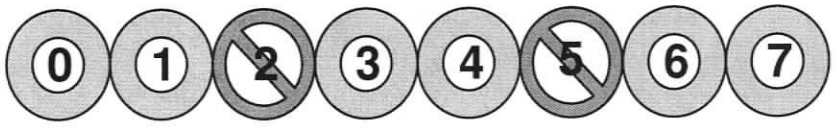


Figure 3.5: A deficiency of the basic subtractive algorithm

The result of the subtractive algorithm will be an LSP whose contents are classes (0, 1), another with contents (3, 4), and finally another (6, 7). Here instead it would be preferable for a single LSP to be established with contents (0, 1, 3, 4, 6, 7) whenever possible and not three separate LSPs. The subtractive algorithm does however work very well when there is ample bandwidth, as is evident if we consider a network where every class in the initial bundle will be admitted. The bundle will be of maximum size, resulting in a single LSP, and it will require only a single invocation of `computePath()` to accomplish this.

```
Algorithm doAdditive( TrafficRequest[] TR )
begin
     c ← sizeOf( TR )
     j ← 0
     TrafficRequest[] tempTR

     for i ← 0 to c-1 do
     begin
          if TR[i] is NULL then
               continue

          tempTR[j] ← TR[i]

          if computePath( tempTR ) is true then
               TR[i] ← NULL
               if i equals c-1 then
                    sendToRSVPModule( tempTR )
               else
                    j ← j+1
          else if j equals 0 then
               TR[i] ← NULL
               notifyBlockingRequest( tempTR[j] )

     end

     if TR is NULL then
          return
     else
          doAdditive( TR )
end
```

The additive algorithm operates in reverse fashion to subtractive. It begins with a single connection request and checks if it can be admitted into the network. If so, add the next (lower) priority level and check the bundle of size 2. If it is inadmissible, remove it and proceed to the next class. Admit the resulting successful bundle if it exists into the network. Any previously inadmissible connections which did not block as singleton connection requests are recursively checked.

The additive algorithm may perform more invocations of `computePath()` than are strictly necessary. If the highest priority class is admissible in the network but all others are not, each of the classes will be paired with the successful request and rejected. Finally the single admissible class will be forwarded to the RSVP module, and each of the others will block as a singleton in the next iteration of the algorithm.

Also, the additive algorithm will not always minimize the number of LSPs established. Consider a network with 8 classes of service where there is one path with a capacity large enough for either classes 0 and 1 combined, or classes 2 through 7 combined. The additive method of bundling will combine classes 0 and 1 and take the available high capacity link, where it may be preferable for classes 2 through 7 to be bundled together and admitted instead. If there are other links with little availability, the number of LSPs may be higher than is strictly necessary. This situation is admittedly rare and would depend also on the relative importance of classes of service. The issue could be rectified somewhat by reversing or otherwise permuting the order of connection request groups supplied to the algorithm.

The next algorithm is a hybrid of additive and subtractive. Exploiting the principle of temporal locality, it retains the configuration of the last successfully admitted bundle into the network and attempts first to compute a path for a bundle with those same contents (or as close as possible). In the following algorithm we assume there is a boolean array `lastAdmitted` available which holds the value true at index i if that class-of-service was a part of the last successfully admitted bundle and false otherwise.

Additionally, the function `doSubtractive()` is modified to update the contents of `lastAdmitted` to match the largest recent grouping accepted into the network.

```
Algorithm doHybrid( TrafficRequest[] TR )
begin
     c ← sizeOf( TR )
     j ← 0
     TrafficRequest[] tempTR

     for i ← 0 to c do
     begin
          if lastAdmitted[TR[i]] equals true then
               tempTR[j] ← TR[i]
               TR[i] ← NULL
               j ← j+1
     end

     doSubtractive( tempTR )
     doAdditive( TR )
end
```

The essential idea presented is that of dealing subtractively with a group of connection requests which contain the same classes of service as were last successfully admitted as a group into the network, and additively processing those which remain. The goal of doing so is to have the largest possible bundles processed with as few path computation invocations as possible. It is more likely that those connection requests which form part of the last admitted bundle will be accepted immediately or with very little difference, hence the subtractive algorithm is a good choice. The classes of service which were inadmissible in the prior bundling attempt are processed additively, as it is presumed to be unlikely they will have become acceptable since then. In practice the algorithm above as it appears in the simulation program was written to use the modified additive and subtractive algorithms which include a preprocessing step, as described below.

If we disregard for now the additional overhead of path computation, it can be seen from the above algorithms that the additive and subtractive methods have a run time complexity of $O(n^2)$. This is acceptable because scenarios in which the algorithm will

consistently run in the worst case are very rare. When there is sufficient bandwidth for most connection requests the algorithms are more closely linear and can be as small as constant time, for example in the case of the subtractive algorithm where all classes of connection requests are admissible in the network. Situations in which quadratic performance would occur are limited to networks in which the initial bandwidth model and classes have been provisioned very poorly. Such a case might occur if a single class-of-service received most or all available link bandwidth, causing the other classes to quickly reach capacity and starve for resources.

With this in mind, an additional preprocessing step has been implemented as a useful variation to modify the additive and subtractive algorithms. This preprocessing step improves upon bundling success with the drawback of an initial increase in calculations. The benefit of this overhead relies on the underlying network scenario, and would only be part of a prudent strategy designed for a specific network and division of classes of service. At the beginning of each call to `doAdditive()` or `doSubtractive()`, the preprocessing algorithm starts first by performing an "admission scan" of each traffic request in the group supplied to the algorithm, calling `computePath()` for each individually. If any of the connection requests block, they will not be admissible within any bundling scheme and are immediately rejected. Those requests which remain are then known to be admissible in the network, and are subject to the remainder of the existing algorithms to determine bundle size and contents. This is designed to provide significant improvements to both algorithms if there are consistently blocking classes of service intermixed with others which are admissible.

The deficiency in subtractive bundling whereby more LSPs are established than strictly necessary is no longer possible with this preprocessing step. Unnecessary checks which might occur with the additive algorithm are also removed, such as where pair wise grouping of an admissible class-of-service with several inadmissible ones checks each unsuccessful variation before blocking requests individually. The preprocessing of the admission scan, having removed all requests which will block individually, is already prepared for successful bundling. It may still be the case that each request must be sent

through the network individually and the E-LSP method provides no improvement over the L-LSP method, however if such a situation occurs we are still better off removing blocking singletons initially. Conversely, if all connection requests are admissible, the algorithms are doing extra work which is unnecessary. It is a careful balance, and another goal of the simulation results to follow is to show in which cases it may be advisable to apply this preprocessing step.

3.5 Network Topology

OMNeT++ offers its own scripting language for defining network topologies and link capacities, NED. The INET framework extensions to OMNeT++ also include facilities for specifying connectivity of switches via routing table files parsed by the RSVPApp module. In combination these were used to create the test networks which were part of the simulation. One of the limitations of the INET framework is the zero convergence time of retrieving information from and updating into the TED. Each switch has immediate access into the database, and hence updating link capacities as paths are established and bandwidth is reserved occurs immediately. This is not a serious concern for the purpose of this work, although it is important to note that the network image as viewed from a single LSR is always consistent; such may not always be the case in actual practice, in particular when observing very large networks.

The linear network is the simplest topology examined, with a source transmitting connection requests at left bound for the destination sink at right. In all topologies the source transmits connection requests following a uniform distribution as previously mentioned. These requests are in the form of messages which reach the ingress LER (LSR1) where the RSVPApp and OSPF modules conduct admission control prior to an acceptable bundle with an explicit route object being passed to the RSVP module. It is at this stage that the PATH message is transmitted ultimately to the destination, and the RESV message makes a return trip, with the FEC bindings being installed at each intermediate switch.

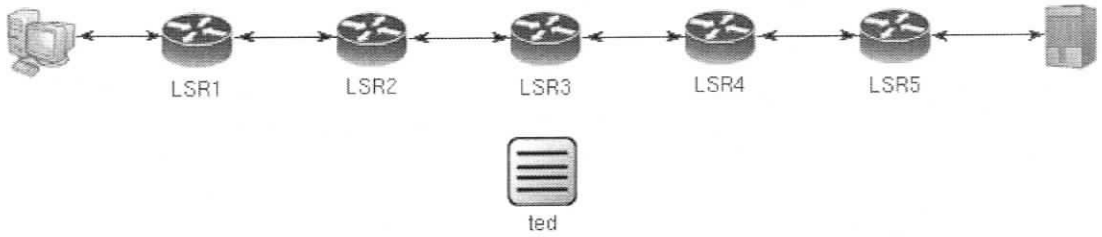


Figure 3.6 : Linear network topology

The "fish" network was chosen as another typical test network topology which features branching from LSR1 and also a bottleneck link between LSR4 and LSR5.

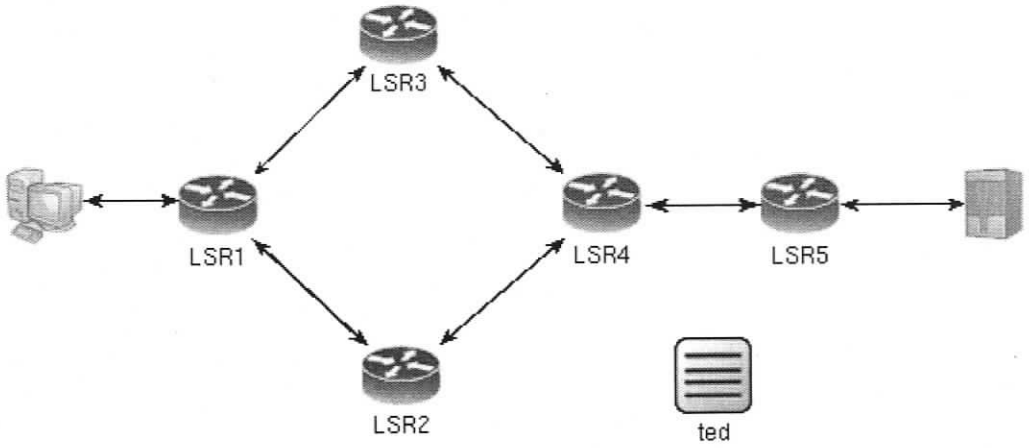


Figure 3.7 : Fish network topology

The bifurcated and mesh topologies were selected due to their multiple paths from the source to destination. The total number of switches and links in the networks are relatively small due in part to routing limitations within the INET framework. Namely, generating random topologies with very large amounts of nodes and connectivity between them using the NED language is relatively straightforward. However, the need to specify a routing table file for each switch, including connectivity and preferred paths for every link, without causing inconsistencies is an extremely arduous task the way

RSVP-TE has been implemented in the framework. In slightly larger trial runs the added complexity did not pay off in terms of interesting new results. The bundling methods discussed are primarily applied on these modest topologies in order to show functionality and behavioural characteristics which scale to larger networks.

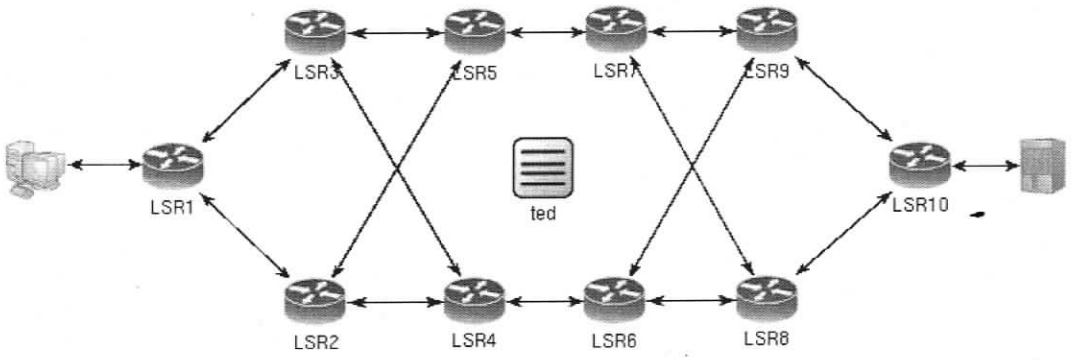


Figure 3.8 : Bifurcated network topology

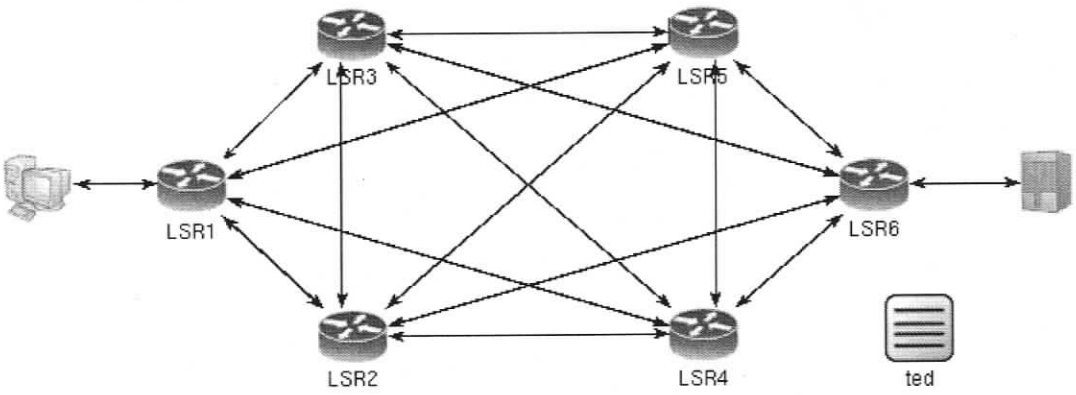


Figure 3.9 : Mesh network topology

3.6 Path Computation

The OSPF module was modified for the simulation to take into consideration a cost associated with each feasible path. By including cost calculation based on the metrics existing at each link along a given path, it is possible to make it far less likely that one link will become overburdened compared to the others, and this prevents the network as a whole from becoming imbalanced.

