DUBLIN CITY UNIVERSITY

School of Electronic Engineering

A Flexible, Abstract Network Optimisation Framework and its Application to Telecommunications Network Design and Configuration Problems

Sean Murphy BEng.

Theses submitted in partial fulfilment of the requirements for the award of PhD.

Name of Supervisors: Prof. Thomas Curran

Dr. Dmitri Botvich

Submission Date: 18th June 2001.

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of PhD is entirely my own work and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ____________________ ID Number: 93700288

Date: 7/9/2001

ABSTRACT

A flexible, generic network optimisation framework is described. The purpose of this framework is to reduce the effort required to solve particular network optimisation problems. The essential idea behind the framework is to develop a generic network optimisation problem to which many network optimisation problems can be mapped. A number of approaches to solve this generic problem can then be developed. To solve some specific network design or configuration problem the specific problem is mapped to the generic problem and one of the problem solvers is used to obtain a solution. This solution is then mapped back to the specific problem domain. Using the framework in this way, a network optimisation problem can be solved using less effort than modelling the problem and developing some algorithm to solve the model.

The use of the framework is illustrated in two separate problems: design of an enterprise network to accommodate voice and data traffic and configuration of a core diffserv/MPLS network. In both cases, the framework enabled solutions to be found with less effort than would be required if a more direct approach was used.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Prof. Tommy Curran for his support, encouragement, insight and input throughout the course of this work.

Dr. Dmitri Botvich was also an invaluable help without whom this thesis would not have been completed. The many long and wide-ranging discussions that took place in his office were simultaneously useful, stimulating, entertaining and enlightening.

Particular thanks are also due to Rob Brennan who gave generously of his time in assisting in the preparation of the defence of my thesis.

There are many others with whom I've had the pleasure to work with in various capacities throughout the course of this work, many of whom have spent some time within the hallowed confines of the Advanced Telecommunications Research Lab that is J119. Most of these people have given me faith, hope and encouragement at various times throughout this work, not to mention a little light relief from the toil of my research. I could mention many people here, but I would certainly omit some. Hence, I don't want to give an exhaustive list. Rather, I want to thank the following in particular for their encouraging words: Dr. Noel-Edward O'Connor, Saman Cooray, Nicola Cooke, Hai Wang and Eddie Cooke.

My family have also been a rock of support over the last number of years, without quite understanding what I was up when I came to DCU each day. Even though my little brother Liam availed of any opportunity to mock me with the 'eternal student' line, it was always said in a good-natured manner. My parents have been particularly supportive over the years and I want to extend a heartfelt thanks to them.

Last, but by no means least, I would like to thank my girlfriend Dr. Ethel Ryan. Ethel and I started seeing each other seven months before I submitted my thesis – a stressful time for any relationship, let alone one that is in its infancy. Ethel has been patient and tolerant while I devoted considerable time and energies to finishing my thesis and she had faith in me when my self-belief was at its nadir. For this, I am and will always be grateful.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AD	Administrative Domain
AF	Assured Forwarding
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
BE	Best Effort
BGP	Border Gateway Protocol
CBR	Constant Bitrate
CBT	Core-Based Tree
CIR	Committed Information Rate
CS	Class Selector
DCR	Dynamic Call Routing
DCS	Digital Cross-connect System
DiffServ	Differentiated Services
DNHR	Dynamic Non-Hierarchical Routing
DSCP	Diffserv Codepoint
D-VPN	Data-Virtual Private Network
ECMP	Equal Cost Multi-Path
EF	Expedited Forwarding
FR	Frame Relay
GA	Genetic Algorithm
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol

IS-IS	Intermediate System-Intermediate System
ISP	Internet Service Provider
LAN	Local Area Network
LDP	Label Distribution Protocol
LSAR	Load Sharing and Alternate Routing
LSP	Label Switched Path
LSR	Label Switched Router
MBAC	Measurement Based Admission Control
MPLS	Multi-Protocol Label Switching
OSPF	Open Shortest Path First
PBX	Private Branch Exchange
PDH	Plesiochronous Digital Hierarchy
PHB	Per-Hop Behaviour
PIM	Protocol Independent Multicast
POP	Point of Presence
PSTN	Public Switched Telephony Network
QoS	Quality of Service
RCAR	Residual Capacity Alternate Routing
RED	Random Early Detection
RIO	RED for In and Out traffic
RIP	Routing Information Protocol
RSVP	Reservation Protocol
RSVP-TE	RSVP with Traffic Engineering extensions
RTNR	Real Time Network Routing
SDH	Synchronous Digital Hierarchy
SHR	Self-Healing Rings

SLA	Service Level Agreement
SONET	Synchronous Optical Network
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TOS	Type of Service
UA	Unified Algorithm
VBR	Variable Bitrate
VCI/VPI	Virtual Channel Identifier/Virtual Path Identifier
VP	Virtual Path
V-VPN	Voice-Virtual Private Network
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WRR	Weighted Round Robin

CHAPTER 1 INTRODUCTION

Network design and configuration problems arise in a number of areas, notably telecommunications and traffic routing. Such problems have received considerable attention over past decades because they can result in considerable cost savings for interested parties and they are interesting and challenging problems to solve.

Much network optimisation work to date has focussed on the development of specialised, efficient techniques that are applicable to one specific problem. This typically occurred because the computing power available to solve problems was very limited. To solve any reasonable size problems, it was necessary to focus on a very particular problem and usually to simplify it and develop some efficient technique to obtain a solution. This took considerable effort.

Development of such efficient approaches to tackle network design problems is a laborious and time consuming process: problems must be modelled, appropriate simplifications must be made and efficient solution algorithms must be designed. Often devising solution approaches is an iterative process in which one approach is tried which works reasonably well but there is some scope for improvement; methods to improve the approach are identified and implemented, resulting in better solutions, further improvements are proposed and implemented etc. The process can go through many such iterations in the search to make the approach more efficient and to solve larger problems in less time. All of these efforts are quite costly.

A different approach to tackling such problems is considered here. A flexible, generic and more abstract approach to network design and configuration problems is proposed. This approach is more flexible and consequently it can be applied to a number of different scenarios. When compared with the specific network design approaches mentioned above and discussed in considerable detail below, the approach described here will, of necessity, be slower. However, since computing power available today is orders of magnitude more powerful and cheaper than that which has gone before, there is an argument for expending more resources in computing power and less on manpower when solving these problems. This is one of the premises on which this work is founded.

1.1 Objectives of this Work

The primary objective of this work is to devise a flexible network optimisation framework that can be usefully applied to a number of different network design and configuration problems. At the core of this framework is a generic network optimisation problem, to which many specific optimisation problems can be mapped. Issues pertaining to development and solution of this generic problem naturally arise.

The objectives of this work can be itemised as follows:

- To develop a flexible network optimisation framework that can be applied to different network design and configuration problems;
- To develop and formulate a suitable generic network optimisation problem on which the framework can be based;
- To devise some approaches to solve the generic problem;
- To demonstrate the use of this framework in the context of some specific network design and configuration problems.

Two specific problems are used to demonstrate the use of this network optimisation framework below. The first scenario is a variant of an enterprise network design problem and the second is an *Internet Protocol* (IP) network configuration problem, which incorporates new technologies that enable delivery of *Quality of Service* (QoS) over IP.

1.2 Thesis Overview

The thesis is structured as follows. Chapter 2 contains quite a comprehensive discussion of the many variants of network design and configuration problems that arise in the telecommunications domain. The discussion ranges from design of facility networks to design of legacy private line based enterprise networks to *Asynchronous Transfer Mode* (ATM) networks to IP networks incorporating new technologies to support new services. This chapter serves to illustrate the variety of network design and configuration problems that arise in the telecommunications area.

The new network optimisation framework is described in chapter 3. The motivation for the idea is first given. Then the generic network design problem that is the crux of this framework is described. The particular problem model that the problem is based on is first presented, together with any assumptions made in the model. Some comments on how the generic problem model can accommodate problems in which assumptions made in the generic problem model do not hold are discussed. The generic problem is then formalised. Some short examples of how the model can be used in particular problems are also presented. This is followed by a discussion of some approaches that can be used to solve the generic problem model.

The next two chapters describe in detail two particular applications of the network optimisation framework. In essence, these are both examples of how the network optimisation framework may be applied.

In the first case, described in chapter 4, the objective is to design an enterprise network consisting of a number of interconnects between premises. The demands consist of a set of voice and data demands and the problem is to determine the lowest cost network design that can accommodate the voice and data demands of the users of the enterprise network. In this chapter, the problem is first discussed in detail, including the assumed network architecture and how it is realised. Then the specific enterprise network problem is formulated. This is then mapped to the generic network design problem, which can be solved using the approaches discussed in the previous chapter. Examples of problems and how they were solved are then given for illustration purposes.

A more topical problem is discussed in chapter 5. A network configuration problem is discussed for IP core networks carrying traffic with QoS requirements. IP-QoS is receiving considerable interest in the research world due to the explosive growth of the Internet over the last 10 years, leading to a near ubiquity of IP. The widespread availability of IP has caused researchers to consider it as the transport layer of choice for future applications. However, traditionally IP has not supported applications with QoS requirements and hence IP networks must be developed to support QoS. Two technologies have been proposed quite recently – *Differentiated Services* (DiffServ) and *Multi-Protocol Label Switching* (MPLS) – which are considered by many to be essential to the delivery of IP-QoS services.

Chapter 5 contains a description of both of these technologies, ranging from high-level architectural description to implementation details that are, of necessity, assumed here to enable the problem to be formulated. A particular set of service offerings are assumed in this problem and these are described. The specific problem is then formulated and the mapping from the specific problem to the generic problem is described. Some example problems are then solved to illustrate the use of the approach. Two sets of examples are given: examples of small problems (problems with 5 to 7 nodes), the solutions of which are simulated and examples of larger problems to demonstrate that the optimisation algorithm is beneficial.

CHAPTER 2 NETWORK DESIGN APPROACHES

2.1 Introduction

In this chapter different network design approaches for different types of networking problems are surveyed. Problems that arise in the following problem domains are discussed:

- Facility network design;
- Packet-switched network design;
- Circuit-switched network design;
- ATM network design.

Problems that arise in each of these different problem domains typically have different objectives and/or constraints. These arise due to the different ways in which network performance is measured in the different problem domains. In facility network design network performance measures are not usually of interest. In packet-switched network design problems, packet delays through the network are typical performance measures and a common objective is to minimise the mean packet delay time for all packets transiting the network. In circuit-switched networks, performance is usually measured in terms of the amount of calls blocked on the network or the amount of revenue generated by the network. Finally, since ATM networks have both packet-like and circuit-like characteristics, both packet level and connection level performance objectives are often specified for ATM network design problems.

Here, different approaches used to solve network design problems that arise in different problem domains are surveyed with a view to constructing a generic model that can be applied to multiple problem domains.

2.2 Facility Network Design

The facility network design problem focuses on how to construct physical networks to carry some set of demands. The networks are typically implemented using *Plesiochronous Digital Hierarchy* (PDH), *Synchronous Optical Network* (SONET), *Synchronous Digital Hierarchy* (SDH) and *Wavelength Division Multiplexing* (WDM) technologies (see [Min92,YST99] for a description of these technologies). These technologies have different characteristics that can alter the characteristics of the design problem.

Various aspects of the overall facility network design problem exist: equipment location problems, architectural design problems and topology and dimensioning problems. Equipment location problems focus on where to locate equipment so as to minimise the overall cost of the network given some set of demands. Typically, these demands would be forecast demands and the problem is a network planning one.

Architectural design problems focus on what architectures to use for different parts of the network. For example, it is often not clear which nodes should be backbone nodes and which nodes should not: the latter are 'homed' on a backbone node. Also, ring topologies are very common when using SDH or SONET and it is often not clear when and where these topologies should be used and when mesh topologies should be used. Determining and costing different facility network architectures is discussed by Doshi and Havardshana in [DH98] and by Cardwell et al in [CMW89].

Topology and dimensioning problems concentrate on where links should exist in the network and what capacity these links should have. These problems can be either green-field problems – problems in which there is no existing network – or problems in which a network exists and the objective is to determine where capacity needs to be added to accommodate increasing demands. These problems could arise as subproblems of a larger facility network design problem which incorporates one or both of the other aspects.

The emphasis in this work is closest to the topology and dimensioning aspects of the overall problem, since it is a fundamental problem in network planning and design. Moreover, only **mesh network design** problems are considered here. In such problems, there is no hierarchical relationship between nodes and there are no architectural restrictions on the way the network can be configured. This is in contrast to networks based on rings, which arise frequently in the SONET/SDH network design, in which the ring nature of parts of the network imposes constraints on the possible node interconnections.

The facility network design problems considered here then are ones in which some set of demands must be routed over some network (which may or may not be given); some cost is associated with routing the demands over the network and the problem is to determine a way of routing the demands that minimises this cost. In this problem, the demands are specified in terms of capacities.

This problem is well-studied since it arises in a number of areas in which significant cost-savings can be made if this is optimised. The problem arises in the areas of telecommunications network design and planning, design of power distribution networks and the more general transportation sciences including problems relating to road network design.

Three different approaches to solve the topology and dimensioning problem are included here. The first approach discussed is one in which the determination of the network topology is decoupled from the problem of determining the routing and link dimensions. In the second approach, the so-called MENTOR algorithm [KKG91] is described which can be used to determine a reasonable topology as well as a routing and hence the resulting dimensions can be obtained. The third approach discussed here is an approach developed by Yaged which was designed to obtain a good routing for a set of demands, but implicitly obtains a reasonable network topology by eliminating uneconomic links. These approaches are discussed in the following sections.

2.2.1 Decoupling the Topology and Dimensioning Problems

A common approach to solve this problem is to decouple the topology and dimensioning problems, i.e. the topological optimisation problem is considered separately from the dimensioning or capacity assignment problem. Indeed, it is common to **nest** the dimensioning problem inside a topological optimisation procedure as shown in Figure 2-1. The dimensioning problem is one of determining how to route the demands on the given network topology such that the overall cost is minimised. The topological optimisation problem involves costing different topologies by choosing different network topologies and determining the cost of these topologies by solving the dimensioning problem. The lowest cost topology is chosen as the best solution.

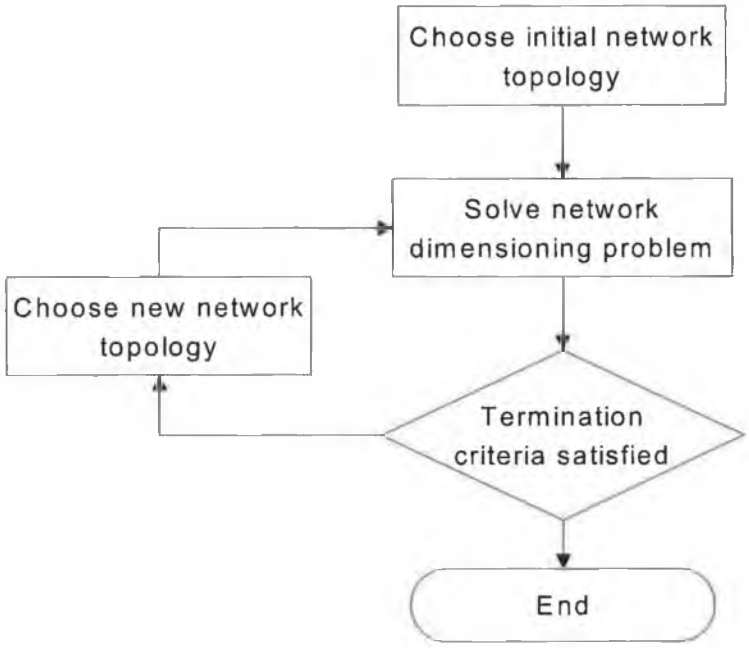

Choose initial network topology
Solve network dimensioning problem
Choose new network topology
Termination criteria satisfied
End

Figure 2-1: Network dimensioning problem nested inside network topological optimisation problem.

The problems are often decoupled in this manner because the coupled form of the problem is often too complex. This is especially true of large network design problems. The decoupled problems can be solved separately and are considerably easier to solve than the substantially more complex coupled problem. They are solved separately in an iterative fashion – the output of one feeding into the input of the other and vice versa until the approach converges to a solution.

This approach arises frequently in network design problems and will be seen below in the context of other types of network design problem.

Topological Optimisation

Two approaches to performing the topological optimisation are discussed here. In the first approach, the topology optimisation problem is considered to be a combinatorial optimisation problem and well-known combinatorial optimisation algorithms are applied to solve it. This approach does not use any specific network knowledge to assist in the optimisation. In the second approach, such knowledge is used to obtain a good network topology.

Both approaches are examples of so-called branch-exchange approaches to solve topological optimisation problems. Branch-exchange approaches are iterative approaches in which links are added and/or removed at each iteration until some convergence criteria is satisfied. Here, it is useful to consider the set of possible network topologies to be represented by a state space. The problem can be solved by iterating through the topology state space and determining the routing and associated cost for each topology until a good solution is obtained.

McGibney [McG95] considers the application of general combinatoric optimization approaches to determine a good network topology in detail. McGibney uses link costs of the form shown in Figure 2-2: the link costs are linear with a non-zero offset. The total network cost is the sum of the individual link costs.

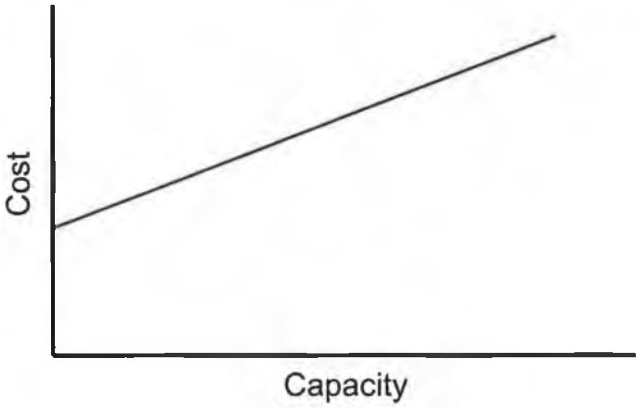


Figure 2-2: Link Cost function used in [McG95].

McGibney compares the use of different well-known heuristic algorithms that can be used to obtain some solution to combinatorial optimisation problems. He uses variants of well-known heuristics such as greedy algorithms, simulated annealing algorithms and genetic algorithms. Interestingly, he found that there are very many local minima in the problems he studied. Furthermore, these local minima often do not differ greatly in cost and hence the choice of any one of these is a reasonable solution to the problem.

The other approach to obtain a good topology is to consider properties of the current solution and to use this to determine how to choose the next solution. One variant of this approach is one in which the starting solution is a fully connected network and links with the highest cost per bit are removed until no more overall cost improvements can be made.

Other variants of this approach are possible in which knowledge of the result of the dimensioning problem can be used to guide the topology optimisation process. This is true of situations in which the topology optimisation problem arises in problems other than the facility network design problem, as will be seen below in the context of packet network design problems.

The Capacity and Flow Assignment Problem for Facility Networks

The capacity and flow assignment problem is to determine the optimal routing of the demands and the optimal link capacities for the given set of demands, cost functions and network topology. In this case, the set of routes is a sufficient solution to the problem: the set of link capacities can be determined from the route configuration and the demands.

Different variations of this problem exist:

- the demands can be a set of time-dependent demands or they can be a single set of time-independent demands;
- splitting of the demands may or may not be permitted;
- there can be an existing network or there may be no existing network: the problem may be a green field network design problem, or may be one in which the objective is to minimise the costs of augmenting capacity to the network;
- the link cost functions can vary.

Different solution approaches have been proposed to solve problems with different characteristics. Some variants are discussed here.

If the link cost function is linear, as is the case in the problem studied by McGibney, the optimal routing for the demands can be determined by solving a standard Floyd-Warshall shortest path routing algorithm [GM84]. In this case, the link weights should be the slope of the link cost functions.

The flow deviation[1] method can be applied quite generally to obtain a routing for a given topology if the link cost functions are differentiable. Yaged used this approach in

[1] The flow-deviation approach is a variant of the well-known Frank-Wolfe method that can be used to solve general, non-linear programming problems with convex constraint sets. It is an iterative approach that involves iterating through link flow vectors until a minimising vector is found. The approach involves determining the derivative of the objective function with respect to the link flow vector for the current link flow. A vector of derivatives is found – one for each link. These values are then used as the link weights for a shortest path routing problem, the solution of which results in another set of link flows. The next link flow vector is that vector between the current vector and that obtained from the shortest path routing which minimises the objective function. If no reduction in the objective function is obtained with this move, then the algorithm terminates. The rationale for this approach – which is based on some unintuitive characteristics of the optimal solution – is discussed in more detail in [BG87].

his work and found that it had the desirable property of removing uneconomic links. Hence, it is considered below as a separate approach to solve the entire topological, routing and dimensioning problem. Here, it serves to note that it is one approach to solving the capacity and flow assignment problem.

If the link cost functions are non-differentiable, then the problem can be considerably more complex to solve, especially when considered as part of a loop iterating through different topologies. The link cost function could be modelled using a differentiable cost function and the more standard algorithms could be used in this case.

2.2.2 The MENTOR Algorithm

The MENTOR algorithm [KKG91,Ker93] is a different approach in which the network topology is generated more directly based on the set of demands. In this approach, a spanning tree is generated quite quickly based on the demands and this is used as a kind of minimal network on which to route the demands: extra links are added if cost efficient. A high-level view of the MENTOR algorithm is shown in Figure 2-3. Each step in the algorithm is discussed in more detail below.

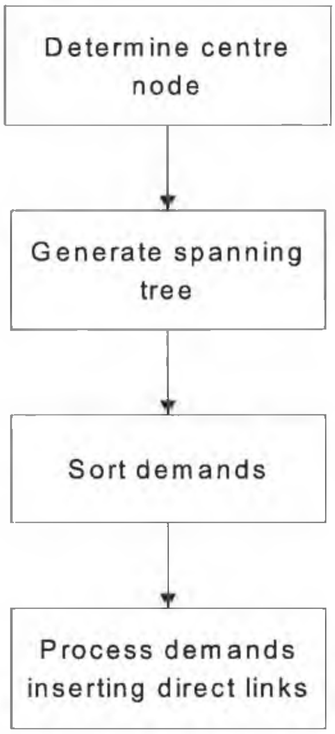


Figure 2-3: The MENTOR algorithm.

The spanning tree is generated by first identifying one node in the problem which is relatively close to the other nodes and has a reasonably large amount of traffic switched through it. This node is termed the centre node and this is the first node in the tree. For each node, a figure of merit which is some combination of distance to the tree and demand generated by the node is calculated. The node with the highest such figure is then added to the tree via a direct link. The figures of merit are then updated and the node with the highest figure is again added to the tree. In this way all the nodes are added to the tree and a spanning tree results.

When the spanning tree is determined, the demands are then considered in turn. A test is performed to see if installation of the direct link is warranted between the demand's source and destination. This test could be to determine if there is sufficient demand on the direct link to use some high proportion of the link. If the direct link is warranted, then it is installed. Otherwise the demand is part-routed on the spanning tree: the demand is routed to the next node in the spanning tree, where it is aggregated with other demands.

The order in which the demands are considered is important in this problem. The approach allows for demands that do not warrant a direct link to be aggregated with other demands. It is possible to aggregate two sets of demands which could warrant a direct link when considered together, but neither of which would warrant a direct link when considered on its own. Some demands can be viewed as dependent on others – if traffic from one demand could potentially be aggregated with traffic from another demand, then second demand is dependent on the first. Kershenbaum et al devised an elegant algorithm to determine the dependencies between the demands to determine a sequence in which to process the demands such that the dependent demands are considered after the demands on which they are dependent. The demands are then considered in the manner described above and a network topology and set of link utilisations results.

The big advantage of the MENTOR algorithm is that it requires considerably less iterations: the algorithm runs in $o(N^2)$ time. However, it does have a significant drawback: the algorithm only caters for demands of a single capacity. The algorithm does not contain a link model in which the cost of the link varies with the used capacity of the link – only a single capacity is permitted.

2.2.3 A Dimensioning Approach that Eliminates Uneconomic Links

Yaged studied the network topological optimisation problem in the early 70's [Yag71]. He studied a problem in which a set of demands had to be routed over a network. In this particular variant, the link cost functions were continuous, concave cost functions of the link capacity. The objective was to determine how to route the demands on the network such that the overall network cost – the sum of the link costs – was minimised.

This can be a difficult problem to solve, especially for larger numbers of nodes. The solution space typically has a substantial number of local minima and hence it is difficult to find a globally optimal solution. For larger problems, it is not even so straightforward to obtain a locally optimal solution.