Dijkstra's shortest path algorithm is used to determine suitable candidates for a path from a given source to destination. Where there exist traffic constraints such as bandwidth and delay, then the network links are subject to pruning one constraint at a time until only those links which satisfy the given constraints remain. A constrained shortest path tree is then constructed. The cost function used to evaluate feasible paths can be generalized as $A*metric1 + B*metric2 + \ldots + Z*metric3$ for each link, where A through Z are administrative weights configured in advance. The final value of the cost function for each path is the sum of the costs of all intermediate links.

For the purposes of the simulation the following cost function derived from [23] is used:

$$Link\ Cost = A*1/ABW(kbits) + B*delay(ms) + C*hopcount$$

The administrative weights are applied as $A = 1000$, $B = 0$, $C = 1$. This is a straightforward approach which negates the necessity for delay inclusion in cost calculation, places the greatest emphasis on the number of hops, and also balances traffic across multiple paths of equal length as reservations are made. The result is that each of the network topologies examined is well balanced throughout the simulation, though such a cost function always elects to prefer paths which are shorter, regardless of bandwidth availability.

3.7 Random Number Generation

Random numbers in simulation are never random. Rather, they are produced using deterministic algorithms. Algorithms take an initial seed value and perform some deterministic calculations on them to produce a "random" number and the next seed. Such algorithms and their implementations are called pseudo-random number generators (RNGs). Starting from the same seed, RNGs always produce the same sequence of random numbers. This is a useful property because it makes simulation runs repeatable with the same results. To avoid unwanted correlation, it is also important that different simulation runs use non-overlappting series of random numbers, so the generators should be started with seeds well apart. The random number generator implemented within OMNeT++ is a linear congruential generator (LCG) which produces uniformly distributed integers in the range 1 to 2^{32} - 2. For selecting good seeds the seedtool program supplied with OMNeT++ was used, with the recommended distance of ten million between them. The RNGs produce series of pseudo-random numbers which are used to produce random variates that correspond to specific distributions. The distributions used (uniform, normal) were applied to the arrival times of connection requests and the variability in bandwidth per request, respectively. For each simulation conducted with a specific set of connection request parameters, bandwidth model, and LSP method, ten simulation runs were completed and their results averaged.

Chapter 4

Simulation Results

For each network topology examined below, three scenarios were used with different connection parameters. For a given set of parameters, runs were conducted for the L-LSP method, E-LSP bundling methods, and each of the bandwidth sharing models. Additionally, each of the above was simulated with five different available bandwidths per link. In the case of the L-LSP method, the average of all simulation runs is shown. The E-LSP method is a special case; as there are five bundling algorithms, consideration of the characteristics of this method in the simulation has been taken to reduce the total amount of graphs to a reasonable size. Namely, the E-LSP method algorithms all produce close to the same number of LSPs, network utilization, and blocked/admitted connections in the network with very little variation. They each accomplish the same result but differ primarily in the number of path computations the OSPF module must conduct to achieve their result. Consequently, when E-LSP information is graphed it is the average of all five algorithms across all simulation runs. The exception to this is path computations, wherein graphs shown display each algorithm separately as an average of its respective results, for the purpose of comparing them to one another as well as the L-LSP method.

The bandwidth sharing models (Capped, Shared, RDM) have very little impact on the simulation results. Due to the nature of the arrival of connection requests in the simulation, bandwidth is fairly consumed by all classes of service until there is little remaining. Thereafter the highest priority classes reserve all of the remaining bandwidth in few LSPs when employing the Shared and RDM. This result is overall a handful fewer LSPs but slightly greater numbers of blocking connections of lower priority. In instances where there is sufficient bandwidth for all connections, the bandwidth model does not come into play. As such the graphs in this chapter show only the Capped model, which is well suited to the E-LSP methods.

The criteria considered in this chapter are:

- *Number of L-LSPs or E-LSPs established*: The total number of label switched paths created and for which resources have been reserved in the network over the course of the simulation. Recall that the total number of connection requests arriving at the LER is two thousand for every simulation run. L-LSPs each contain a single connection, that is, no bundling of traffic occurs. E-LSPs may contain a single connection up to the maximum number of classes of service available.

- *Network utilization*: When the simulation completes, statistics for reserved and unreserved bandwidth are gathered from each link. The utilization of a single link is the fraction of bandwidth reserved over total initial bandwidth. The network utilization is then calculated as the average of all individual link utilizations. It is important to make the distinction that utilization less than 1 does not always indicate there is space for additional traffic. Resource availability depends not only on link utilization but also topology; for example, the fish network has a bottleneck link between LSR4 and LSR5 which reaches capacity when the network utilization is approximately 0.6. Network utilization is thus employed as a good indicator of used/unused network capacity as link bandwidths are adjusted, but only insofar as they are considered as directly coupled with the topology being examined.

- *Blocked connections*: Connection requests which arrive at the LER but are not admissible in the network and are rejected by the RSVPApp module in conjunction with no feasible path determined by the OSPF module. The component numbers of blocked connections per class are shown as a stacked bar which signifies the total number of connection requests which have blocked. This is also a good indicator of the extent to which all available network resources have been reserved. The test sets across the various topologies were selected in such a way that a range of blocking occurs, from extreme to relatively little, to illustrate the behaviour of each algorithm under various conditions.

- *Path computations*: A single path computation is an invocation of the OSPF module to attempt to compute a path from source to destination for a given connection request. This is the most interesting case for the E-LSP method and the least interesting for the L-LSP method. As the L-LSP method is simply no change from how MPLS currently operates, each of the two thousand connections will be processed individually and the number of path computations will always be two thousand.

4.1 Simulation Parameters

Selecting "suitable" parameters for all simulation runs is challenging because specific results are desired without a complete loss of generality. As with almost any simulation, further tests could always yield a clearer view of the entire picture. However, the number of combinations of connection request sets, bandwidth distributions across links, and network sizes is vast to the point of being almost limitless. Further simulation scenarios beyond those outlined here remain as future work. The values used for bandwidth, connection requests, class types, and variations thereof across scenarios and topologies were applied with the intention of showing sample sets of results which give an indication of general behaviour, while not sacrificing specific details which are prevalent upon close examination.

The parameters which follow were applied to every topology, at each of the link bandwidths shown: 12000, 24000, 48000, 72000, and 96000 kilobits. Each of the three sets of parameters form a single scenario. Scenario 1 is the initial parameter set. Scenario 2 doubles the requested bandwidth but otherwise parameters are unchanged. Scenario 3 uses the maximum of 8 classes of service, with adjusted bandwidth partitioning and requested values. For the purpose of comparing results across scenarios, the number of connection requests initiated is source limited to 2000.

Table 4.1 : Parameters for simulation scenario 1

Parameter	Value
Classes of Service	4
Priority of Classes (High to Low)	0 to 3
Mean Bandwidth Requested per Class (High to Low)	128, 64, 32, 16 kbps
Variation in Bandwidth Requested	Normal distribution with standard deviation 0.15, 0.3
Bandwidth Share Model	Capped
Bandwith Partitioning per Class (High to Low)	0.533, 0.267, 0.133, 0.067
Total Connection Requests	2000

Table 4.2 : Parameters for simulation scenario 2

Parameter	Value
Classes of Service	4
Priority of Classes (High to Low)	0 to 3
Mean Bandwidth Requested per Class (High to Low)	256, 128, 64, 32 kbps
Variation in Bandwidth Requested	Normal distribution with standard deviation 0.15, 0.3
Bandwidth Share Model	Capped
Bandwith Partitioning per Class (High to Low)	0.533, 0.267, 0.133, 0.067
Total Connection Requests	2000

Table 4.3 : Parameters for simulation scenario 3

Parameter	Value
Classes of Service	8
Priority of Classes (High to Low)	0 to 7
Mean Bandwidth Requested per Class (High to Low)	512, 384, 256, 128, 96, 64, 32, 16 kbps
Variation in Bandwidth Requested	Normal distribution with standard deviation 0.15, 0.3
Bandwidth Share Model	Capped
Bandwith Partitioning per Class (High to Low)	0.344, 0.258, 0.172, 0.086, 0.065, 0.043, 0.022, 0.01
Total Connection Requests	2000

The key differences between each scenario are: the number of classes being differentiated in the network, the amount of bandwidth each connection of a given class type is requesting, and the consequent partitioning of bandwidth (Tables 4.1, 4.2 and 4.3).

4.2 Linear and Fish Network Topologies

The linear and fish topologies show very similar results and have thus been combined. This occurs because the fish topology spreads traffic evenly over both possible routes due to the link cost calculation taking account of available bandwidth. However, all traffic must eventually take the bottleneck link, which is a feature of the linear network. The difference between the two topologies occurs in terms of network utilization. Where the linear network achieves almost 100% utilization, the fish network achieves 100% capacity only on the bottleneck link between LSR4 and LSR5. The

remaining four links are reserved to half capacity, ie. 50% utilization * 4 links + 100% utilization * 1 link = 3 / 5 = 0.6 or 60% total network utilization.

4.2.1 Linear/Fish Topologies, L-LSP Method, Scenario 1

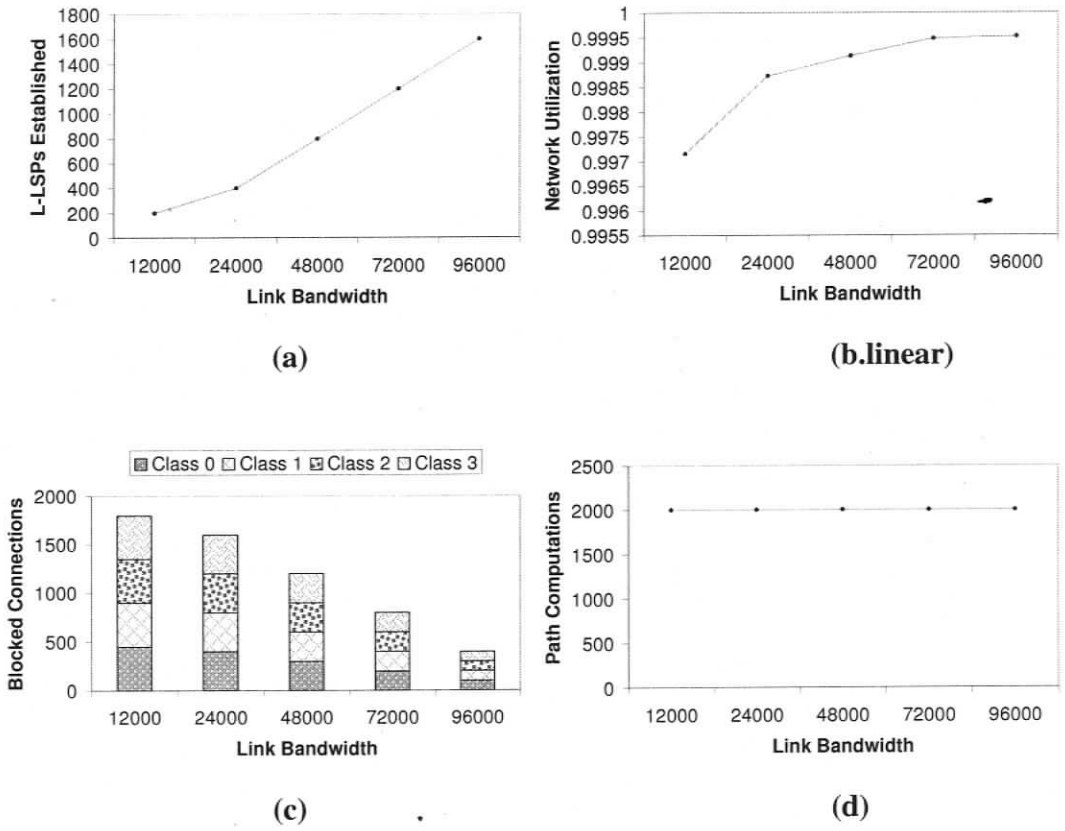
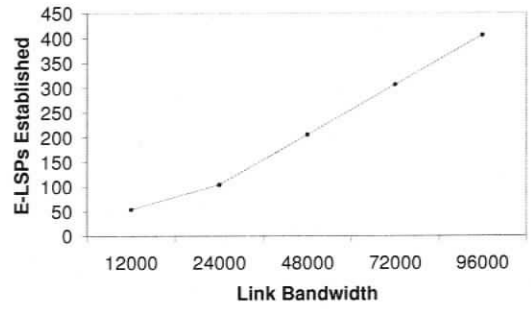


Figure 4.1 : Linear/Fish topologies, L-LSP method, simulation scenario 1

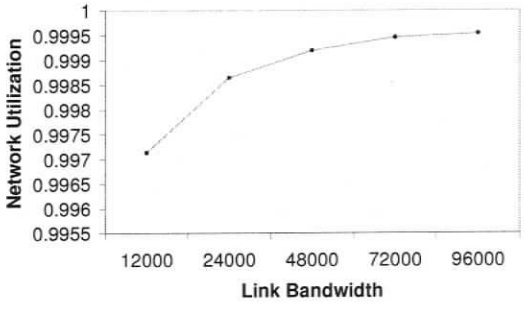
The number of L-LSPs established increases linearly with the available bandwidth per link, and the number of blocked connections decreases. As the class-of-service of connection requests cycle from 0 to 3, each class is fairly represented in blocked connections. Figure 4.1 (d) applies to all simulation scenarios examined throughout the chapter and is the standard by which the E-LSP method algorithms will be measured. Because the L-LSP method checks each connection request individually and

two thousand connection requests are issued, then no matter whether these connection requests are admitted or rejected by admission control, there will always be precisely two thousand path computation invocations for L-LSP establishment.

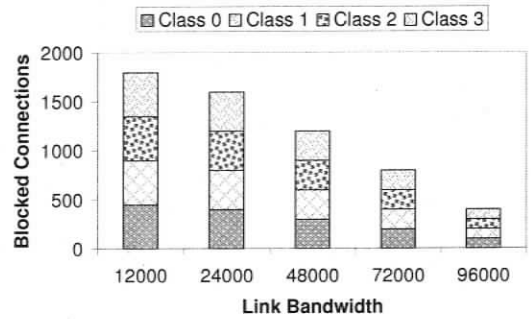
4.2.2 Linear/Fish Topologies, E-LSP Method, Scenario 1



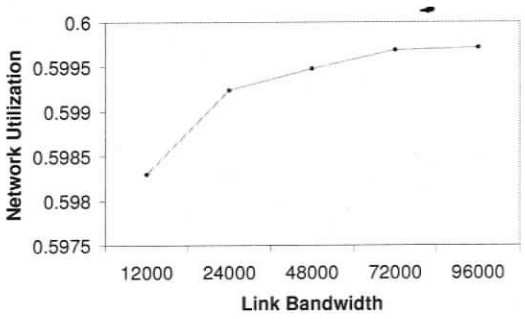
(a)



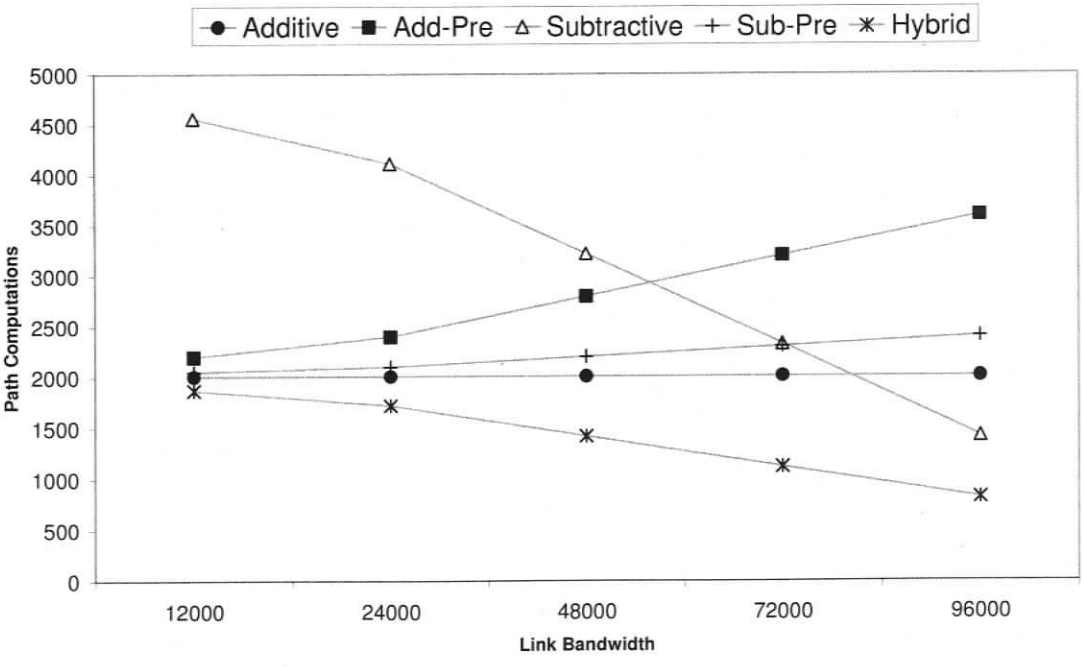
(b.linear)



(b)



(b.fish)



(d)

Figure 4.2 : Linear/Fish topologies, E-LSP method, simulation scenario 1

The number of E-LSPs established is approximately one quarter the number of paths created by the L-LSP method (Figures 4.1 (a) and 4.2 (a)). The contents of each E-LSP are nearly always maximal in terms of the number of connection requests being aggregated together, that is, the four classes of service being differentiated in this scenario. Figures 4.2 (b.linear) and (b.fish) show the network utilization of the linear and fish networks, respectively. They are nearly identical to the utilization in the L-LSP. This is consistently the case throughout scenarios and topologies. This also applies to blocked connections of figures 4.1 and 4.2 (c). Regardless of how many connection requests can be packed into a single E-LSP, if there are insufficient network resources to admit a connection using either method it must be rejected. The classes of service are also fairly represented in Figure 4.2 (c). Here connection requests are signaled in groups of four, comprised of the four classes of service. While network resources are unreserved, these groups will each form an E-LSP until such a time as network resources begin to grow scarce. E-LSPs comprised of fewer classes will shortly appear to fill in the remaining "gaps" prior to the network bandwidth being completely reserved and remaining requests being blocked.

In figure 4.2 (d) the number of path computations invoked by each of the five bundling algorithms described in section 3.4 is shown: Additive, Additive with a preprocessing step to remove singly blocking connection requests (Add-Pre), Subtractive, Subtractive with preprocessing (Sub-Pre) and the Hybrid Additive/Subtractive algorithm.

Both the linear and fish network topologies have their resources rapidly consumed, and maximum capacity is reached even at maximum link bandwidth. At the lowest point of 12000, nearly all connection requests are blocking, and at the highest point of 96000 approximately one quarter of connection requests are still being blocked. This is an important consideration because the Add-Pre and Sub-Pre algorithms with preprocessing perform best when the network resources are rapidly exhausted and many, if not most connection requests will block. As resources become more available, the added work of preprocessing begins to lose its usefulness because the majority are being admitted into the network. This is apparent with the Add-Pre algorithm, and to a lesser

extent the Sub-Pre algorithm in terms of the number of path computation invocations growing as link bandwidth increases. The primary difference between the performances of the two is the initial, large benefit of the Sub-Pre algorithm starting with large bundle sizes when network reservations are low. Many bundles containing all four classes can be sent consecutively with a single path computation each, whereas Add-Pre does the initial preprocessing work and then constructs bundles with a bottom-up approach.

The Subtractive algorithm shows a sharp decline in the number of path computations required as bandwidth increases, for the same reason that Sub-Pre has initial gains: when network capacity is plentiful, sending through consecutive bundles of maximum size with a single path computation is very efficient. As will be shown later, this algorithm performs very poorly when resources are scarce, and so careful consideration must be taken before claiming it would always be a good choice.

When bundling connection requests using the Additive algorithm, there is very little difference from admitting single connection requests via the L-LSP method. This is the case in the simulation scenarios outlined here and in any network where the bandwidth is fairly divided among classes and connection requests from each class are represented approximately equally. When there is disparity between the bandwidth allocated to each class based on the amount it requests, the classes which starve for resources will cause the Additive algorithm to perform slightly more path computations as it searches for an acceptable grouping of classes which will ultimately block no matter what. This is however rectified by the singleton connection request checks in the initial stage of Add-Pre and to a lesser extent the base case of the Additive algorithm.

Performing well in terms of path computations is the Hybrid algorithm, which consistently performs better than the L-LSP method while still reducing the number of paths established by a factor equal to or just below the number of classes in the network. The Hybrid algorithm accomplishes this by its adaptability; it has the positive attributes of the Subtractive algorithm whereby it takes very few path computations to successfully create large bundles when there are ample network resources, and shows the resilience of

the Additive algorithm to an explosion in the number of path computations when the network is congested and there exist a large number of blocking connections. While exploiting temporal locality is a significant benefit in the scenarios outlined here, such may not always be the case, for example in an environment in which incoming connection requests are very unpredictable. This is the most fundamental cause for due care, and as much foreknowledge as possible of network conditions when considering the Hybrid algorithm.

4.2.3 Linear/Fish Topologies, L-LSP Method, Scenario 2

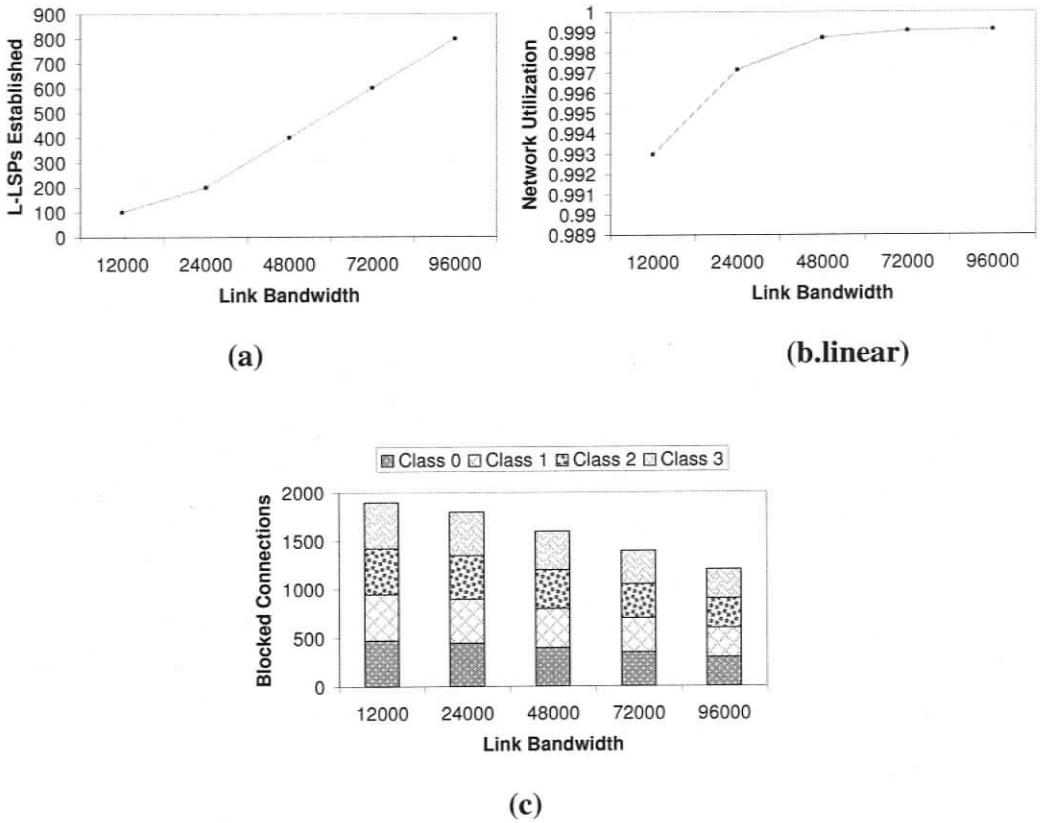
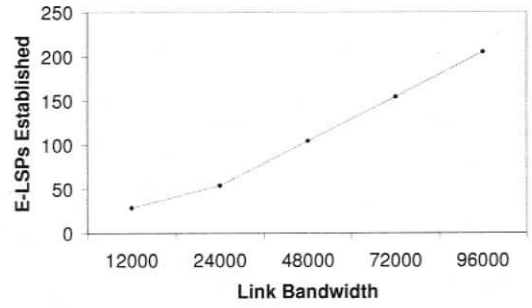
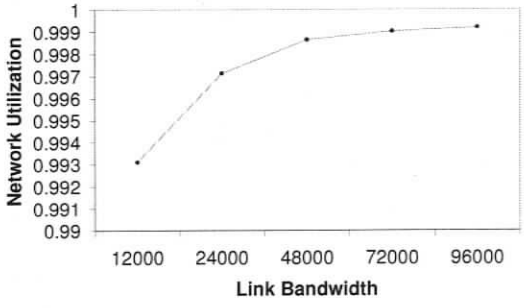


Figure 4.3 : Linear/Fish topologies, L-LSP method, simulation scenario 2

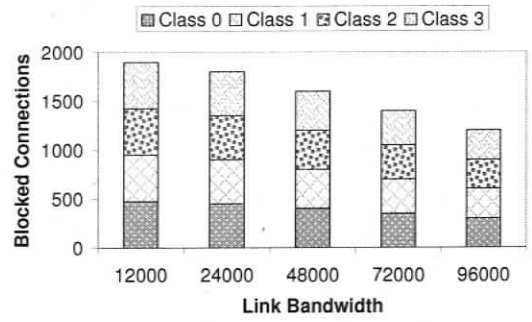
4.2.4 Linear/Fish Topologies, E-LSP Method, Scenario 2



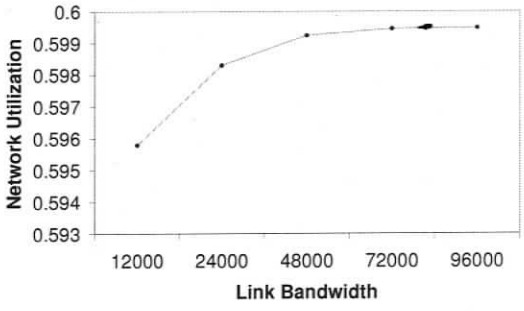
(a)



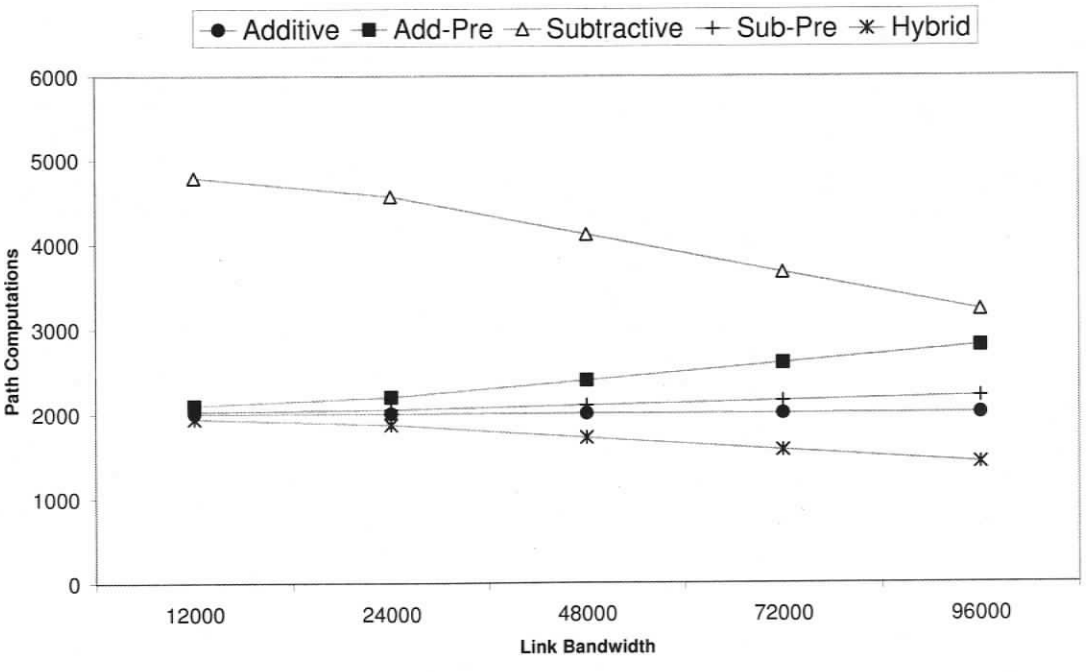
(b.linear)



(c)



(b.fish)



(d)

Figure 4.4 : Linear/Fish topologies, E-LSP method, simulation scenario 2

As scenario 2 has connection requests doubling the amount of bandwidth requested in each class-of-service, the number of explicit paths established in both label and exp-inferred LSP methods of figures 4.3 (a) and 4.4 (a) are half those of scenario 1, as expected.