In essence, the approach used by Yaged was a flow-deviation approach. He determined some characteristics of locally optimal solutions: specifically, such solutions are shortest path solutions to a problem with the same topology and some specific set of link weights. The problem then becomes one of determining these link weights. Yaged proposed an iterative approach to solve the problem in which the solution at iteration $i+1$ is obtained by performing a specific mapping on the solution obtained at iteration i. This continues until the solution converges. The mapping that Yaged suggests is one in which the new routing can be obtained by determining the shortest path routing on a network with a specific set of weights. These weights are obtained by differentiating the link cost functions and choosing the value of the derivative at the current level of demand.

This approach has a tendency to reduce the capacity of uneconomical links to 0. As such, these links can be removed and the topology that emerges is simply the set of links which have a capacity greater than 0. Thus, this approach can solve the topology, routing and dimensioning problems.

2.2.4 Reliability Problems

Network design with reliability is an area that has received considerable attention in recent years. This is due to the improvements in functionality of telecommunications equipment; in particular the fault recovery functionality available in *Self-Healing Ring* (SHR) and *Digital Cross-connect System* (DCS) equipment mean that networks have support for automatic reconfiguration in case of failure [Wu92,VHS96]. This new functionality gives rise to new and interesting network design problems. Also, many organisations are more dependent on their network to meet their business needs and consequently are demanding that operators guarantee some level of service e.g. 99.999% service availability over some time period. This means that the network operator has to take some measures to ensure that the network can react in the case of failure of a network component or a link failure to ensure that this level of service can be guaranteed to the customer.

Automatic recovery mechanisms also mean that the network operator does not have to dispatch a maintenance team to immediately solve the problem – the situation can be examined and the problem solved at some later date. This means that it can be less costly to fix the problem.

Network reliability has had an impact on the way in which networks are designed. Networks, and facility networks in particular, are now designed with reliability constraints in mind. A number of variations of network design problems incorporating reliability concerns have been discussed in the literature. Some have concentrated on spare capacity placement in an existing network, while others have been more concerned with the network dimensioning problem and considering how to plan spare capacity into the network at design time.

In [IMG98] an integer programming formulation for the network design problem including reliability constraints is given. A standard integer program solver is then used to solve the problem. In this approach, it is assumed that the costs are linear in capacity and that the demands are given in capacity units and that these must be rerouted. The approach that they propose can be used to design a mesh restorable SDH/SONET network which uses path restoration. It can also be used to solve a variant of the ATM analogue in which the demands are multiples of some fundamental unit of demand. This work can also be used to dimension the spare capacity required to be added to an existing network with given demands and demand routing to obtain some level of restorability.

Kheradpir et al consider the reliability problem in a dynamic bandwidth context [KGS96]. The network must cater for demands dynamically. The problem is to determine whether the demand can be accepted without compromising the reliability of the network. The authors consider how the network can be configured such that the maximum amount of demands can be carried while still meeting the reliability constraints.

They wish to obtain full network restorability. By this they mean that any one of some predefined set of failures can occur and the network will be able to reroute **all** the traffic carried on the network such that all the original demands are met. If the demands are sufficiently large, it may not be possible to route the demands on the network while maintaining the survivability conditions. Consequently, some of the demands may be rejected. The author's objective is to maximise the residual capacity on the links in the case of failure and minimise the maximum blocked demand in normal operating conditions. This results in an equilibrium programming problem.

Kheradpir et al propose a parallel algorithm to solve this problem. The algorithm works by considering the normal operating situation and each of the failure scenarios almost independently and solving a problem for each individual scenario. There is some coupling between the problems that manifests itself if for any source-destination pair the residual capacity on the least loaded path between the nodes has capacity less than or equal to 0. If it is not possible to route a predicted demand in a particular failure scenario, then the demand is limited to the demand in the previous period for all failure scenarios. For each iteration of the loop, some of the demand is allocated to the least loaded path. The least loaded paths are again recalculated and some of the demands are again apportioned to the least loaded path. This is repeated until the all the demands are allocated or no more can be allocated without breaking the reliability constraints.

Herzberg and Bye concentrate on the problem of adding capacity to an existing network to ensure certain survivability constraints are met [HB94,HBU95]. They assume that there is an existing network, and that some fraction of the traffic for each node pair needs to be rerouted in the case of failure. Some set of failures is given in advance – typically a set of single link failures – and the problem is to determine how to add capacity to the network such that the cost of the additional capacity is minimised.

Herzberg and Bye formulate the problem as an integer program. To obtain integer solutions, they determine a solution to the relaxed linear program, and then round up the solution to the nearest integer-valued solution. Next, they apply a so-called 'tightening algorithm' to the solution in order to reduce the cost of the solution. This operates by considering each link in turn and determining if unit reduction of the link capacity still results in a feasible solution. If so, then the link capacity is reduced by one. The next link is then considered, etc. The tightening algorithm reduces the solution so that it approaches the solution found by the linear program.

Other contributions have focussed on different aspects of the reliability problem: Balakrishnan et al consider the reliability problem in the context of topological optimisation [BMM98] and there are others. Network reliability issues are not incorporated into this work. However, reliable network design is an important issue and any review of network design cannot omit a discussion of reliability concerns.

2.3 Packet-switched Network Design

Considerable efforts were expended on the design of packet-switched networks during their earliest implementations. In particular, pioneers in the field such as Kleinrock and Gerla spent much time modelling and analysing the ARPAnet before its introduction. They formulated some design problems and developed interesting approaches to solve the problems.

Today's packet-switched networks have evolved significantly from those of the mid-late 70's; improvements in both computing power and transmission technology have dramatically increased the speeds at which networks can operate. The exponential growth in demands fuelled by the internet over recent years has meant that the aforementioned technological improvements have been deployed in existing networks to accommodate the increasing demands.

While networking technology in general has developed significantly from the networks studied by Kleinrock and Gerla, the use of IP, which was first used in the ARPAnet, has grown enormously and IP is fundamental to the operation of today's internet. Consequently, the early work done on packet-switched network design is still somewhat applicable in today's networks.

However, new sophisticated applications are creating a demand for more complex network functionality. Applications require two important network functionalities which are as yet not implemented on a wide scale: multicast and QoS. Broadcast applications, for example require multicast [Obr98,Hui95] functionality in the network and most probably will also require QoS support.

These new functionalities may have a considerable impact on the network design problem. Multicast can have a profound impact on network dimensioning due to the tree of network resources that a multicast service requires. The number of parties involved in a communication can now be very large and consequently a large amount of network resources may be used by a single communication. Clearly this will have an effect on network design. Similarly, applications requiring QoS will require reserved resources: the network will no longer operate in simple best effort mode. This, too, will have an impact on the way that the network is designed. Little work has been done to date on the way multicast and QoS will affect the network design problem.

In practice, today's network designers – both enterprise network designers and *Internet Service Provider* (ISP) network designers – tend not to use a rigorous problem formulation and the application of optimisation algorithms to design their networks. Much of the design work is done using rules of thumb and tested using simulation software, and, if necessary, network testbeds [Ker93]. This is because the growth of a network with an organisation has typically been a very evolutionary process in which capacity was added as and when bottlenecks were identified. Little planning took place.

Lloyd-Evans [Llo96] notes that a design based on ad hoc or rules-of-thumb methods can be 10-20% more costly than a design based on a more systematic analysis. This difference can be very significant for reasonable sized networks. Also, the vast increase in the use of communications technologies that has occurred over the last number of years means that networks are considerably larger than they were before and there is more scope for finding ways to make cost-savings.

The remainder of this section is structured as follows. First, routing in packet-switched network is discussed, since routing typically has a strong impact on dimensioning. Next, some systematic approaches to network dimensioning in packet-switched networks are discussed. These are followed by a discussion of a slightly different problem: that of determining how to configure a logical packet-switched network on some physical network.

2.3.1 Routing in Packet-switched Networks

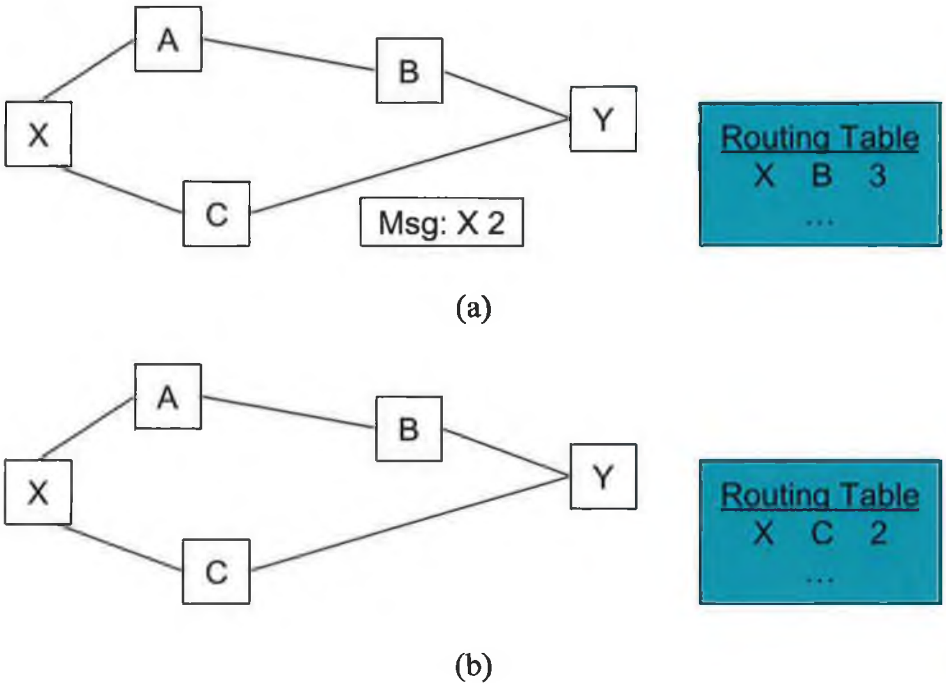
A reasonably concise overview of routing in packet networks is included here. Routing in packet networks is quite a complex subject and has been studied for many years. The purpose of routing in networks is to determine a path from a source to a destination in the network. While this is the fundamental purpose of routing, there are other concerns that affect the way that routing operates – efficiency and stability for example.

Here, a number of different aspects of routing are discussed. First, unicast routing is discussed. Three possible approaches to performing unicast routing are discussed: distance vector based routing, link state based routing and MPLS. This is followed by some discussion of multicast routing, which, in turn, is followed by some discussion of QoS and routing.

Distance Vector Routing

Distance vector routing was the first approach to facilitate routing in packet-switched networks. It is quite a straightforward approach in which each node broadcasts reachability information to its neighbouring nodes. These neighbouring nodes forward the newly received reachability information onto their neighbours in turn. In this way, the reachability information is propagated through the entire network.

When a node obtains reachability information for a new address or set of addresses, this information is added to the node's routing table. The routing table is augmented with the destination address/set of addresses and the node from which this reachability information was obtained; this node then becomes the next hop in the path to this/these destination(s).



(a)

(b)

Figure 2-4: Illustration of the operation of distance vector-based routing. In (a) Node C sends a message to Node Y informing it that C is 2 hops from X. Node Y then incorporates this information into its routing table.

In the distance vector approach, distance information is propagated through the network with reachability information. Thus, each node receives information which can be interpreted as "I am x distance away from destination a.b.c.d" from its neighbouring nodes. Each node can then use this information to identify the shortest path to each destination node. Only information pertaining to the shortest path is retained in the routing tables for each node: if a node receives reachability information from a neighbouring node and an entry already exists for the specific destination addresses, then the node compares distance information. If the new information indicates that a shorter path to the destination exists through this neighbouring node, then the routing table is updated to indicate that this node has become the next-hop for this destination. This is illustrated in Figure 2-4.

Transmission of the distance information is quite simple in practice. Each node receives distance and reachability information from its neighbouring nodes. It then forwards this on to its neighbouring nodes. However, before forwarding on this information, it increases the distance to the destinations. For example, if a node receives information from a neighbouring node that can be interpreted as "I am 3 nodes away from node a.b.c.d," it will then forward information which can be interpreted as "I am 4 nodes away from node a.b.c.d" to its neighbours. Note, however, that the distance does not have to be a simple hop count measure: more sophisticated measures are possible if more control over the network resources is desired.

Routing Information Protocol (RIP) [RFC2453] and *Border Gateway Protocol* (BGP) [RFC1771] are examples of distance vector based routing protocols.

Distance vector based protocols are not so flexible. In all cases, the shortest path is chosen to the destination. However, the shortest path between a source and destination may be quite congested and for this reason may not be the best path between source and destination.