Figure 4.4 (d) shows a slower increase in the number of path computations for the Add-Pre and Sub-Pre algorithms as link bandwidth increases, compared to figure 4.2 (d). This is due to the fact that the extra preprocessing pays off to a much greater degree under high network congestion conditions, which here are even more intense than previously.

Also shown is a more gradual decline in path computations for the Subtractive and Hybrid algorithms than scenario 1, because available resources are consumed more quickly by the greater requirements of the connection requests, making the highly efficient, high yield maximal bundle sizes with a single path computation less frequent.

4.2.5 Linear/Fish Topologies, L-LSP Method, Scenario 3

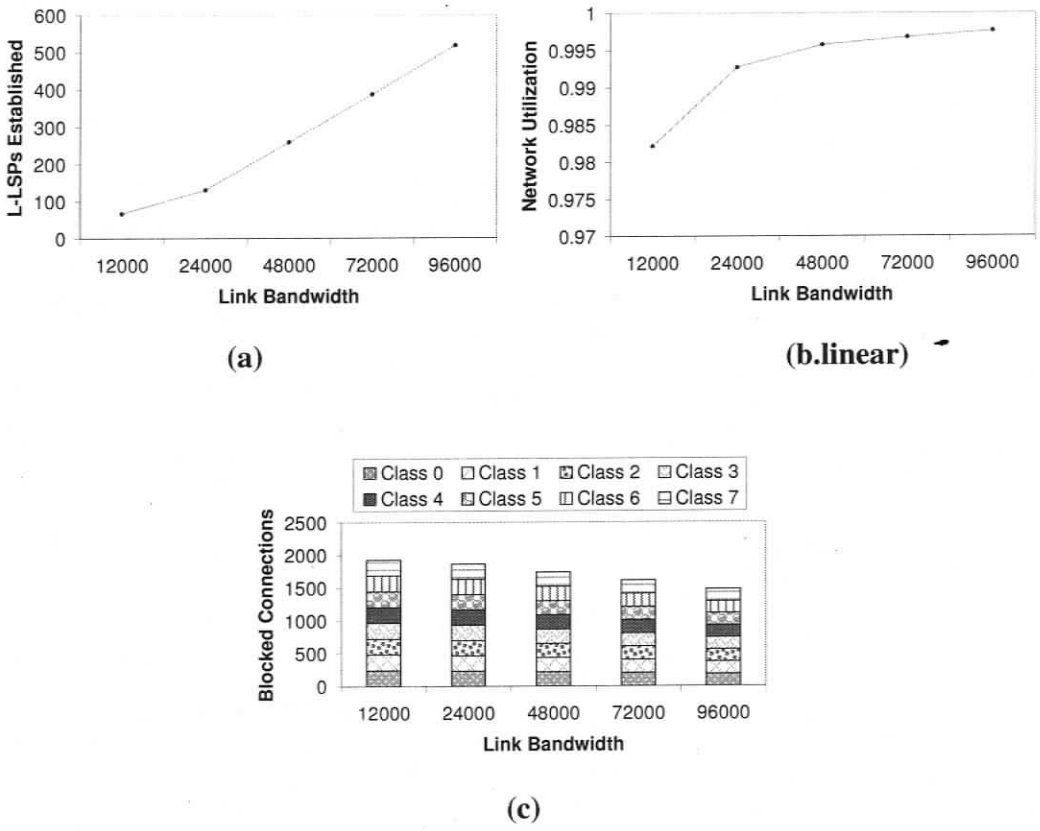


Figure 4.5 : Linear/Fish topologies, L-LSP method, simulation scenario 3

4.2.6 Linear/Fish Topologies, E-LSP Method, Scenario 3

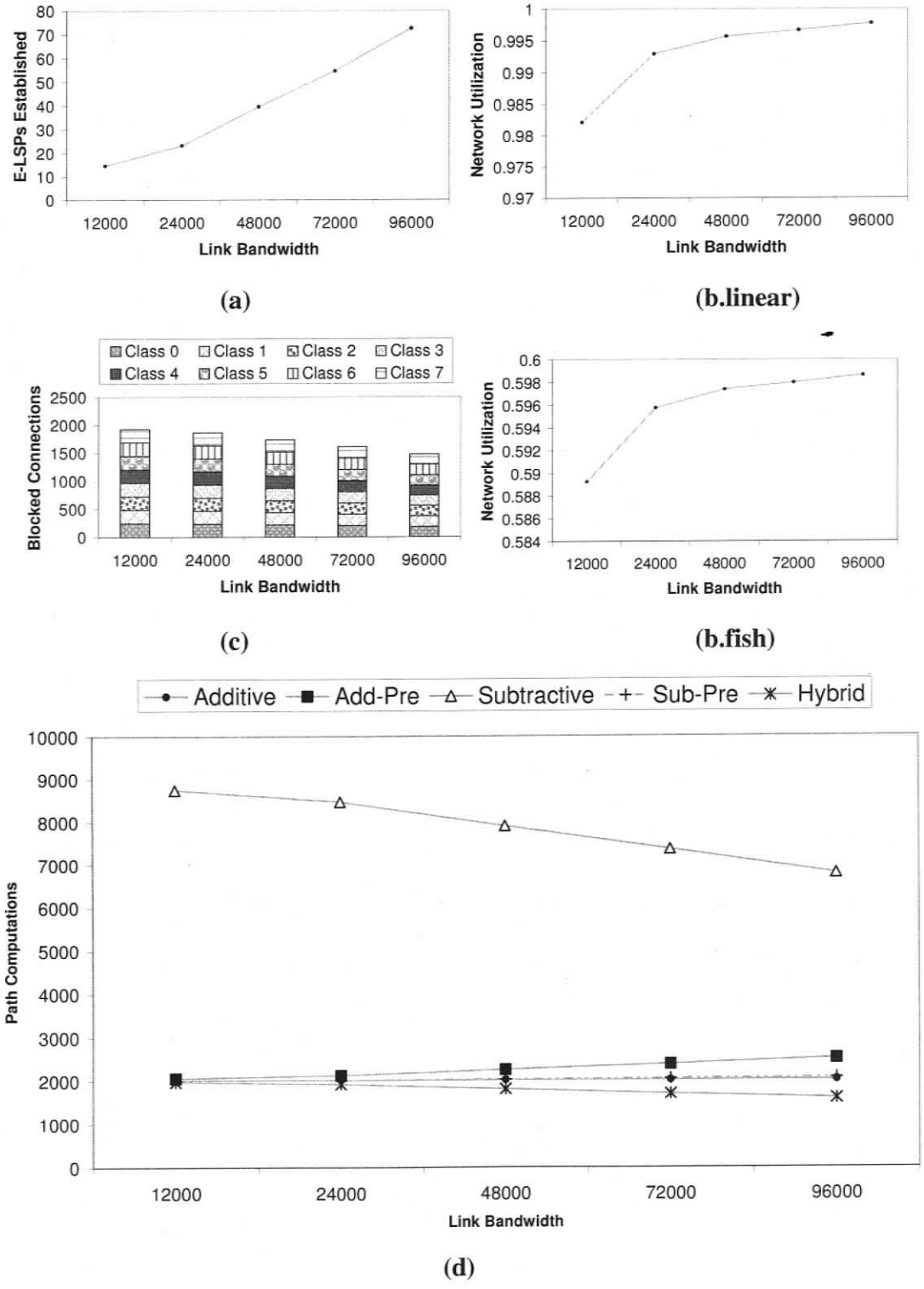


Figure 4.6 : Linear/Fish topologies, E-LSP method, simulation scenario 3

The addition of four more classes of service in scenario 3 does not affect the general results found in scenarios 1 and 2, with the exception that the number of LSPs established when bundling is now reduced to a greater degree when comparing figures 4.5 (a) and 4.6 (a). The amount of bandwidth partitioned for each class is still fair with respect to the amount of bandwidth each class requests, and so they are all approximately equally represented in the amount of blocked connections in both the L-LSP and E-LSP cases of figures 4.5 (c) and 4.6 (c).

It is in this scenario that the amount of total bandwidth requested of the network is the greatest. Complete utilization of network resources occurs very quickly, leaving the vast majority of connection requests to be rejected. For this reason the primary drawback of the Subtractive algorithm is the most pronounced in figure 4.6 (d), with the number of path computations in the range of 7000 to 9000. Fortunately it is unlikely a network would be so under provisioned or flooded with such a large quantity of incoming requests; nonetheless it is an important factor to consider, contrasted below by the bifurcated and mesh networks which are not so heavily reserved, showing excellent performance for the Subtractive algorithm at larger link bandwidths. The algorithms Add-Pre and Sub-Pre perform as close to optimal as can be achieved here, while the Hybrid algorithm still performs well, but suffers slightly from the same drawback as the Subtractive algorithm, if only to a small extent prior to the last successful bundle becoming an empty one, when nothing more can be admitted. It is at this point that the Hybrid algorithm switches solely to the Additive method and inherits its desirable qualities.

4.3 Bifurcated Network Topology

Unlike the linear and fish topologies, the bifurcated network does not reach maximum capacity at link bandwidths greater than 48000 in scenario 1. All two thousand connection requests are accepted, as traffic is balanced evenly across both paths which can be established via LSR4 to LSR6 and LSR5 to LSR7.

4.3.1 Bifurcated Topology, L-LSP Method, Scenario 1

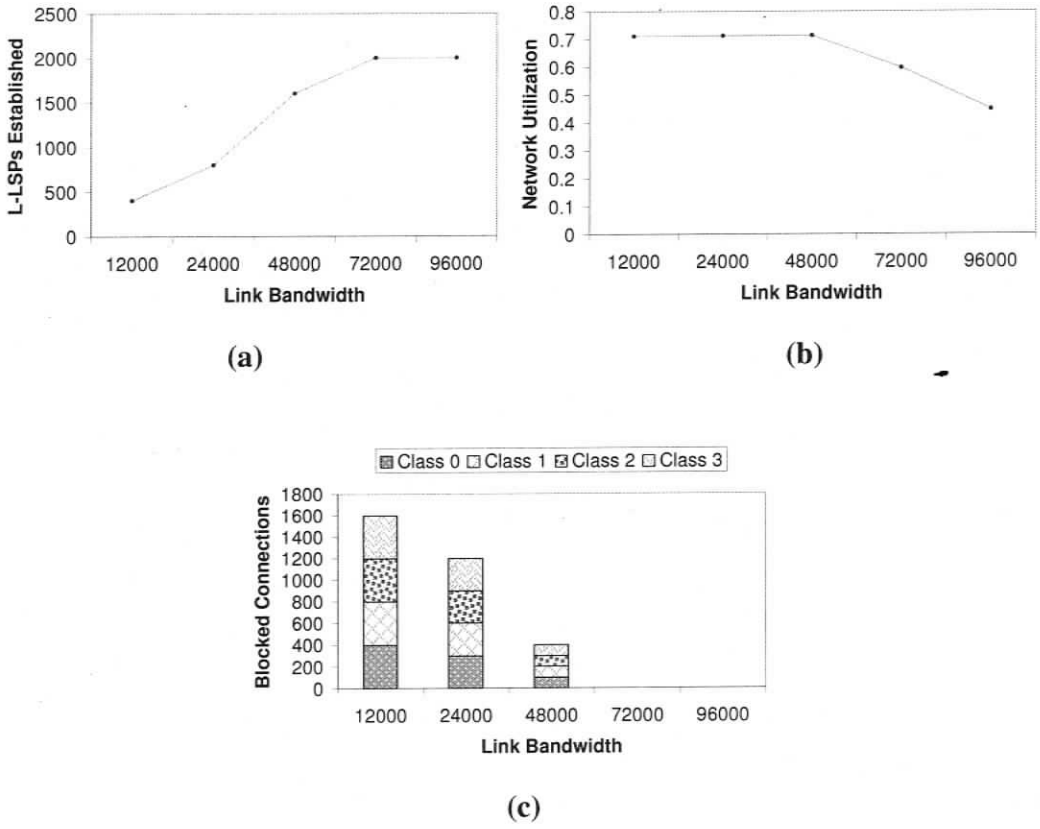
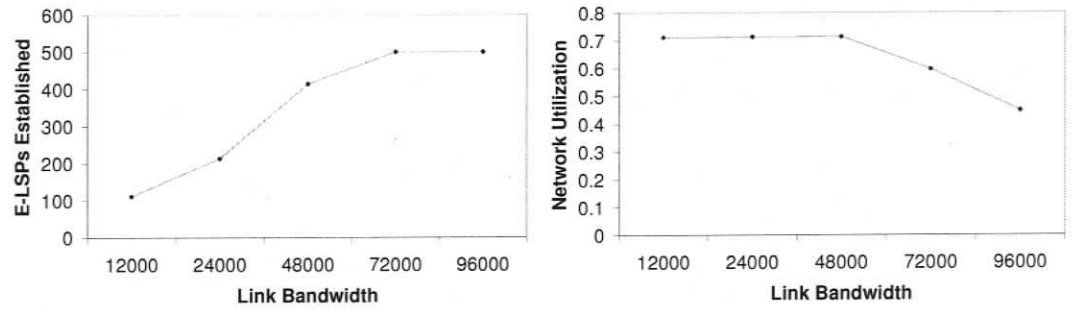
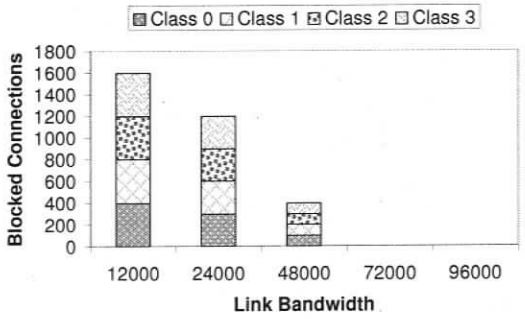


Figure 4.7 : Bifurcated topology, L-LSP method, simulation scenario 1

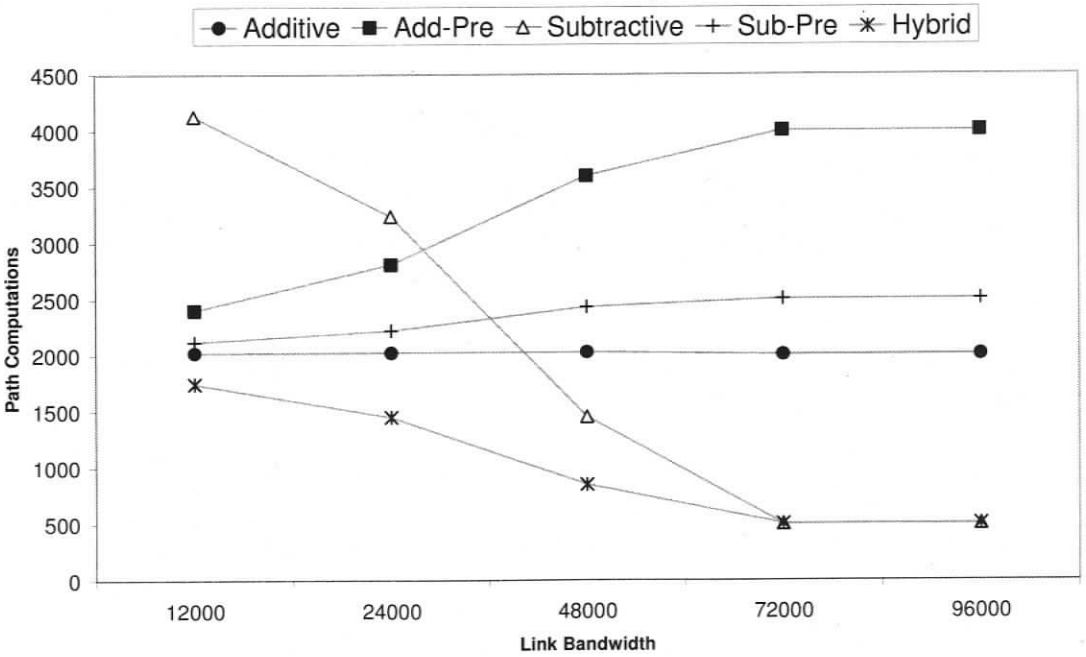
4.3.2 Bifurcated Topology, E-LSP Method, Scenario 1



(a) (b)



(c)



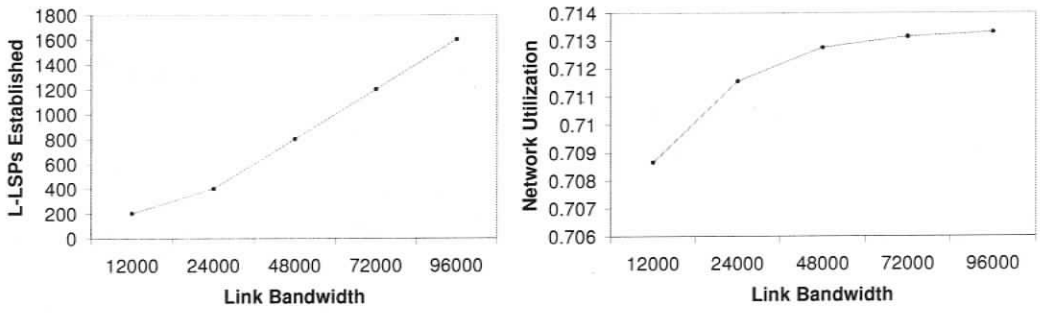
(d)

Figure 4.8 : Bifurcated topology, E-LSP method, simulation scenario 1

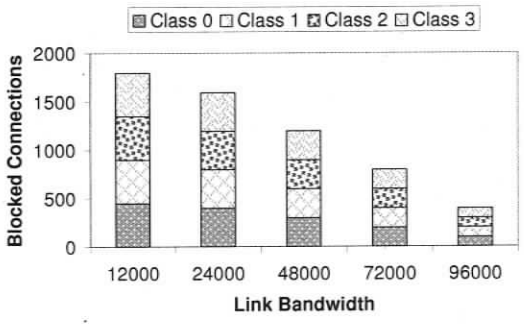
Comparing figures 4.7 and 4.8 shows us that the findings from the previous scenarios in section 4.1 hold. With the extra bandwidth available when link bandwidths are assigned 72000 and 96000 each, the Subtractive algorithm performs very well, with a sharp decrease in path computations as these additional resources become available. The Sub-Pre, and Add-Pre algorithm in particular show an increased amount of path computation calls as available bandwidth increases. This supports what was discussed earlier. The Hybrid method performs extremely well, again as we might expect with plentiful resources.

It is important to note that the decrease in network utilization which occurs above in figure 4.8 (b) at link bandwidth greater than 48000 is not due to blocking connections. Instead what occurs is the opposite; all connection requests are being accepted into the network, and increases in bandwidth after this point are not necessary in terms of accommodating additional traffic. These results are nonetheless still significant, as they are further indicators of how the bundling algorithms perform when there is more than sufficient bandwidth available in the network for all connections.

4.3.3 Bifurcated Topology, L-LSP Method, Scenario 2



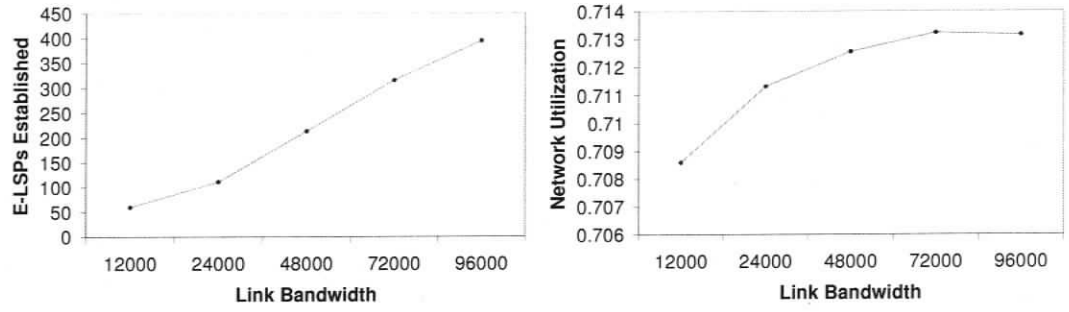
(a) (b)



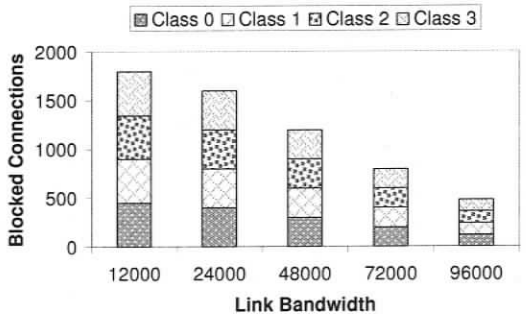
(c)

Figure 4.9 : Bifurcated topology, L-LSP method, simulation scenario 2

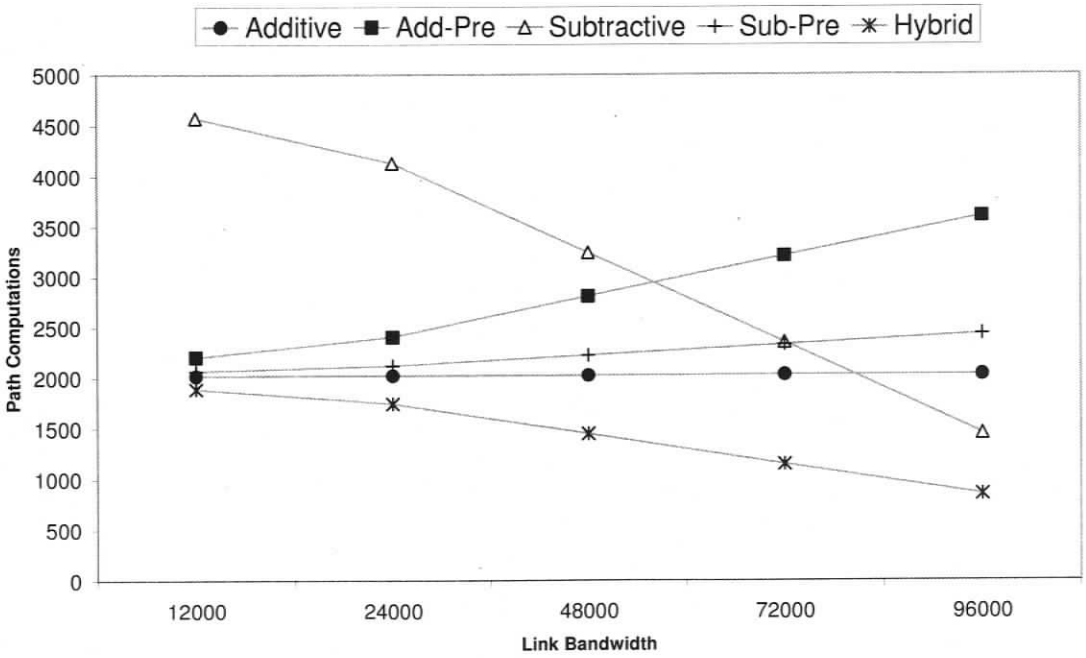
4.3.4 Bifurcated Topology, E-LSP Method, Scenario 2



(a) (b)



(c)



(d)

Figure 4.10 : Bifurcated topology, E-LSP method, simulation scenario 2

Doubling the amount of bandwidth requested by each connection results in a situation more closely resembling that of section 4.1, where some connections will always be rejected, even when link bandwidth is set at 96000. A comparison of figures 4.2 (d) and 4.10 (d) exemplifies the similarities which exist between these scenarios.

4.3.5 Bifurcated Topology, L-LSP Method, Scenario 3

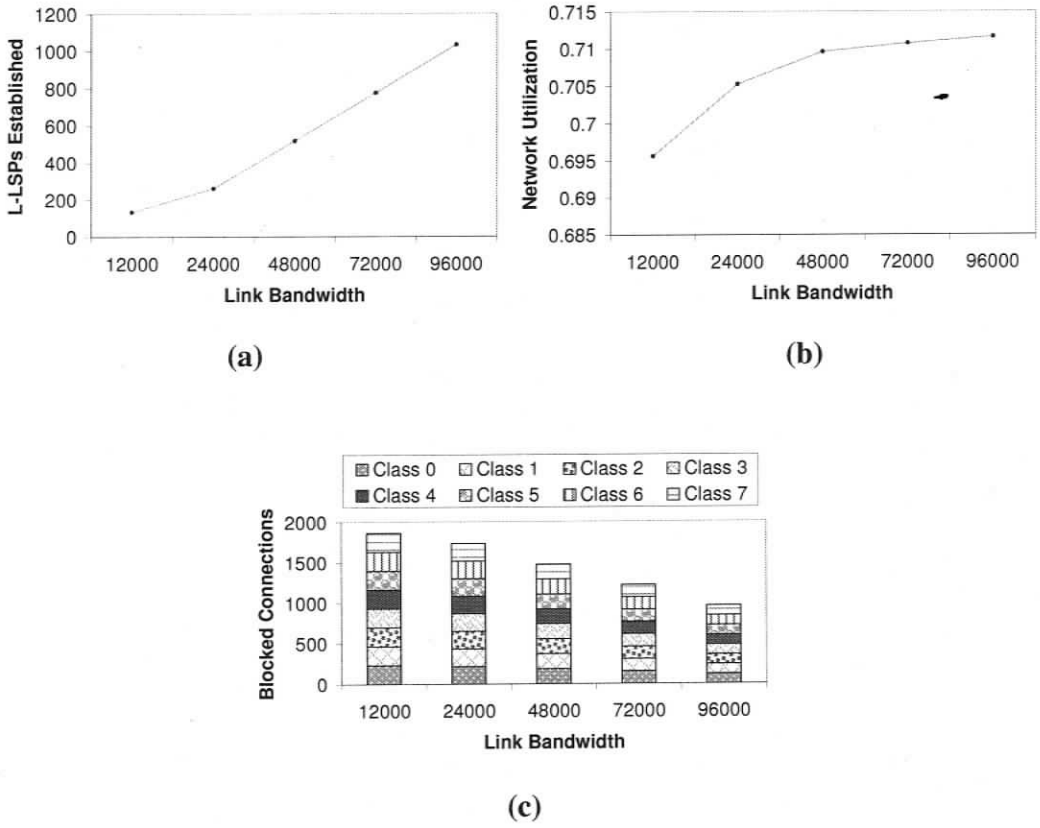
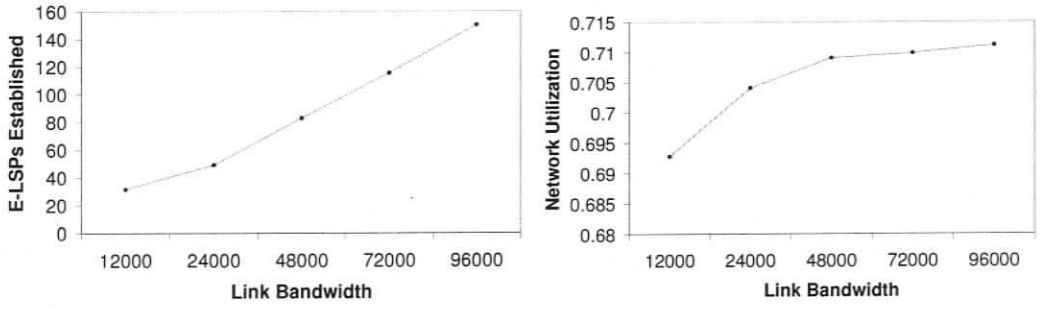
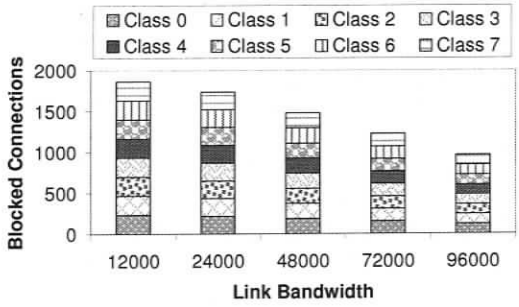