Distance vector based protocols do have some disadvantages:

- they can take some time to converge. This can be a problem in the case of link failure when the network needs to react quickly to minimise the impact of the failure on the network traffic;
- they are susceptible to oscillatory behaviour in certain circumstances;
- in large networks, they can require transmission of huge amounts of information, and can have a detrimental effect on the network performance as a result;
- they do not differentiate between different link types, and the cost associated with routing on different links.

The above deficiencies with distance vector based protocols stimulated the development of link state routing schemes. It is worth noting, however, that distance vector based routing schemes are still very much used, particularly in the form of the interdomain BGP.

Link State Routing

Link state routing operates using a different paradigm than that of distance vector based routing. In link state routing, network topology information is distributed throughout the network. Each network node then builds up a map of its local domain and each node then determines how to route traffic based on this network map.

In a link state protocol, the reachability information is transmitted with the link state information. Each edge router may be directly connected to a number of stub networks. When broadcasting link state information to other routers, each edge router also distributes information relating to the stub networks it is connected to. This information is then propagated through the entire network, such that the resulting network map in each network node contains information pertaining to the stub networks and not just the core network nodes that provide interconnectivity. In this way, the reachability information is propagated through the network with the link state information.

Since each node has its own network map each node can make its own decision on how to forward packets. In theory, each node can run almost any routing algorithm to determine a next hop to each destination. In practice, each node typically runs a shortest path routing algorithm to determine how to forward packets to each destination node. Consequently, traffic typically follows a shortest path between source and destination nodes.

Open Shortest Path First (OSPF) [RFC2328] and IS-IS (short for Intermediate System-Intermediate System) [RFC1142] are examples of link state routing protocols.

Note that in contrast with distance vector based protocols, the shortest paths can be based on different metrics. In the link state routing paradigm, each link has an associated parameter. This parameter is then used in the shortest path calculations. Typically, these link parameters are simply the inverse of the link capacity, causing routing algorithms to favour links of high capacity over low capacity links. However, they do offer some flexibility in the routing of traffic on the network, and recently some authors [FT00] have been studying how to choose these parameters well to efficiently carry the traffic on the network. Thus, while it is not possible to implement arbitrary routing on networks using link state protocols, a significant degree of flexibility is possible, and many different route configurations can be realised by prudent choice of the link weights. This is certainly more flexible than the more restrictive distance vector based approach, although there is a big problem with this idea: changing the link weights can result in very substantial changes to the way in which traffic is carried on the network.

Link state routing can also be used to support demand splitting: traffic between two nodes can be carried over multiple equal cost paths – this is implemented using so-called *equal-cost multipath* (ECMP) routing and is available in existing equipment. This offers more fine-grain control over routing in the network since it offers the possibility to split demands rather than having all of the demand carried on a single path. ECMP is implemented in some routers by dividing the demand between a number of paths each having the same cost. This provides an extra level of flexibility in routing traffic over the network, but does introduce much greater complexity to the problem of choosing network parameters. Some work has been done on facilitating multiple paths between source-destination pairs to enable an unequal distribution of demand between different paths using OSPF [OMPID].

Multi-Protocol Label Switching

MPLS [RFC3031,Arm00] is a newer approach to routing in packet-switched networks, which is currently receiving much attention in both industry and the research community. MPLS is a technology that was initially developed to facilitate flexible routing of multiple protocols in data networks. Today, due to the overwhelming success of IP, MPLS is mainly considered in the context of IP networks and is considered primarily for use in core IP networks for three purposes:

- implementation of tunnelling;
- traffic engineering;
- QoS support.

In MPLS, each packet has a label. Packets are switched at MPLS routers (so-called *Label Switched Routers* or LSRs) based on labels. When a labelled packet arrives at an LSR on one interface, the LSR looks up a routing table to determine what interface the packet should be sent to and what label should be assigned to it. Using this label switching approach, *Label Switched Paths* (LSPs), or paths which packets follow through the network can easily be configured. MPLS permits arbitrary paths to be configured for arbitrary traffic in the network: clearly, this is much more flexible than the either distance vector or link state based routing.

An important element of the MPLS architecture are filters at the edge of the network. These filters may or may not exist and can be used to control what traffic enters which LSP. If a number of LSPs exist between a particular source and destination, then these filters can be used to control which traffic is routed over which LSP. The use of these filters provides more fine-grain control over traffic flows in the network. They can be useful, for example, in situations in which different customers pay for different levels of service: in this case, the different customer's traffic should go into different LSPs.

As with other routing protocols, the essence of MPLS is a protocol that facilitates distribution of routing information. In the case of MPLS, this is the *Label Distribution Protocol* (LDP) [RFC3036]. It is used to distribute information pertaining to the assigned labels and also includes information such as the destination addresses that are reachable via a particular label. The LDP is not discussed in any detail here.

One important aspect of MPLS is that it facilitates a separation of routing and forwarding. Routing relates to the way that the LSPs are routed in the network and forwarding relates to the determination of the next hop for a packet at each node. In the layer 3 routing schemes, routing and forwarding are tightly coupled, but MPLS decouples these operations such that it is possible to implement quite arbitrary routing schemes.

MPLS provides functionality to configure paths through networks that follow an explicit route. Thus, arbitrary paths can be configured through the network and arbitrary routing schemes can be implemented. In practice, some tools would be required to implement such routing schemes.

Explicit routes do not have to be used to configure LSPs; the network maps or forwarding tables generated using layer 3 routing protocols could be used to generate paths in MPLS networks. Using this approach, the resulting set of LSPs will be exactly the same as those resulting from the use of link state routing. However, the resulting network is one in which MPLS routing is used and hence it is inherently more flexible. This could be introduced to facilitate easy migration from a network based on link state routing to one based on MPLS routing. As more sophisticated MPLS control tools become available, the advantages offered by MPLS can be exploited more systematically, but with MPLS routing, even manual mechanisms can be employed to reroute traffic – something that is considerably more difficult with either link state or distance vector based routing.

MPLS was designed with flexibility in mind. Consequently it does permit arbitrary paths to be configured through the network and hence it permits very great flexibility in routing. Also, MPLS permits demand splitting, such that traffic between particular sources can easily be split between different paths. MPLS then offers great flexibility that can be used in the network design problem. The level of flexibility offered is probably too great for network design and some assumptions should be made to limit the amount of design variables.

A more detailed description of MPLS is given below. Here, the most significant aspects of MPLS were discussed with an emphasis on the level of flexibility offered by the technology and how it can impact the network design problem.

Multicast Routing

Multicast routing can be used to support multicast applications which involve communications between a number of parties. These can range from large-scale broadcast applications such as wide-scale video distribution over IP – something similar to today's television service – to much smaller scale videoconference applications or interactive gaming. Multicast can also be used to distribute other data such as say stock prices, news etc.

The fundamental concept in multicast networking is that of the multicast group. Members of a multicast group can either be senders or receivers: senders send to the group, while receivers listen to any information sent to the group. In essence a multicast group translates to a single address that is used to distribute traffic for the group. Routers (in some sense) broadcast traffic sent to this address. Group senders send to the multicast address and rely on the network to distribute information to the receivers; receivers listen to the multicast address to receive information sent by the senders.

Routing protocols are then used to ensure that all parties subscribed to multicast groups receive traffic destined for those groups. Efficiency concerns arise quickly here: it is quite straightforward to ensure that all receivers receive the information: this can be done by broadcasting all multicast information everywhere. However, this obviously generates very large amounts of traffic. Hence, the routing protocol must ensure that information can be delivered to the receivers in an efficient manner.

Two approaches to distribution of multicast information have been proposed: *flood-and-prune* [Dee91] and *core based tree* (CBT) [BFC93]. In the flood-and-prune approach, it is assumed that the multicast information should be distributed to all routers. Hence, the routers automatically broadcast multicast information to all neighbouring routers. If some routers do not want to receive a multicast session, then they explicitly tell their upstream router not to send it via a pruning message. In the CBT approach, the default behaviour is not to transmit a multicast session unless a downstream router explicitly requests it.

These two approaches are essentially different and are most suitable for different applications. The flood-and-prune approach is most suitable for multicast applications in which there is a very wide interest – applications which will be broadcast to a very large set of users. Alternatively, the CBT based approach is much more suited to applications in which the number of users is small.

Protocol Independent Multicast (PIM) [DEF94] is an effort to combine these different approaches. PIM has two modes of operation – dense-mode PIM and sparse-mode PIM – corresponding to the flood-and-prune approach and the CBT approach respectively.

Current approaches to facilitating multicast over IP networks are described above. These approaches, however, have their shortcoming, particularly when viewed in an interdomain context. For this reason there are still ongoing research efforts to provide scalable network support to multicast applications [Alm00].

QoS and Routing

QoS is an issue that is currently of great interest in packet-switched networks. This is due to the wide availability of packet-switched networks, and the need to ensure QoS to achieve acceptable performance for some applications.

Many different proposals have been made in the literature to address aspects of the overall QoS problem, but the question of how end-to-end QoS will be delivered to end-user applications over IP networks in the real world remains unanswered. The problem here is not only to provide QoS but do this both economically and efficiently for both users and service providers. There are a number of reasons that no realistic solution exists, not least of which is the fact that there is not yet a significant application-level demand for QoS support from the network.

A number of different architectures have been proposed which can deliver some levels of QoS to the end-user: most noteworthy amongst these are the diffserv and *Integrated Services* (Intserv) architectures, since they have received considerable interest from both the research community and industry. These architectures do not make any stipulations relating to the operation of routing in the network, and indeed, they can operate with routing mechanisms that are unaware of QoS. However, it is likely that QoS-aware routing will enable more efficient use of the resources and delivery of better QoS to the end user.

Most of the work in routing in QoS networks has focussed on routing in the context of the Intserv architecture and the authors think of routing traffic on a per-application basis. Similar ideas, however, can be employed in the diffserv architecture, although in this case the flows become aggregate flows and the queuing is done on a per-class rather than per-application basis.

Chen and Nahrstedt provide a good overview of QoS routing in [CN98]. There, they describe a number of different ways routing can be implemented in to support QoS. These vary in terms of the assumptions made on the amount of knowledge retained in each node, the granularity with which network state information is distributed, the criteria for choosing routes and the algorithms used to choose routes. Some of the work reviewed there uses source routing, while other work uses distributed next-hop routing[2]. In all cases, routes are chosen based on network state information and hence, the routing schemes can be considered to be adaptive to the state of the network.

Adaptive routing in this manner is quite a change from the more traditional methods of routing in packet-switched networks. For this reason, the impact it will have on the network design, operation and configuration are unclear. However, it is clear that some of the knowledge learnt from the design and operation of telephony networks can be used in this context to help solve the design, operations and planning problems that may arise with QoS-based networks.

[2] In source routing, the route for the traffic is chosen by the source – the nodes between source and destination do not decide how the traffic is routed. In next-hop routing, each node on the route determines the next-hop for the traffic.

2.3.2 Network Dimensioning

Routing has a considerable impact on network dimensioning – an inefficient routing scheme can result in an expensive network to meet some performance objectives, while an efficient routing scheme can result in a considerably cheaper network.

From a dimensioning point of view, observe that the above protocols can impose some constraints on the routing that is effected in the network. If RIP is used, then all the traffic to a particular destination must follow a single path. OSPF permits some splitting of the demand between paths. This is of interest below, and will be discussed further in relation to algorithms that make assumptions on the way in which the routing is performed in the network. MPLS is a much more flexible routing technology and can be used to implement arbitrary routing, although support tools may be required to fully exploit the flexible routing capabilities.

The effects of multicast and QoS traffic on network design are not at all clear. Multicast traffic has the potential to greatly impact the network design problem if a large number of multicast applications will be using the network. Similarly, QoS traffic can have a considerable impact on the network design problem if it is to accommodate a significant amount of traffic with QoS requirements. Since these technologies are not widely deployed, network design approaches taking these into account are not included here.

A significant difference between packet-switched network design problems and network design problems in other contexts is that packet delays are usually of interest in packet-switched network design problems. These can arise either as constraints on the problem or as elements of the objective function. In either case, packet delays are usually aggregated into the overall mean packet delay and this appears either in the problem constraints or the objective function.

Approaches to solve this problem are described next. As with the facility network design problem, two different approaches can be used to solve this problem. In the first approach, the algorithm iterates through the topology state space determining a network cost for each topology as shown in Figure 2-1. In the second, the problem is solved without iterating through different network topologies.

To use the iterative approach it is necessary to be able to find a low cost solution to the capacity and flow assignment problem. This is discussed next, followed by some comments on the iteration through the topology state space. This is followed by some comments on the network design approaches that do not use topological optimisation approach.