Figure 4.11 : Bifurcated topology, L-LSP method, simulation scenario 3

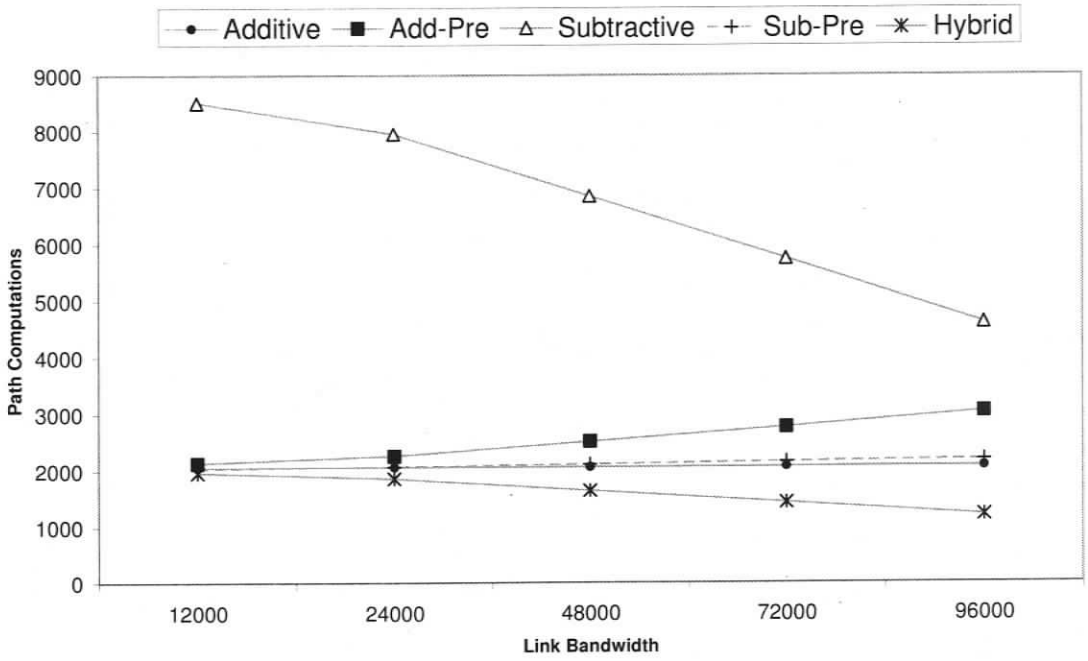
4.3.6 Bifurcated Topology, E-LSP Method, Scenario 3



(a) (b)



(c)



(d)

Figure 4.12 : Bifurcated topology, E-LSP method, simulation scenario 3

There exists a correspondence between figures 4.4 (d) and 4.12 (d). As with the linear and fish topologies, doubling the number of classes also increases the total amount of bandwidth being requested of the network. The number of LSPs, path computations, and blocked connections remain conformant to previously established trends.

4.4 Mesh Network Topology

The mesh network has many equal paths from source to destination, yielding a lower overall network utilization than the previous topologies. Nevertheless it can be seen through the progression of scenarios 1 to 3 that as the total amount of bandwidth being requested of the network continues to increase, indicators again appear to confirm what has been shown in sections 4.1 and 4.2. As mentioned for the bifurcated network topology, here a sharp decline in network utilization is due to all of the 2000 connection requests being accepted into the network at link bandwidth values greater than 24000, 48000, and 72000 in each of the three scenarios respectively.

4.4.1 Mesh Topology, L-LSP Method, Scenario 1

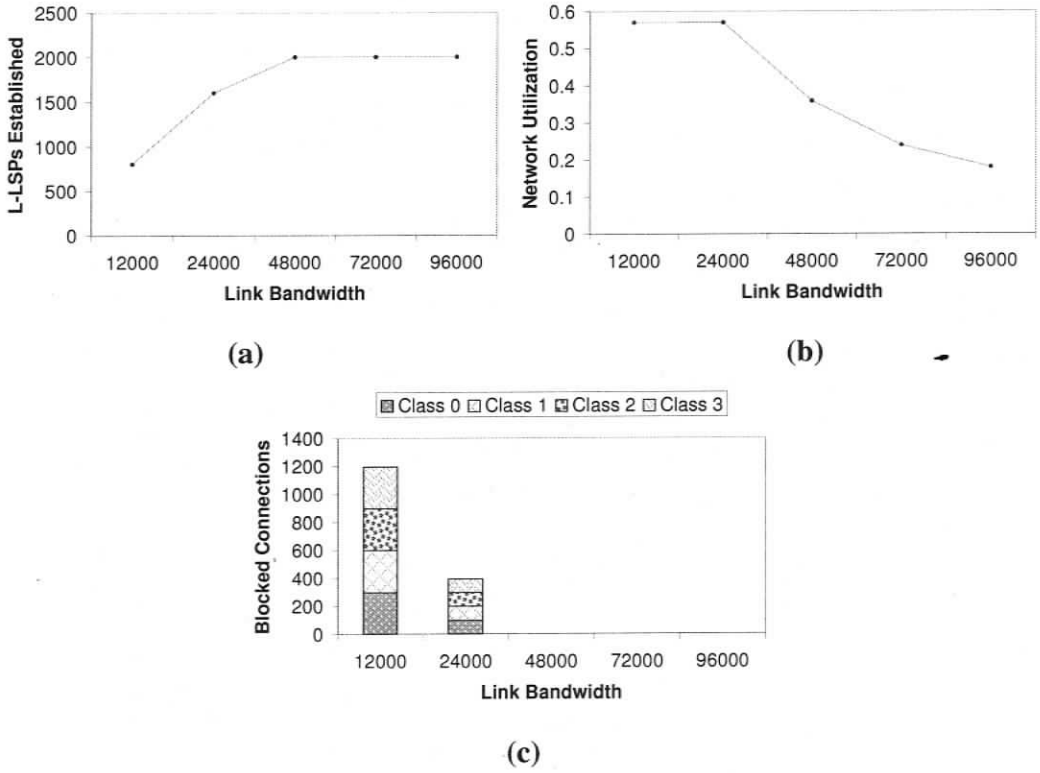
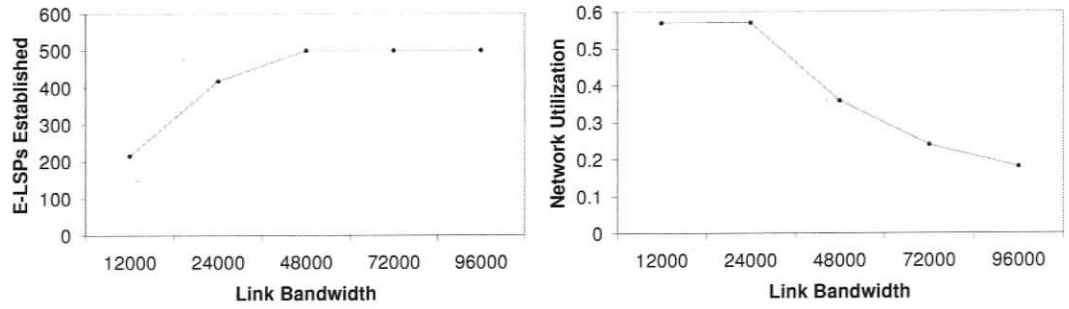
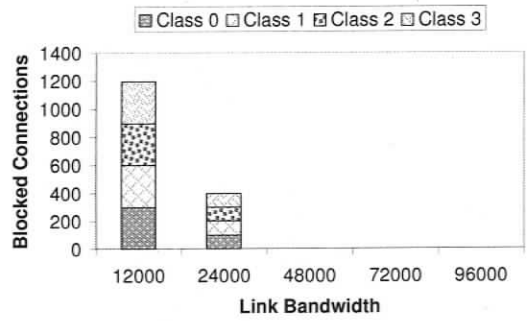


Figure 4.13 : Mesh topology, L-LSP method, simulation scenario 1

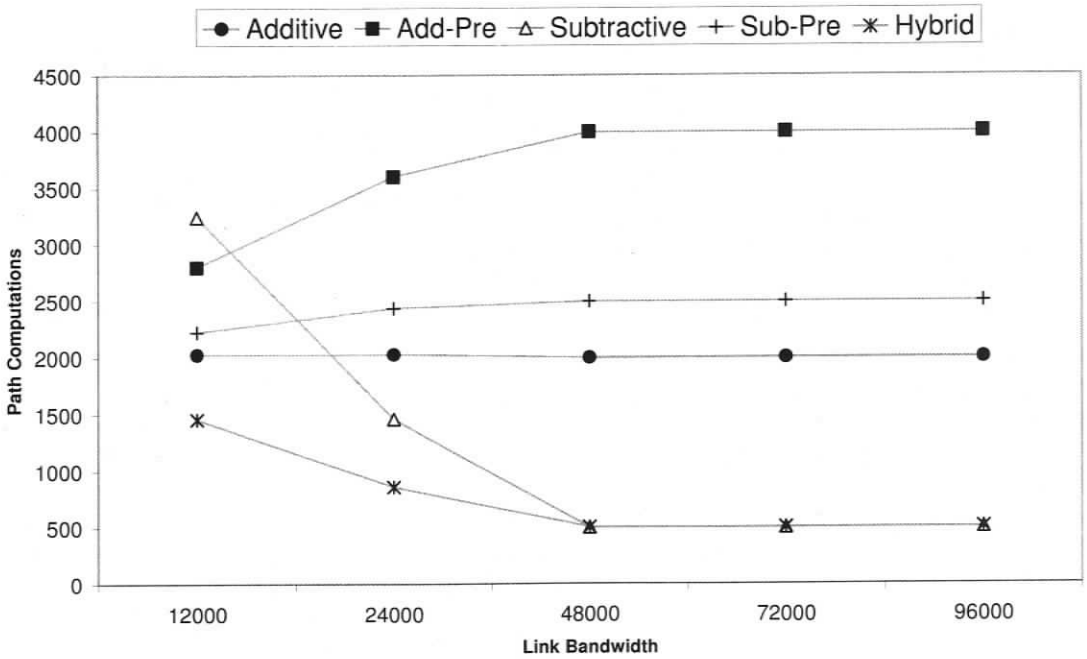
4.4.2 Mesh Topology, E-LSP Method, Scenario 1



(a) (b)



(c)



(d)

Figure 4.14 : Mesh topology, E-LSP method, simulation scenario 1

4.4.3 Mesh Topology, L-LSP Method, Scenario 2

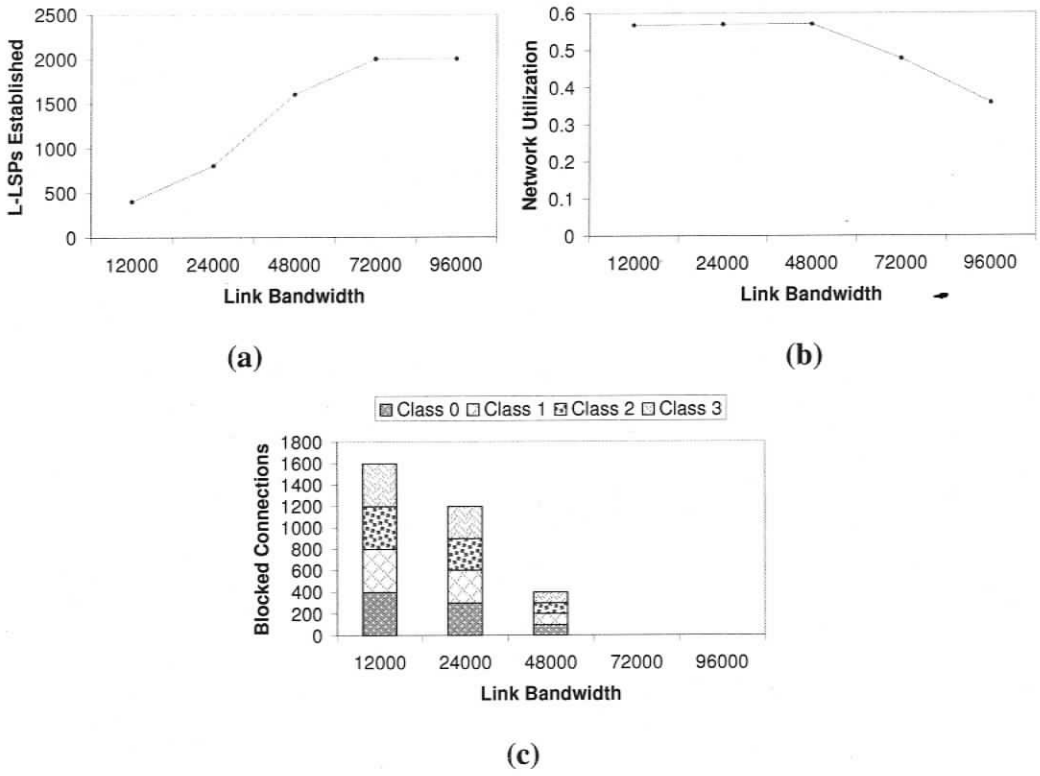
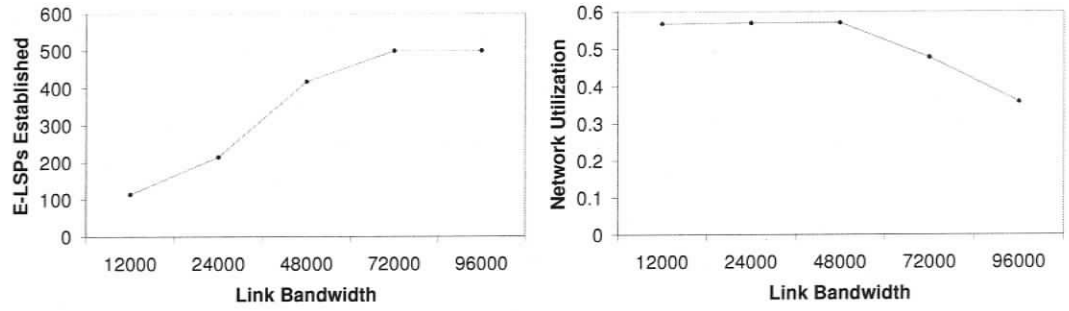
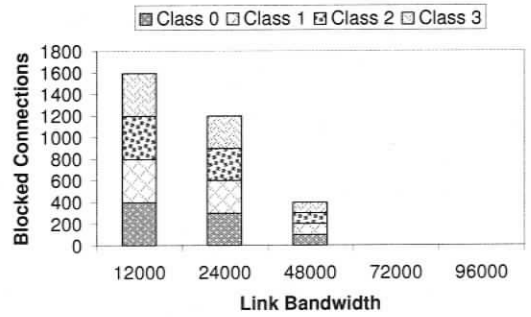