The Capacity and Flow Assignment Problem

The capacity and flow assignment problem involves determining how to route a set of demands on a given topology such that some cost is minimised. In this case, the routing alone does not immediately imply some set of link capacities. Rather, the link capacities are also design variables in the problem. This is a characteristic of the packet-switched network design problem. In these problems, the demands are specified in terms of packet arrival rates and the capacities chosen have an impact on the delays experienced in the network. Small capacities can be chosen, but will result in large packet delays. The link capacities are not as obvious from the routing and the set of demands as they are in the facility network design case. Note that mean delay constraints are usually added to the problem to ensure that the resulting network does not have arbitrarily large delays. These constraints ensure that the link capacities remain sufficiently large.

Gerla and Kleinrock [GK77] give a comprehensive discussion of issues associated with solving this problem and they propose a number of different solution techniques. They consider different variants of the problem: essentially, the different variants differ in the choice of link cost function used. A number of different solution algorithms are proposed and the applicability of these algorithms to each of the different problem variants is discussed.

Some simplifying assumptions are made in the model used by Gerla and Kleinrock. Firstly, all demands are assumed to consist of packets arriving according to a Poisson process. This means that all of the queues in the network act like M/M/1 queues: consequently, the queues can be easily analysed and the characteristics of aggregate arrival processes are known. Secondly, some mean packet length is assumed. Thirdly, it is assumed that entire demands are routed on a single path, i.e., a demand is not split between multiple paths.

The simplest case is that in which the link costs are linear in capacity. In this case, a closed form expression for the optimal link capacities in terms of the flow variables can be obtained by assuming that the delay is equal to the bound. The dependence on the link capacities can be eliminated and the resulting problem is one in which the overall cost function is a concave function of the link flows. A large number of local minima exist and they all exist at the edge of the domain. In particular, they all exist at corners of the polyhedron that constitutes the domain. There are very many such corners and it is not practical to search them all individually. The flow deviation approach can be used to obtain a locally optimal solution to this problem.

In the case in which the link costs are a concave function of the link capacities, the objective function is also a concave function of the link flow variables. Consequently, the flow deviation approach can again be applied. However, this situation is slightly more complex because it is not possible to obtain a closed form expression for the optimal link capacities in terms of the link flows and hence the objective function cannot be written as a closed form function of the link flow vector. However, it is still possible to apply the flow deviation approach to obtain a good solution.

The discrete costs case is more complex. Gerla and Kleinrock propose two different high-level approaches to solve the problem. The first approach obtains a solution by iteratively solving a routing problem and then a dimensioning problem until a local minimum is found. The solution to the routing problem is used in the dimensioning problem and the solution to the dimensioning problem is used in the routing problem. The second approach involves approximating the discrete cost function with some concave cost function and solving the resulting problem. The resulting continuous capacities are then converted to the closest discrete capacities and the routing problem is again solved to obtain a good routing.

Gavish and Neuman [GN89] consider an alternative version of the capacity and flow assignment problem. They wish to design the network such that both the delay and network cost is minimised. They consider the link costs to be composed of two components – a fixed cost and some capacity dependent component. This is the same as the cost model use by McGibney and is illustrated in Figure 2-2. In their formulation, the objective function consists of both a delay term and a link cost term. Hence, the optimal solution will be some trade-off between a minimal delay solution and a minimal cost solution.

They make the same assumptions about packet arrival rates and packet sizes as those of Gerla and Kleinrock. They also assume a single route is chosen for each demand. However, their formulation is slightly different in that the set of routes possible for each demand is specified in advance. They also assume that a number of different link types are possible for each individual connection and that each of these links has different cost characteristics.

Gavish and Neuman formulate the problem such that the decision variables are binary variables which represent whether or not a particular route is used and a particular link type is used. These decision variables are the paths that the demands are carried on and the link type that is chosen for each link. The problem is then formulated as an integer programming problem. The number of variables in this formulation can be very large, even for moderate networks.

Gavish and Neuman solve the problem using a variant of the methods of Lagrange multipliers. First, they relax one of the constraints. Specifically, they relax the constraint that relates the link flow variables to the routing variables: the precise relationship that exists between the routing variables and the link flow variables then no longer exists. Both of these are then decision variables in the new problem. They do however ensure that the amount of resulting flow on each link is no less than the amount of flow carried on the link as determined by the routing variables. Then they develop a Lagrangian function. They observe that terms in the resulting Lagrangian function can easily be grouped in a natural way and the Lagrangian function can be decoupled into sets of smaller functions. Minimisation of the Lagrangian can then be broken down into many smaller minimisation problems. Solutions to these subproblems can be obtained separately resulting in set of simple relations between the decision variables and the Lagrange multipliers. The problem then reduces to determination of the optimal set of Lagrange multipliers. This is a non-smooth optimisation problem and a subgradient optimisation approach is used – this is a variant of the steepest descent algorithm that is applicable to non-smooth optimisation problems.

One problem with this approach is that the resulting solution can be quite a distance from any feasible solution in the original problem. This can be attributed to the relaxation of the relationship between the routing parameters and the link flow variables. The authors attempt to improve the solution by imposing tighter bounds on the link flow variables. These bounds are determined through knowledge of the candidate routes. Some demands must be carried on specific links and hence it is possible to determine a lower bound on the amount of flow carried on a link. Similarly by examining the candidate links for each flow, it is possible to determine an upper bound on the amount of flow that can be carried on each link. This technique significantly improves the solution quality.

Gersht and Weihmayer also consider this network design problem in [GW90]. They formulate the entire problem – topology optimisation, flow and capacity assignment. They include node switching capacity as a consideration in the design problem. They impose an upper bound on the delay and they assume linear link cost functions. They assume some set of candidate routes is given as an input to the problem. They also consider multiple different types of facilities in the problem, with different facilities having different cost/capacity characteristics. In their problem formulation, the decision variables are the amount of traffic carried on each path and the presence/absence of links in the resulting network. In the formulation they propose, the demands can be split over multiple paths; the variables determining the amount of traffic carried on each path are continuous variables. The link existence variables are binary variables. The resulting problem is then a mixed integer/linear programming problem.

The problem can be solved using general techniques to solve mixed integer/linear programming problems but Gersht and Weihmayer advocate an approach developed specifically for this problem. The approach they propose follows the decoupling approach described here, viz., the topology design is separated from the capacity and flow assignment problems.

The capacity and flow assignment part of their work is described here. They use a straightforward approach to determine how traffic is carried on the network in their design. They decouple the capacity determination from the flow assignment by obtaining a simple relationship between the flow carried on a link and the link capacity: they introduce upper bounds on utilisation for links and nodes. These upper bounds can be related to the delay constraints. They show that the optimal solution is obtained when the load on the nodes and links is equal to this upper bound. Hence a relationship can be obtained between the link capacities and the flow carried on each link. Once this is established, the link capacities are no longer free variables and they focus on the routing problem.

To solve the routing problem, they assign a cost to each of the candidate paths for each demand. The lowest cost path is then chosen to route the demand. The cost of each path is simply a sum of the variable costs of each link and node that constitutes the path. If the node switching costs are ignored, then the cost of the path is simply the variable costs associated with each link. This is equivalent to obtaining the shortest path using the variable costs associated with each link as the link weights and using this to route the demand and is exactly the same approach as that used by McGibney in his work.

Determining a Good Network Topology

Choosing a good network topology in the context of packet-switched networks is very similar to choosing a good network topology in the context of other network design problems. The approaches described above – the branch-exchange approach in particular – can be applied here. Indeed, this is what the authors of the articles mentioned above propose. However, the approaches differ in this case in that packet-related information from the capacity and flow assignment problem can be used to influence the topology selection process.

Gerla and Kleinrock propose branch-exchange algorithms for their problem in [GK77]. They discuss straightforward greedy approaches as well as more complex branch-exchange approaches based on cut-sets. In the cut-set approaches, instead of identifying all possible link topologies that could be used for the next iteration, a cut-set is identified which contains the most saturated set of links that connects two separate parts of the network. Saturation is measured using packet-level characteristics of the problem. An extra link is added across this cut-set, while a lightly loaded link is removed from another part of the network.

Gersht and Wiehmayer propose a so-called link reduction algorithm to iterate between different network topologies in [GW90]. In their approach, they start with a highly connected graph and proceed to remove links until removal of any more links causes either an increase in network cost or a resulting network configuration that does not meet the some of the design constraints. They propose a greedy approach in which links are removed that correspond to the maximal reduction in network cost.

Approaches in which the Topology Design Problem is not Decoupled

Two approaches have been proposed in which the topology design problem is not decoupled from the capacity and flow assignment problem. The first is the MENTOR approach, which is described in section 2.2.2 above. This can be used in the context of packet-switched network design as well as other network design types. In essence, the one aspect of the network MENTOR design procedure that is network specific is the loading on each link in the network. Recall that a direct link is installed between two nodes if there is sufficient demand. In a packet switched network design problem this can be determined if the delay on the link exceeds some threshold.

An alternative approach in which the topology design problem is not decoupled from the flow and capacity assignment problem is discussed in Gerla and Kleinrock [GK77]. This is similar to the work described by Yaged discussed in section 2.2.3 above. The flow deviation approach that they propose to solve the flow and capacity assignment problem naturally aggregates traffic when possible. This has the effect of assigning no flow (and hence no capacity) to many links in the problem. Hence, if this algorithm operates on a fully connected topology, the result is often a network which is quite sparsely connected. In this way a reasonable topology is a natural output of the use of this algorithm. Note that this approach is only applicable to situations in which the link cost function is concave. Consequently, it cannot be generally applied. However, there are a substantial number of cases in which the link cost is either concave or can be approximated by a concave cost function.

2.3.3 Logical Network Design

Lee and Yee consider a logical packet-switched network design problem in [LY91,LY95]. In this problem, the objective is to determine a logical network configuration that minimises the overall network delay.

Lee and Yee assume that the physical network consists of a set of given connections that are easily divided into channels. They consider interconnects such as T1 interconnects that can be divided into 64kb/s channels. These channels offer the flexibility to implement a logical network and the problem is then how to configure the channels to obtain a logical network that minimises the delay over the network.

Lee and Yee formulate the problem as a convex optimisation problem with linear constraints and hence a unique local optimum exists. The constraints on the problem are all linear. The number of variables in the problem becomes very large very quickly and consequently, the problem becomes difficult to solve quickly.

Lee and Yee propose the use of a partial branch and bound procedure to solve the problem. First, the relaxed form of the problem is solved. This is used as a starting point from which feasible solutions can be obtained. d paths are then chosen from the optimal solution and a number of modified problems are formed in which there are extra constraints on these paths. For each path two extra constraints are added – one in which the capacity of the path exceeds the value obtained in the initial solution and one in which the capacity is upper bounded by this value. This results in 2^d new problems. A relaxed version of each of these problems is solved and if the relaxed solution is of lower cost than the current minimum, rounding is performed to obtain a feasible solution. If the feasible solution is of lower cost than the current minimum, it replaces the current minimum cost solution. This procedure continues until a low cost, feasible solution is obtained.

The approach to choosing paths is done on a "longest and shortest first" basis: paths requiring small amounts of resources from many physical links are chosen first. The algorithm proceeds until the shortest and fattest paths are being considered.

2.4 Circuit-switched Network Design

Circuit-switched networks have been well studied over the last few decades. Advances in routing techniques made possible by the development of stored-program-control switches introduced new complexities and challenges to the analysis of circuit-switched networks. Circuit-switched network design has never been a trivial problem, even for networks employing simple hierarchical routing. Network design for networks using advanced routing techniques is a difficult problem and consequently has received considerable attention over the last couple of decades.

The circuit-switched network design problem differs from the facility and packet-switched network design problems in that the focus is on circuit level performance. The grade of service measure is the connection blocking probability and the demands are typically measured in call arrival rates. This makes the analysis of the problem quite different to that of the earlier problems.

This section is divided into four subsections. First, routing in circuit-switched networks is discussed. This has a very considerable impact on network performance, and hence an effect on network design approaches. Different network routing mechanisms are described. Next, a number of different algorithms proposed to solve design problems for circuit-switched networks using specific routing are described. These algorithms are quite complex and warrant substantial discussion. Approaches to designing logical networks for circuit-switched networking are discussed next: this is followed by a discussion on multirate circuit-switched networking and an approach to design such networks is described.