Figure 4.15 : Mesh topology, L-LSP method, simulation scenario 2

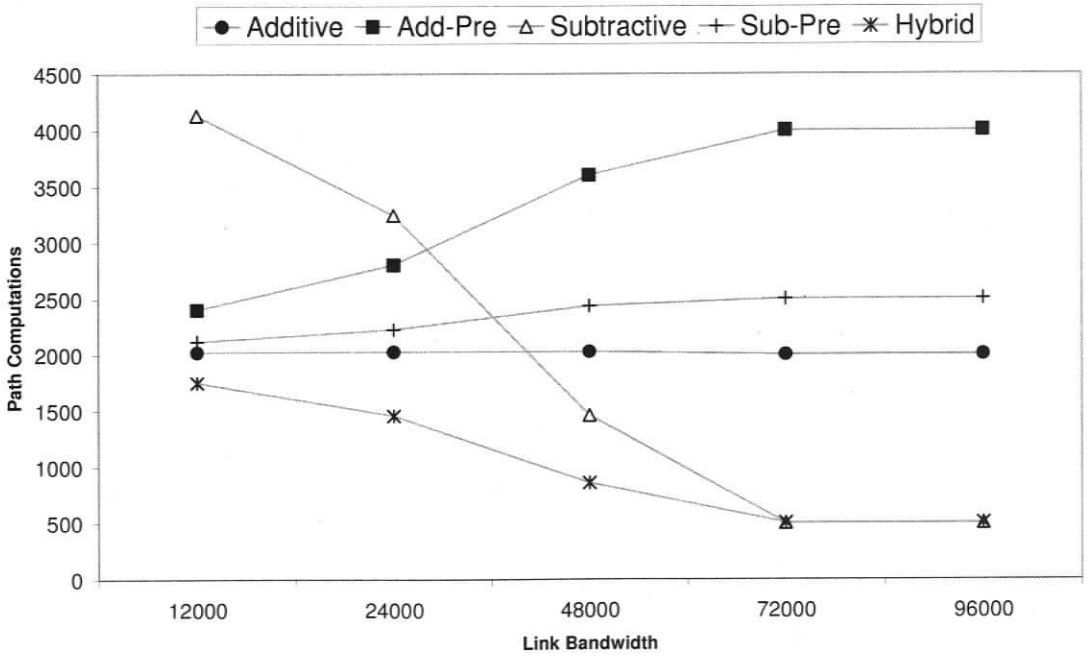
4.4.4 Mesh Topology, E-LSP Method, Scenario 2



(a) (b)



(c)



(d)

Figure 4.16 : Mesh topology, E-LSP method, simulation scenario 2

4.4.5 Mesh Topology, L-LSP Method, Scenario 3

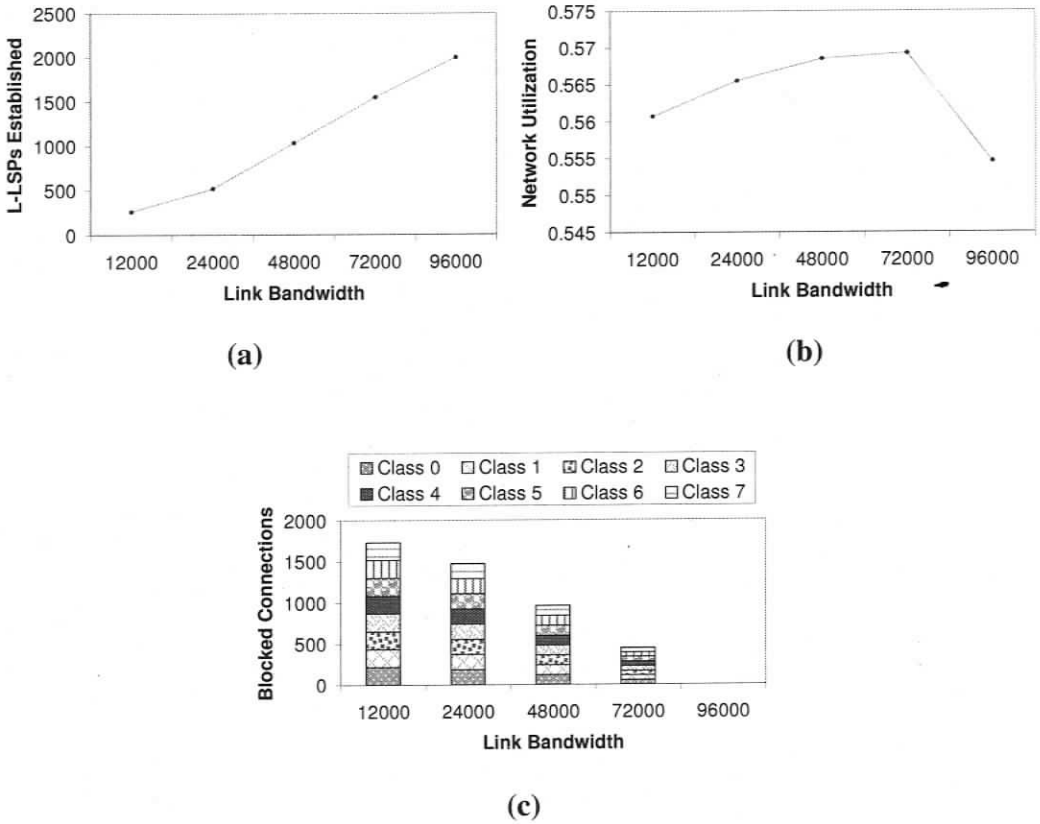
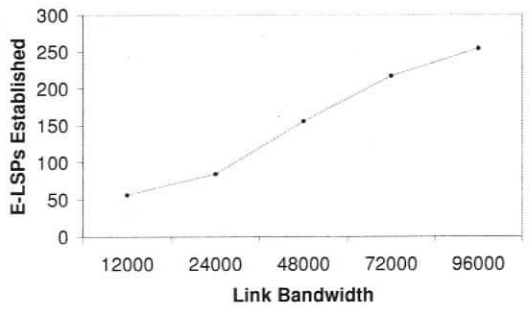
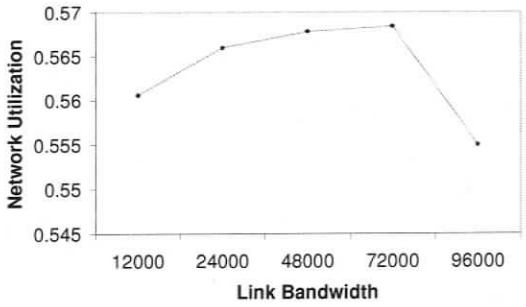


Figure 4.17 : Mesh topology, L-LSP method, simulation scenario 3

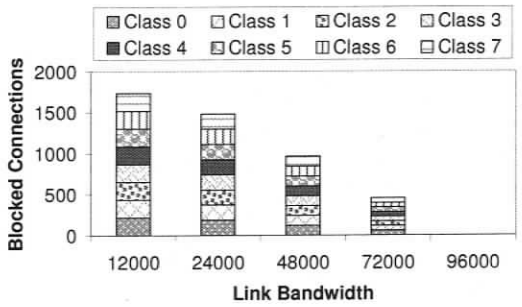
4.4.6 Mesh Topology, E-LSP Method, Scenario 3



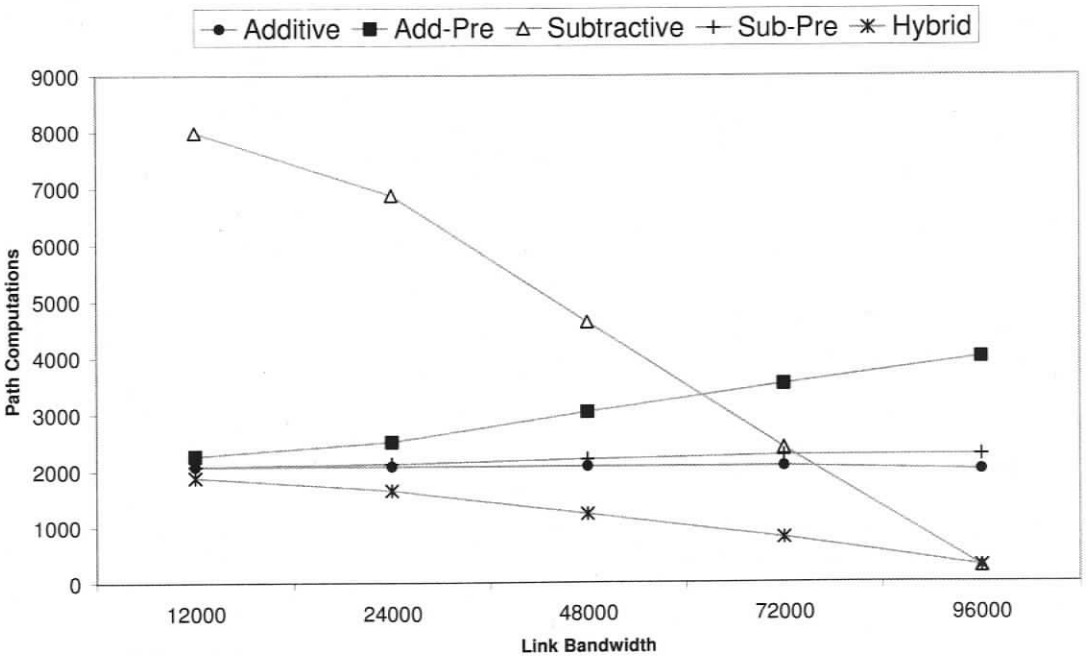
(a)



(b)



(c)



(d)

Figure 4.18 : Mesh topology, E-LSP method, simulation scenario 3

4.5 Increasing Simulation Variability

Another element of the simulation which was addressed in the fourth and final scenario is the randomization of connection requests. In the case where single connection requests arrive and the establishment of L-LSPs takes place, their arrival order was randomized along a uniform distribution, such that each class of traffic is represented approximately equally but their order of arrival at the ingress LER is unpredictable. Where aggregates of traffic classes arrived as a single connection request in the E-LSP method, their contents were randomly permuted within the signaled group. The effect of this permutation is that when adding or removing connection requests from the bundle, the algorithms are no longer selecting those requests with highest/lowest priority to add or remove, but instead are selecting a random connection request to operate upon.

The purpose of this scenario is to show how an unpredictable arrival pattern, and randomized adjustments performed on bundles will impact the results presented thus far. The results confirm that there is not a significant difference in network behaviour, and across each topology there is little variation from the conclusions already presented. For brevity the bifurcated network topology is shown below, as an example selection from the complete scenario results across all topologies. Apart from the increases in variability mentioned, parameters used here are the same as those for scenario 1.

4.5.1 L-LSP Method, Random Traffic Class Arrival Pattern

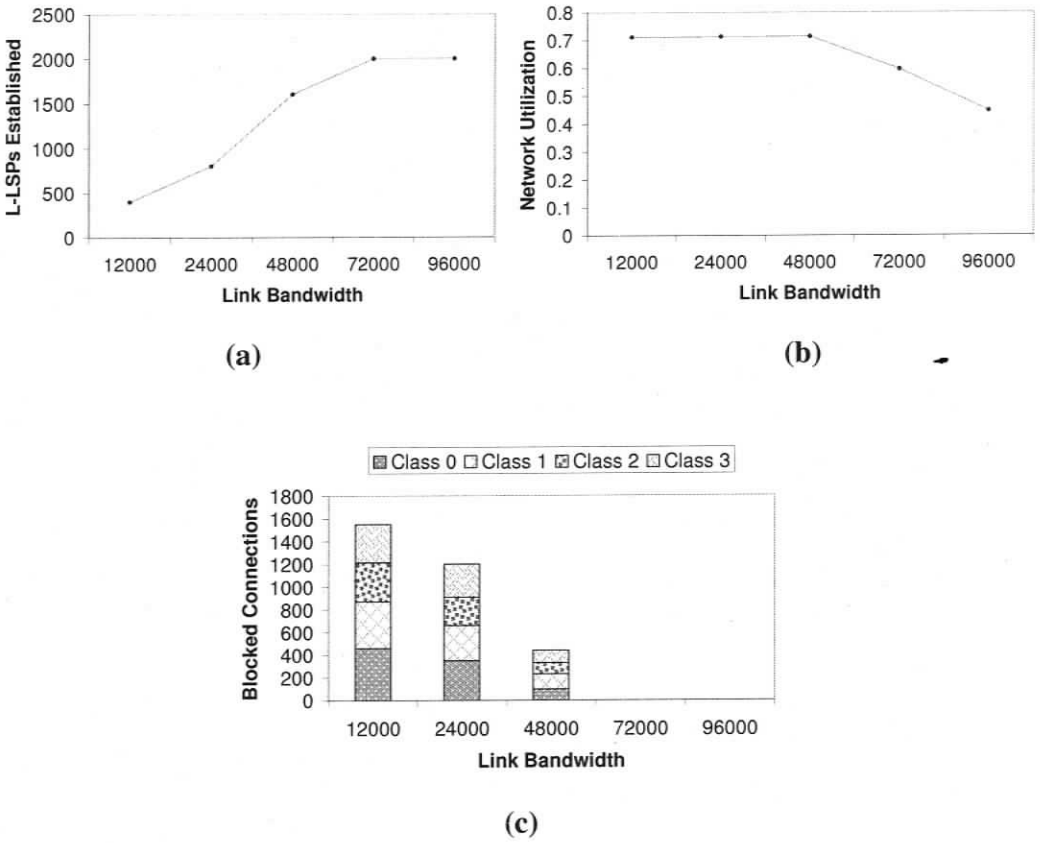
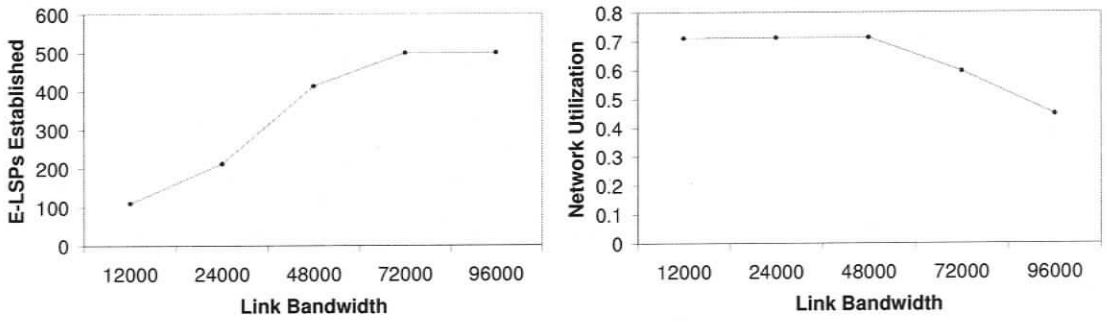
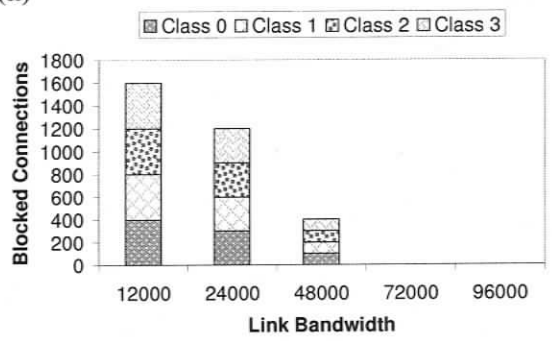


Figure 4.19 : L-LSP method, random traffic class arrival pattern

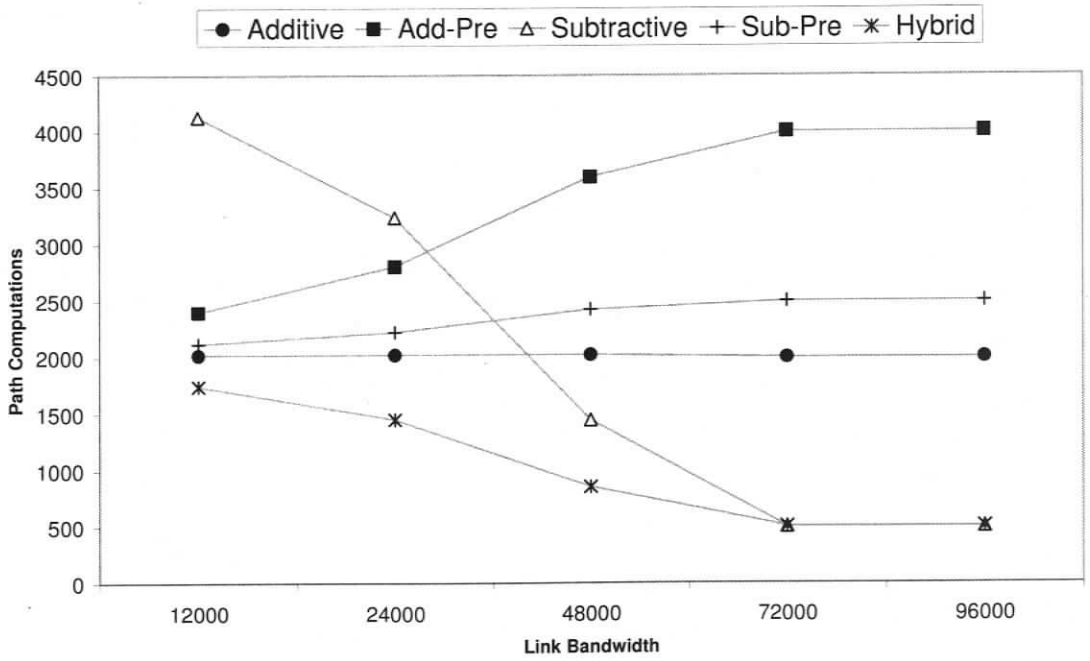
4.5.2 E-LSP Method, Random Permutation of Signaled Group



(a) (b)



(c)



(d)

Figure 4.20 : E-LSP method, random permutation of signaled group

The most significant difference shown above is that there is greater variability in the number of blocked connections per class, due to the imbalance in the total number of connection requests arriving per class for the L-LSP method. In the case of the E-LSP method, this also occurs to a smaller extent, because randomization of the addition or removal of classes from prospective bundles creates a less deterministic environment.

Chapter 5

Conclusions

5.1 Summary

The results presented in this chapter have revealed that aggregating multiple classes of service into a single LSP can be done successfully, with some extensions to the path computation, signaling and label distribution mechanisms which exist today in MPLS networks. The benefits of bundling multiple classes of service in this way include scalability and the simplification of network administration and maintenance. The E-LSP method of differentiating classes of service across a single LSP presented here shows that the number of LSPs which must be established in a network can be reduced by a factor of at least two, and sometimes a great deal more. The resultant reduction in network state complexity is an important gain from a traffic engineering perspective.

The comparison between algorithms for aggregating these traffic classes together reveals that there may exist a large disparity in the number of path computation invocations which must be made, depending upon the network characteristics and administratively configured variables. This additional overhead can be reduced to tolerable levels with prudent network provisioning and an understanding of the conditions under which the algorithms perform well or poorly. It is believed that in cases where a network might adopt the strategies mentioned in the body of this thesis, a combination of fair bandwidth partitioning along with well established class-of-service requirements must provide the foundation from which the required algorithm(s) can be included in an effective manner. In this way, the results presented here serve as an indicator of an efficient means to reduce the total number of LSPs which must be signaled and established in an MPLS network, with the important caveat that

administrative decisions with regard to network setup and provisioning are critical to any successful deployment.

5.2 Main Contributions

The extensions described for signaling, path computation and link state maintenance protocols together provide a new framework upon which classes-of-service in MPLS networks may be aggregated. The consequent reduction in the total number of LSPs which must be established in the network is an important contribution to scalability and reduced complexity in management and administration.

An analysis of this proposed framework across multiple simulation scenarios, with differing topologies and traffic parameters, is a significant and concrete contribution to understanding the characteristics of such an approach. With results derived from the simulation environment speculation is greatly reduced; the current state of the art in this regard can move forward with inferences from this work as to the results of deployment on a still larger scale.

Additionally, the algorithms developed for "bundling" classes-of-service is a new undertaking which yields information not addressed in the general view of MPLS networks. Comparisons have been made not only between the proposed extensions and the existing implementation, but also across various algorithmic alternatives for aggregation with diverse applications and features. By examining the total number of path computation invocations required and the number of paths established, there exists here a quantitative measure of each. As such, the circumstances under which these alternative methods may be applied with favourable results is a contribution which has not previously been addressed.

5.3 Future Work

Throughout the course of this work further refinements and additional areas of research revealed themselves. Some of those which grew beyond the scope of this thesis, but are of interest and relevance to the work presented are outlined below.

- *Complexity of parameters for simulation*: With a limited availability of time and computing resources, a subset of possible traffic requirements was selected. There exist many more variations, with the inclusion of a greater number of constraints. The selection of traffic parameters is a somewhat arbitrary endeavor, and simulation studies across many more sets of potential scenarios could yield further interesting results.

- *Additions and optimizations to the bundling algorithms*: There exist many possible sources to draw algorithmic inspiration from for handling the problem of how to compose and divide classes of service into aggregate bundles. Some changes may be substantive, such as a lookahead window for additive bundling to require fewer path computation invocations, where others could be as simple as a reduction in the number of path computations by one or two with the realization that occasionally a check is unnecessary as the results are implied by a previous step. As the results have shown, the preprocessing step of removing blocking single connection requests has a high overhead when bandwidth is widely available in the network; an adaptive algorithm to determine when preprocessing will become beneficial is another possible enhancement. Testing refinements and new algorithms are other areas of interest.

- *Theoretical formulations*: The theoretical work required to analyze and give formal treatment to the problem of QoS routing in terms of aggregated traffic classes grew beyond the scope of this work, but is of importance to further understanding the challenges involved. Further research would prove beneficial from an algorithmic and simulation development standpoint.

- *Inclusion of more MPLS functionality*: To keep the simulation relatively straightforward some MPLS functionality was not incorporated. For example, path protection (establishing one or more additional paths for a single set of connections as a failsafe), and setup/holding priority for pre-emption of existing paths were not implemented. The inclusion of such mechanisms to the model would give an increased understanding of the results of deployment in a dynamic network environment.

Bibliography

[1] S. Alvarez, "QoS for IP/MPLS networks", pp. 57-77, Cisco Press, June 2006.

[2] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP specification", RFC 3036, Internet Engineering Task Force, January 2001.

[3] L. Andersson and G. Swallow, "The multiprotocol label switching (MPLS) working group decision on MPLS signaling protocols", RFC 3468, Internet Engineering Task Force, February 2003.

[4] D. Awduche, L. Burger, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: extensions to RSVP for LSP tunnels", RFC 3209, Internet Engineering Task Force, December 2001.

[5] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS", RFC 2702, Internet Engineering Task Force, September 1999.

[6] A.R. Bashandy, E. Chong, and A. Ghafoor, "Generalized quality-of-service routing with resource allocation", IEEE Journal on Selected Areas in Communications, vol. 23, no. 22, pp. 450-463, February 2005.

[7] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview", RFC 1633, Internet Engineering Task Force, June 1994.

[8] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol", RFC 2205, Internet Engineering Task Force, September 1997.

[9] S. Chen and K. Nahrstedt, "On finding multiconstrained paths", Proc. IEEE Int. Conf. Communications (ICC'98), vol. 2, pp. 874–879, June 1998.

[10] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: problems and solutions", IEEE Network Magazine, vol. 12, pp. 64-79, December 1998.

[11] A. Farrel, P. Papadimitriou, J. Vasseur, and A. Ayyangar, "Encoding of attributes for MPLS label switched path establishment using RSVP-TE", RFC 4420, Internet Engineering Task Force, February 2006.

[12] S. Ganti, N. Seddigh, and B. Nandy, "MPLS support of differentiated services using E-LSP", Internet Draft draft-mpls-diffserv-elsp-02.txt, Internet Engineering Task Force, June 2002.

[13] S. Ganti, N. Seddigh, and B. Nandy, "DS-TE requirements for support of multiple-COS on an E-LSP", Internet Draft draft-tewg-diffserv-multicos-elspreq-00.txt, Internet Engineering Task Force, February 2002.

[14] INET Framework: Documentation and Tutorials, [Online] Available: http://www.omnetpp.org/staticpages/index.php?page=20041019113420757

[15] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, and P. Doolan, "Constraint-based LSP setup using LDP", RFC 3212, Internet Engineering Task Force, January 2002.

[16] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering extensions to OSPF version 2", RFC 3630, Internet Engineering Task Force, September 2003.

[17] K. Kompella and J. Lang, "Procedures for modifying the resource reservation protocol", RFC 3936, Internet Engineering Task Force, October 2004.

[18] F.A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem, "Performance evaluation of constraint-based path selection algorithms", IEEE Network Magazine, vol.18, no. 5, pp. 16-23, September/October 2004.

[19] F. Le Faucheur, "Maximum allocation bandwidth model for diffserv-aware MPLS traffic engineering", RFC 4125, Internet Engineering Task Force, June 2005.

[20] F. Le Faucheur, "Russian dolls bandwidth model for diffserv-aware MPLS traffic engineering", RFC 4127, Internet Engineering Task Force, June 2005.

[21] F. Le Faucheur and W. Lai, "Requirements for support of differentiated services-aware MPLS traffic engineering", RFC 3564, Internet Engineering Task Force, July 2003.

[22] F. Le Faucheur, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen, "MPLS support of differentiated services", RFC 3270, Internet Engineering Task Force, May 2002.

[23] B. Lee, S. Ganti, A. Srinivasan, W.J. Carpini, U.M. Neustadter, C. Dang, and V. Wong, "Multi-constraint routing system and method", United States Patent 6925061, August 2005.

[24] E. Mannie, "Generalized multi-protocol label switching (GMPLS) architecture", RFC 3945, Internet Engineering Task Force, October 2004.

[25] I. Minei, "MPLS diffserv-aware traffic engineering", Juniper Networks white paper, part number 200048-001, 2004.

[26] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers", RFC 2474, Internet Engineering Task Force, December 1998.

[27] OMNeT++ Discrete Event Simulation System, [Online] Available: http://www.omnetpp.org

[28] E. Rosen et al., "MPLS label stack encoding", RFC 3032, Internet Engineering Task Force, January 2001.

[29] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture", RFC 3031, Internet Engineering Task Force, January 2001.

[30] A. Tanenbaum, "Computer Networks", Fourth Edition, Prentice Hall PTR, August 2002.

[31] P. Van Mieghem and F.A. Kuipers, "Concepts of exact QoS routing algorithms", IEEE/ACM Transactions on Networking, vol. 12, no. 5, pp. 851-864, October 2004.