2.4.1 Routing in Circuit-switched Networks

Many different routing schemes have been proposed to improve the efficiency of circuit-switched networks. These range in complexity from the simple dynamic routing to more complex adaptive schemes. Four routing schemes are described here. Three of these have been used in operational environments; the fourth is useful for modelling other more realistic routing schemes and is included for this reason. The most important ideas in advanced circuit-switched routing are encapsulated in these routing schemes.

Dynamic Non-Hierarchical Routing

Dynamic Non-Hierarchical Routing (DNHR) was originally proposed in [AKK81] and discussed in a network design context in the seminal paper by Ash, Cardwell and Murray [ACM81]. The purpose of DNHR was to increase the efficiency of AT&T's network through the introduction of two important developments in telephony networks: the introduction of a dynamic aspect to network routing and a relaxation of the strict hierarchical rules that governed routing heretofore.

A specific implementation was developed for the AT&T network, but the concept is slightly more general than this. The DNHR concept allows for complex routing trees and associated parameters to be varied throughout the day: the routing trees themselves can vary and/or the associated parameters. In particular, parameters such as route weighting parameters may vary. One example of a routing scheme that would fall under the general class of dynamic non-hierarchical routing would be one in which a number of routes exist for each destination and one of these routes is chosen at random for each call. The probability of choosing each route varies throughout the day according to some predefined schedule. Other variants are possible. The particular variant of DNHR chosen to be implemented by AT&T is often referred to as DNHR. Below, DNHR will be used in this sense.

In AT&T's DNHR, a set of routes exists in each node for calls to each destination. These routes are tried in order. If the first route fails, then the second route is tried etc. until all possibilities are exhausted, in which case the call is blocked. In this scenario, *crankback* must exist in the network – i.e. if the call routing fails somewhere on the route other than the originating node, then the call must be 'cranked-back' to the originating node where another route will be tried.

DNHR is dynamic in the sense that the routing tables are updated periodically throughout the day. A number of routing tables exist in each node for each destination. These are pre-determined and the routing table that is in use at a particular time is dependent on the time of day. This enables the network to cater for different traffic patterns throughout the day. This is particularly beneficial in a network that spans a number of time zones such as the AT&T network that the scheme was designed for.

It is important to note that this scheme is not adaptive to the state of the network; the node behaviour varies throughout the day with the changing traffic patterns, but this is pre-programmed behaviour – the nodes do not adapt their behaviour to the state of the network.

Load Sharing and Alternate Routing

In *Load Sharing and Alternate Routing* (LSAR) a fixed set of routes exists for the traffic between the source and the destination nodes. When a call needs to be routed, one of these routes is chosen at random to route the call. If the call cannot be routed successfully, then an alternate route is chosen. This can again be chosen at random. Like the DNHR scheme described above, this scheme requires crankback to operate.

This approach has the effect of splitting the demand for a particular node-pair over a number of routes in proportion to the weights associated with each route.

This type of network routing is not used in real networks. However, networks using other routing schemes can be modelled by this routing scheme under appropriate assumptions. For this reason, this short description of LSAR is included here.

Residual Capacity Adaptive Routing

Residual Capacity Adaptive Routing (RCAR) is a general term to describe a number of different algorithms which are adaptive to network conditions and use (predicted) residual capacity on routes when determining how to route a call. Link status information is relayed via the signalling network in order to facilitate determination of lightly loaded routes.

In the two implementations described here a central processor periodically collates link status information, processes the data and sends updated routing information to the nodes as necessary. The central processor examines the variation in link utilisation during the time interval and uses this to predict link occupancy during the next time interval. This information is then used to determine lightly loaded routes and to configure routing functions in nodes.

It is important to note that the network status information is only updated at certain intervals – the nodes do not always know the states of the network links; there can be a discrepancy between the nodal information and the status of the links.

A particular scheme that falls into the general category of RCAR routing schemes is called *Dynamic Call Routing* (DCR) [Gir90]. This was implemented and trialled in Canada. In DCR, a fixed set of routes exists for routing traffic between two nodes. All of these routes contain two links. The call is first offered to the direct route. If this is full, then an alternate route is chosen with some probability. These probabilities are proportional to the amount of forecast free capacity on the route and are updated by the central processor. If the call is blocked on the alternate route, then it is lost.

Updating the probabilities forms the crux of the routing scheme, and it was found that the frequency of the updates was a very important factor in the performance of the scheme.

Another approach that falls into the category of RCAR is *Trunk Status Map Routing* proposed by AT&T [Ash85]. Like the DCR scheme, this approach uses a central processor to collate information on the status of the links in the network. The central processor analyses the link status information and, if necessary, effects changes in the routing in the nodes. In this scheme, it was found that the best routing strategy was to first try the route proposed by non-adaptive DNHR, and, if this fails, attempt to route the call on the least loaded alternate path.

Real Time Network Routing

Real Time Network Routing (RTNR) is another approach that was developed by AT&T [ACFH91] to improve the efficiency of their trunk network. This approach is very adaptive to the state of the network and is, in principle, quite a simple approach. The approach uses signalling between the source and destination nodes to determine the most lightly loaded route between the source and destination. It does not require the use of a centralised processor collating link status information to determine lightly loaded paths.

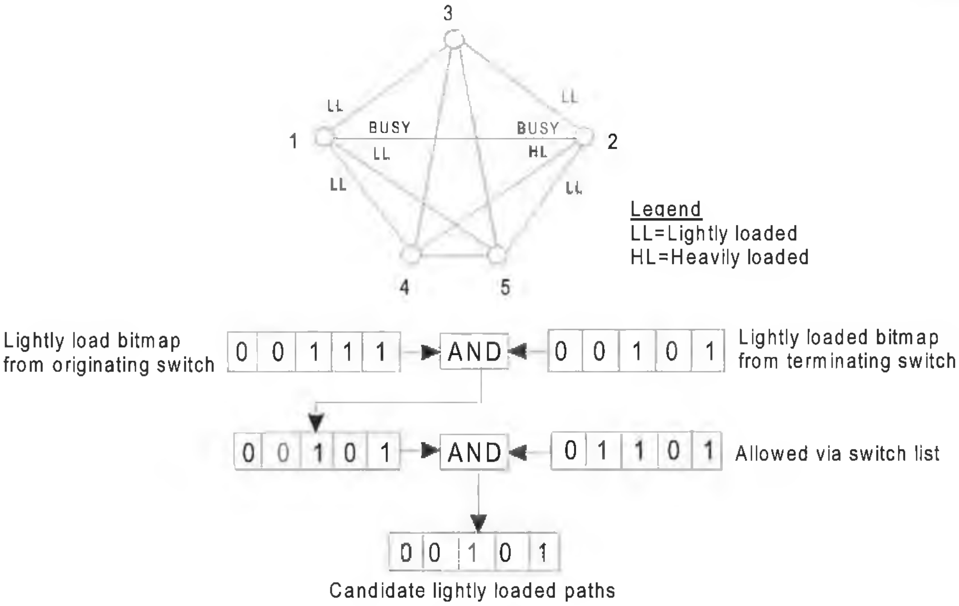


Figure 2-5: Operation of RTNR – determination of lightly loaded paths.

In this approach, one of six states is associated with each link at any time – Lightly Loaded 1 (LL1), LL2, LL3, Heavily Loaded (HL), Reserved and Busy. When a call is to be routed on the network, the direct route is first tried if it exists. If the direct link is congested, a message is sent to the terminating switch requesting its link status information. The link status information is returned to the originating node via the signalling network. The link status information takes the form of 6 bit maps – one for each link state – that can be used to determine the status of each of the links connected to the terminating node. The originating switch then performs a bitwise AND operation using the local link information and the link information obtained from the destination node to determine the most lightly loaded alternate route for the call. A check is then performed to see if the routing scheme permits this route to be used as an alternate route for this node pair: the routing scheme supports a mechanism to restrict particular routes being used for alternate traffic between node pairs. The determination of a lightly loaded path is illustrated in Figure 2-5. If a number of routes are in the same state, then a round robin approach is used to distribute the calls evenly between the different routes.

2.4.2 Circuit-switched Network Dimensioning

Since the routing scheme used in the network has a considerable impact on the network performance, different dimensioning algorithms were proposed for networks employing different routing schemes. These different network dimensioning approaches for the different network routing schemes are discussed here. In these dimensioning problems, the network topology is assumed to be fixed and given: the topology problem is considered to be a separate problem.

Dimensioning DNHR Networks

Ash, Cardwell and Murray developed the *Unified Algorithm* (UA) for the dimensioning of DNHR networks, and it is discussed in some detail in [ACM81]. The algorithm consists of a set of heuristics that can be used to dimension a circuit-switched network that uses DNHR routing to accommodate a set of given demands with some specified grade of service at a low cost.

Two variations of the algorithm are discussed: the *route formulation* and the *path formulation*. In the route formulation of the algorithm a set of routing tables are specified as inputs to the optimisation procedure, and one element of the design problem is to determine how to assign weights to these routing tables to obtain a routing that results in a low cost network design. In the path formulation of the algorithm, paths for demands are specified as problem inputs. The design algorithm then determines how demands are split over the paths. This differs slightly from the route formulation in that there is still the outstanding problem of how to choose the routing table parameters to realise the desired flows.

The path formulation of the problem is more flexible, since the constraints it imposes on the routing are less restrictive than those of the route formulation. Also, the authors found that the path formulation of the problem usually finds lower cost solutions. This is because the route formulation requires routing tables to be specified at the input to the problem – if bad routing tables are input then it is difficult to obtain an efficient routing of demands. Hence it is difficult to obtain an efficient network operation and good network performance. Consequently, the network required to meet the performance constraints is more costly. The path formulation is much less likely to have unsuitable routing tables since choosing the routing tables is part of the problem. For these reasons, only the path formulation of the UA is considered here.

The path formulation of the UA is an iterative procedure (as is the route formulation) in which quantities such as the routing of the demands, the link blocking probabilities and the link dimensions are repeatedly recalculated until they converge. A flowchart illustrating the algorithm is shown in Figure 2-6.

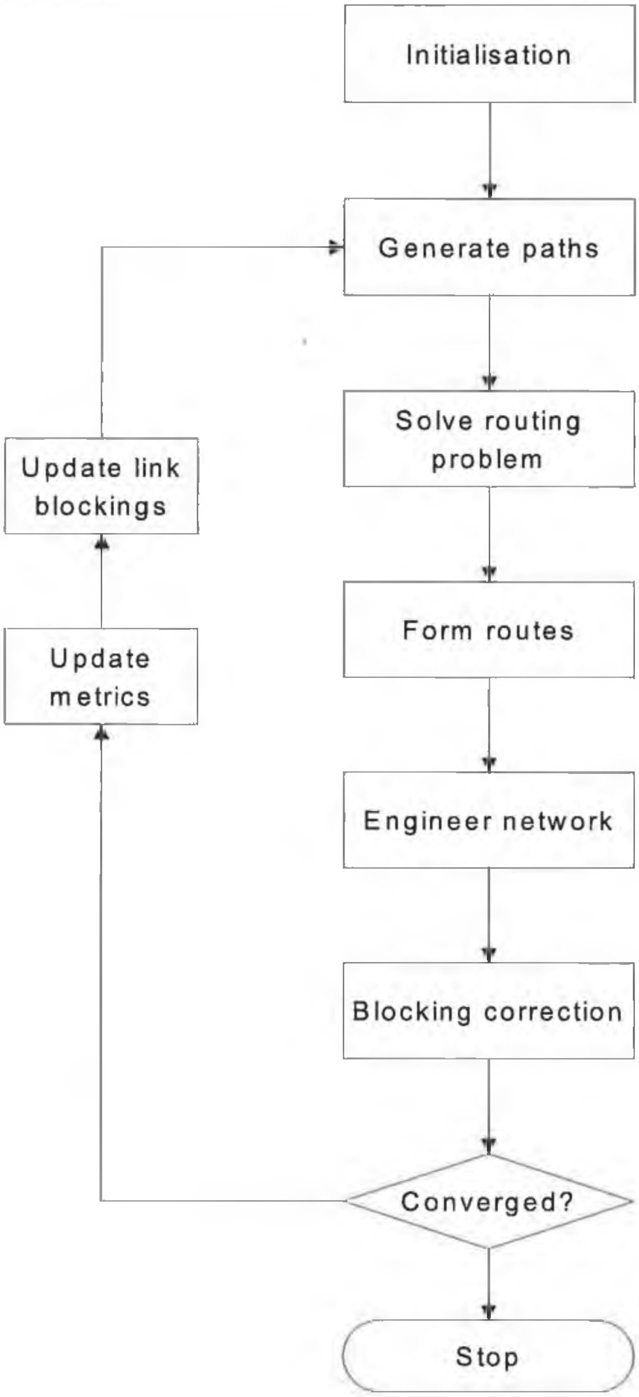


Figure 2-6: Flowchart for Path Formulation of Unified Algorithm.

The iterative process contains the following steps:

1. Generate paths;
2. Obtain an optimal routing of the demands by solving a route optimisation problem;
3. Dimension the links using the routing obtained from the previous step;
4. Perform blocking correction: dimensioning the links can cause the link blocking parameters to be changed. In some cases, the link blocking may exceed the desired

link blocking for some busy hours. Small adjustments to the link capacities may be necessary to ensure that link blocking does not exceed the required blocking.

If the process has not converged, then a further two steps are performed:

1. Update the link metrics;

2. Update the link blocking probabilities.

Each of these steps is discussed in more detail.

The first step is to generate a set of paths. Short paths are preferred since these require the least amount of resources: only direct and two-link paths are considered. In a large highly connected network, there can be very many two-link paths. Hence, some way of reducing the number of paths to some reasonable amount of candidate paths is required. This is not discussed in [ACM81], but some reasonable path generation heuristics can be envisaged based on some combination of shortest geographical distance, trunks with largest demand and possibly non-coincidence of busy-hours.

Once the paths are determined, the problem is to determine how the demands are routed on the network. The objective is to minimise the cost of the resources required to route the demands. A linear program is constructed to solve this problem. The variables in this problem are the amount of traffic carried on each of the links and the amount of traffic carried on different routes. The objective function is the sum of the marginal link costs and this must be minimised by solving the linear program. The output of the linear program is the set of traffic flows that should be carried on the network. This can be used to determine how much traffic is carried on each link.

The next step is to dimension the network using the routing obtained above. This is done in a reasonably straightforward manner. For each link, the maximum amount of flow through the link is determined over the different time periods. The maximum flow, together with the link blocking probability are used to determine the required link size. The Erlang-B blocking formula is used here to determine the required link size. The blocking for all hours is then determined using these link sizes.

The convergence test is then performed. If the parameters – routing, link blocking and link dimensions – have converged sufficiently closely, then the algorithm is terminated. Otherwise, the link blocking parameters are recalculated and the next iteration performed.

The link blocking parameters are recalculated based on the routing of the demands and the link dimensions. The amount of traffic incident at each link is easily calculated based on the routing. The cost of routing the link traffic is then formulated as the sum of the link costs and the cost of the traffic overflowing to other links. In general, the latter quantity is difficult to estimate and some approximation must be made. The resulting function is a non-linear function of the link dimension. This is then optimised with respect to the link dimension to obtain a new link dimension and a new link blocking parameter which is used as input to the next iteration of the process.

The link metrics, which are essentially derivatives of the link cost function with respect to the link dimensions are also recalculated. These are used in the objective function of routing optimisation procedure. Since these parameters are sensitive to the link dimensions, they are recalculated at each iteration.

This is the essence of the UA algorithm. There are particulars/details of the algorithm which are not so relevant to this discussion and hence they are omitted here. The algorithm is important since it is the first algorithm that was successfully applied to large-scale circuit-switched dimensioning problems.

Dimensioning LSAR Networks

Girard discusses modelling of LSAR networks in [Gir90]. One problem considered there is dimensioning of LSAR networks. The problem is to obtain a network design that results in a minimum cost network that can maximise the revenue generating capability of the network. The objective function is a sum of the network costs and a term reflecting the amount of revenue generated by the network. The problem also has some grade-of-service constraints: the blocking between node pairs must not exceed some pre-defined threshold.

In the dimensioning problem, there are two sets of parameters that need to be determined – the proportion of flow carried on each path and the link dimensions. The link blocking probabilities are implicitly defined once these parameters are defined; however, Girard chooses to introduce these as decision variables in the problem formulation and to introduce explicit constraints relating the load carried on each link, the link dimensions and link blocking probabilities.

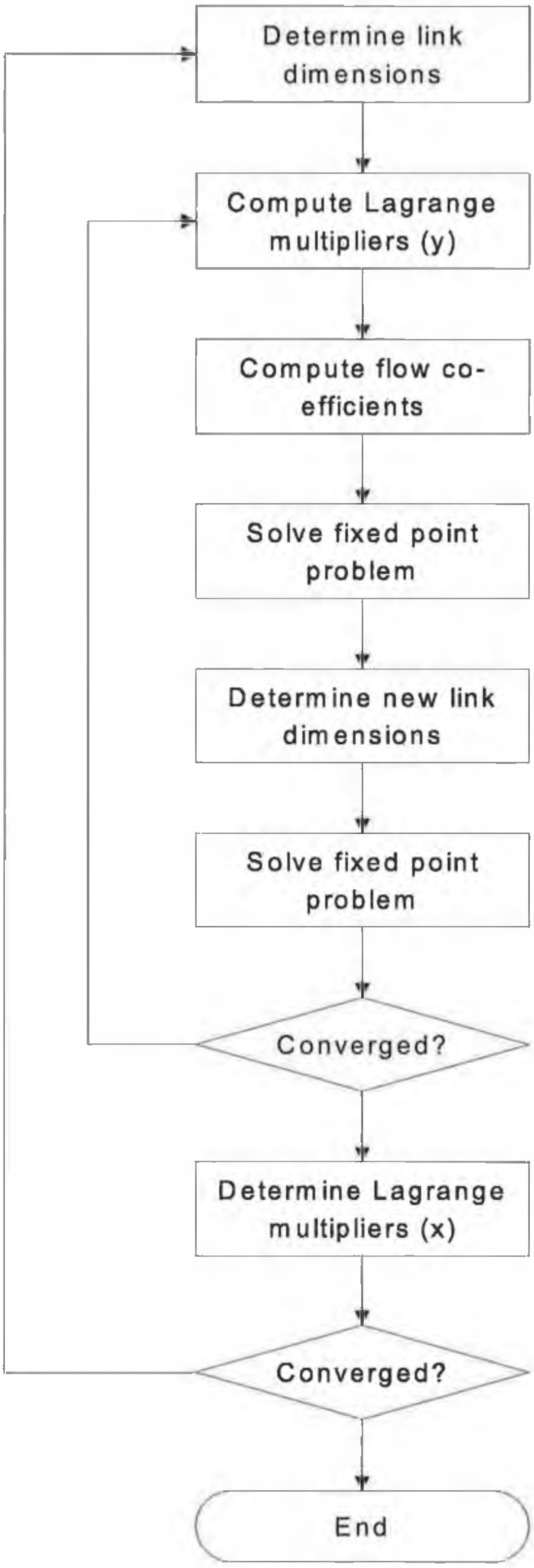

Determine link dimensions
Compute Lagrange multipliers (y)
Compute flow co-efficients
Solve fixed point problem
Determine new link dimensions
Solve fixed point problem
Converged?
Determine Lagrange multipliers (x)
Converged?
End

Figure 2-7: Flowchart for LSAR dimensioning algorithm.

The resulting problem is a large constrained non-linear programming problem. Girard forms a Lagrangian function incorporating the objective function and the problem constraints. Relations between the Lagrange multipliers and the optimal values of the problem decision variables can then be determined by taking appropriate derivatives of the Lagrangian function. A system of equations relating all of the unknown parameters results. This system of equations, however, is large and difficult to solve.

Girard proposes a heuristic iterative approach to solve the problem in which some variables are fixed and the remaining ones calculated until the system converges to some solution. This iterative approach is illustrated in Figure 2-7.

A detailed description of the algorithm is not necessary here. However, a few comments are in order to give some idea of the complexity of the algorithm. The link dimensioning problem can be decoupled into individual link dimensioning problems in this formulation. The link dimensions can then be obtained using a univariate optimisation technique. The y Lagrange multipliers can be determined by solving a linear program. The flow coefficients can be determined using a multicommodity flow solver. This part of the algorithm is the most time consuming. The fixed point problems can be solved by a repeated substitution method. Finally, the x Lagrange multipliers can be obtained by minimising the Lagrangian function with all other variables fixed.

The overall procedure does converge and it does converge to a local minimum of the LSAR dimensioning problem. However, the algorithm is quite complex and does require considerable amount of processing time. Girard and Liau [GL93] report that the solution algorithm takes of the order of hours on their systems for a 50 node network. This would probably take of the order of 10's of minutes on today's computer systems. However, the algorithm doesn't scale up well.

An interesting point relating to this work is that the optimisation algorithm used has its roots in a mathematical model. This is in contrast to the UA, which is simply a set of heuristics, albeit quite reasonable ones. The mathematical model is then analysed to identify characteristics of the solution and speed up the solution algorithm. The fact that the optimisation algorithm is (in some sense) derived from a mathematical model also means that it is possible to consider how close to the optimal the derived solution is. Furthermore, since there is a definite mathematical model to solve, different optimisation techniques can be applied to obtain some solution.

Dimensioning RCAR Networks

The design of RCAR networks – in particular the DCR variant of RCAR – is discussed in [GL93]. The basic idea behind this approach is to approximate a DCR network by an

LSAR network and use the dimensioning procedure for LSAR. Girard and Bell used the same idea in [GB89] in an attempt to devise a fast algorithm to determine the network performance of a DCR network.

A DCR network can be approximated by an LSAR network by noting that over the long term, the behaviour of the DCR network appears stationary. The effects of updating the routing periodically in response to changes in the demand disappear when the network is observed over a long period of time. Hence, the network can be modelled as an LSAR network. However, care must be taken when doing this. Since the routing is periodically updated in DCR, the weights for each route are not arbitrarily chosen, unlike LSAR. When modelling DCR using LSAR, extra constraints are introduced that relate the weights for each route to the amount of free capacity on each route. Girard and Bell showed that DCR can be reasonably well modelled using LSAR using this approach [GB89].

The DCR network is then dimensioned using the LSAR dimensioning procedure and the relation coupling the free capacity on each route to the weight for each route. This is not quite as straightforward as it may at first appear. In the LSAR dimensioning problem, the amount of flow offered to each route was a design variable. In this problem, the amount of flow offered to each route is no longer a free design variable since it is dependent on the occupancy of each route.

The approach used to solve this problem is also an iterative one. A flowchart illustrating the approach is shown in Figure 2-8.

The design approach does have some stability concerns. It is not clear that the approach will converge to a solution. In the DCR design algorithm, there are three loops that must converge – the flow variables, the link dimensions and the subgradient optimisation loop. The flow variables loop converges well. The link dimensioning loop, however, does not converge quite so well, and some measures are taken to dampen the variation of the resulting link dimensions. The link dimensions are weighted sums of the link dimensions obtained during the current iteration using the dimensioning rule and the link dimensions used in the previous iteration. Similarly, the x parameters can exhibit non-convergent behaviour. In the same way, dampening mechanisms are used to ensure that these do indeed converge.

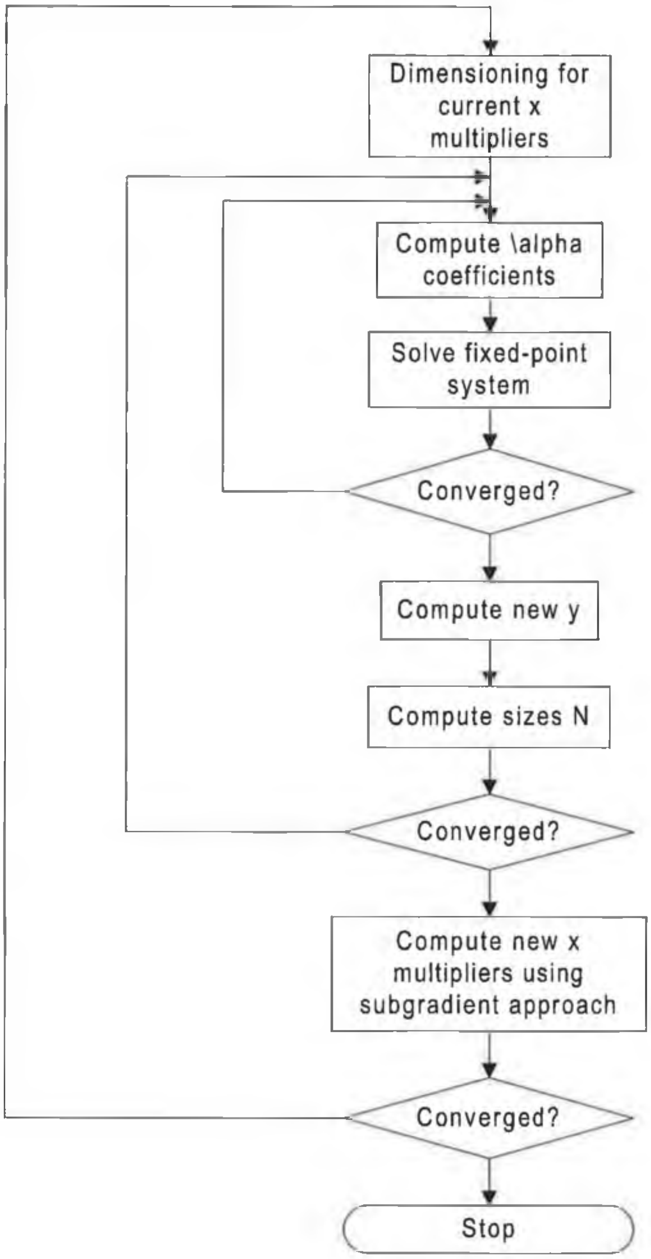

Dimensioning for current x multipliers
Compute \alpha coefficients
Solve fixed-point system
Converged?
Compute new y
Compute sizes N
Converged?
Compute new x multipliers using subgradient approach
Converged?
Stop

Figure 2-8: Flowchart for DCR dimensioning scheme [GL93].

One interesting observation made by the Girard and Liau is that they find that the network cost resulting from the DCR network design approach is typically higher than that resulting from the LSAR approach.

Dimensioning RTNR Networks

In [ACL94] the dimensioning of RTNR networks is discussed in the context of a logical network design problem. The overall approach is applicable to facility network design, but considers the inclusion of RTNR traffic on the facility network. To this end, some discussion of the dimensioning problem for RTNR networks is discussed.

The RTNR dimensioning algorithm described in [ACL94] is based on the RTNR network performance model developed in [AH93]. In the dimensioning problem, it is assumed that some grade-of-service performance objectives, e.g. blocking probabilities, are given and that the resulting network is fully connected. The objective is to obtain a set of link dimensions that enable the performance objectives to be just met.

The objective does not explicitly involve obtaining a minimum cost network since this work was done in the context of a logical network design problem, although the network dimensioning algorithm should not result in link dimensions that are unnecessarily high. Once some set of interswitch link capacities are obtained that meet the grade-of-service constraints, these can then be input to the topology design problem. The real cost considerations come into effect in the link dimensioning problem. The link dimensioning problem is formulated as a linear programming problem in which the objective is to minimise the extra capacity required to meet the demands. The design model incorporates reliability requirements by ensuring some fraction of each demand is routed over a physically diverse path.

The dimensioning algorithm is an iterative heuristic dimensioning algorithm. It is illustrated in Figure 2-9. The algorithm uses three separate models developed in the performance model of [AH93]. The link state probability model is used to determine the probability that the link is in each of the different link states – LL1, LL2, LL3, HL, Reserved, Busy – as discussed above. In the traffic flow model, the link state probabilities are used to obtain the route state probabilities. These, in turn, are used to determine what proportion of flow pertaining to each demand is carried on each route. These flow variables are the output of the link flow model. The third aspect of the performance model is the adaptive trunk reservation model. This models the dynamics of the trunk reservation level for each link and attempts to determine the mean trunk reservation level. This is dependent on the node-to-node blocking and is used in the link state probability model.

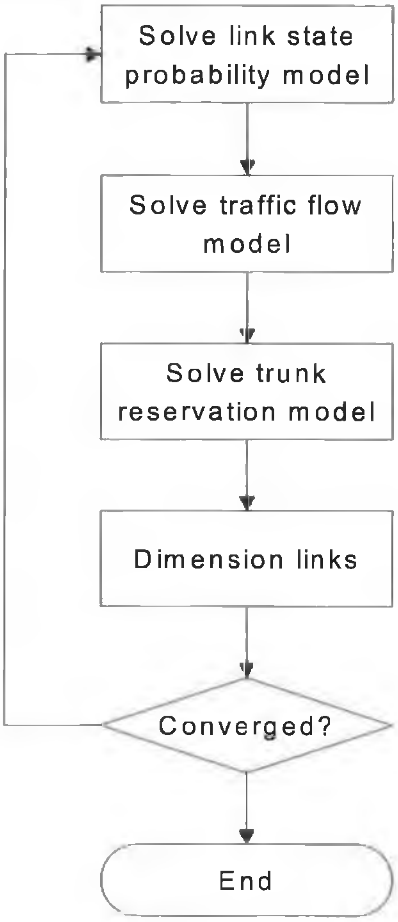

Solve link state probability model
Solve traffic flow model
Solve trunk reservation model
Dimension links
Converged?
End

Figure 2-9: The dimensioning algorithm for RTNR networks.

The link dimensioning method is not described, although it is noted that the links are dimensioned in terms of T1 transmission capacities. Consequently, there is considerable granularity in the dimensioning of the links and hence the link dimensioning problem should not prove very difficult. The flow incident at each link is an input to the link dimensioning problem and this should be used with an inverse Erlang B formula to obtain dimensions for the links.

Once the dimensions of the logical network design are determined, these can be input to the logical network design optimisation step. The design of logical networks is discussed separately in the following section.

2.4.3 Logical Network Design for Circuit-switched Networks

Development of more sophisticated cross-connect equipment in the 80's added extra flexibility to the configuration of telecommunications networks. This enabled **logical** networks to be implemented over physical infrastructures in circuit-switched networks (see [Ash95] and [AS90] for a more detailed exposition of this concept). This extra flexibility can enable more efficient use of the network resources. This concept was discussed above in the context of packet-switched networks, and will be discussed below in the context of ATM networks.

Three variants of the logical network design problem are discussed here. In the first, the objective is to determine how to configure a logical network on some given physical resources such that the logical network can best accommodate the demands. In the second problem, the objective is to determine a logical network on some physical network which minimises the amount of extra capacity that must be added to the physical network. Finally, a discussion of a combined logical and physical network design and configuration problem is discussed.

The problem that arises then is how to configure the logical network using the available physical resources so as to maximise the network efficiency. Other variants of logical network design problems are possible and arise in other networking contexts, but this is the only one that is discussed here.

Gopal, Kim and Weinrib studied the logical network dimensioning problem for circuit-switched networks in [GKW90,GKW91]. In their problem formulation, each node-pair was offered a certain amount of Poisson traffic. The problem was to maximise the amount of traffic carried by the network, and hence the revenue generated by the network, by appropriately configuring the logical network. There are no specific constraints on the problem. In particular, there are no constraints on the blocking between node pairs – this can be arbitrarily large in the solution.

They propose a greedy algorithm to solve the problem. For each demand and each available path, some benefit is calculated for carrying an extra unit of the demand on the path. The benefit function is a sum of the resulting increase in the amount of carried traffic and some measure of the amount of traffic that cannot use these resources if they are assigned to this path.

The algorithm used by the authors does not assume any advanced routing strategies in the network. Hence the increase in carried traffic that results from the addition of capacity to a path can be determined using the Erlang-B function.

The authors find that this direct approach to solving the problem can result in networks that operate more efficiently than do networks with advanced routing strategies. The performance of a physical network with logical overlay network is compared to that of a physical network with least loaded routing when the load differs from that of the design load for the physical network. The results show that the extra layer of flexibility can result in better network performance. This is especially true of cases in which the incident load differs considerably from the design load. Indeed, the greater the difference between the design load and the actual load, the better the performance obtained using the logical network configuration.

Ash, Chen and Labourdette discuss such logical network configuration in [ACL94]. They consider the problem in a more realistic context than that considered by Gopal et al. The problem that they consider is somewhat different from that of Gopal et al. Instead of trying to determine a logical network which can carry the maximum amount of traffic on the given physical network, they attempt to carry a specific set of demands on a given physical network in such a way as to minimise the amount of capacity that needs to be added to the existing network. Since the demands are specified in terms of concrete capacities rather than call arrival rates, this problem is closer to the facility network design problems of section 2.2.

Ash, Chen and Labourdette formulate the logical network design problem as a linear programming problem. The problem can be solved using some standard approach but Ash et al propose an algorithm to obtain some solution to the problem which uses specific domain knowledge to reduce the time required to find a solution.

Medhi also considers the logical network design problem in [Med94]. He considers this problem in association with the physical network design problem. He assumes some fixed physical topology is specified and the problem is then to determine the physical network capacities and the logical network configurations that result in the minimum cost network that can accommodate the constraints. Medhi considers two types of constraints: survivability constraints and grade-of-service constraints. He also considers multiple sets of demands – one for each different time period. The problem he considers is quite comprehensive.

Medhi formulates the problem as a linear program. However, the very large number of variables in the problem make it difficult to solve. Consequently, some approaches to simplify the problem and obtain some solutions are discussed.

Medhi eliminates the circuit-switching characteristics of the problem at a high level by determining the required amount of trunks between each node-pair to deliver the grade-of-service constraint. This is done by assuming that direct routing is used throughout the network – if more complex routing is used, then it is considerably more difficult to determine the required capacity between each node pair. In this way, the demands are mapped from a set of loads measured in Erlangs to a set of capacities, which are much easier to deal with. This reduces the problem to one which is similar to the facility network design problem; there are no longer any specific circuit-switched aspects to the problem. Medhi solves the problem by decoupling the different failure scenarios and obtaining both a logical network configuration and a physical network configuration in each case. The physical link capacities chosen are the maximum capacities required over all failure scenarios. Details are omitted since the emphasis here is on design of logical circuit-switched networks.

2.4.4 Multirate Network Dimensioning

Medhi and Guptan consider the design of multiservice, multirate switching networks in [MG97]. In this network, calls can occupy more than one circuit. Different services can then be supported: services requiring more capacity than conventional voice telephony. The network can be used to carry traffic for, say, a videoconference, which could consist of two or more circuits. The problem they consider is how to dimension the network to accommodate the multiservice demands with some grade of service performance objectives.

In general, multirate problems are considerably more complex than those of single-service networks. This is because the characteristics of the different services supported by the multirate network can differ greatly. Different services can have different capacity requirements and different traffic patterns. For example, the arrival rates and the call durations may not be consistent with the Poisson model that approximates voice so well and is easy to work with analytically. Analysis and performance evaluation of multiservice networks is quite a difficult problem.

Medhi and Guptan approach the design problem by decoupling the problem in the same way as before. This time, however, the decoupling does not apply simply to a single service, but rather to the set of multiservice demands are mapped to a set of capacities. Each service is considered in isolation and the capacity requirements per node-pair per service are calculated. These are then aggregated to obtain per node-pair capacity requirements and these requirements are then provided as input to a mixed integer/linear programming solver to obtain some solution.

The capacity required per service per node-pair is determined using the traffic characteristics and the capacity requirements. Direct routing is assumed, so the complexities introduced by multiplexing many different traffics onto a single link can be avoided. In this particular case, Poisson traffic is assumed and the required number of connections is determined using the Erlang B formula. This is converted to a required capacity by multiplying by the capacity required per connection.

2.5 ATM Network Design

ATM was chosen as the technology of choice for B-ISDN and received considerable interest in the research community in the early-mid 90's. It was proposed as a technology with the ability to support many different services which may have differing requirements of the network. In particular, ATM made possible the notion that different services could have different QoS, and ATM was the first technology to offer proper QoS support for connections.

ATM is a cell-based technology: it exhibits some of the characteristics of both circuit-switched and packet-switched technologies. ATM transmission is cell-based transmission – data is transmitted in 53-byte cells. Consequently, packet delays, loss and jitter can be unpredictable. In this way, ATM is similar to packet-switched networking. On the other hand, when a request for service arrives, the network determines whether or not a route exists that has sufficient available resources to meet the QoS requirements of the traffic. This is similar to the behaviour of circuit-switched networking.

Much of the work reported on ATM design problems in the literature concentrates on the extra level of flexibility introduced by the *Virtual Path* (VP) concept. This enables traffic to be aggregated into some kind of virtual trunks in the network. Three possibilities exist for the network design problem:

1. the logical network design problem – the problem of determining the optimal logical network given some physical network;
2. the problem of designing an optimal physical network to accommodate some given logical network;
3. combined logical and physical network design.

These different problems are discussed below.

Since ATM exhibits characteristics of both circuit-switched and packet-switched networks, the emphasis in the design problem can be on either packet or connection level characteristics. In some earlier work, the emphasis was on packet level characteristics, but in later work, much more emphasis was on connection level measures. Each of these is discussed in the next section.

ATM services are currently on offer in the marketplace. However, it has not reached the levels of penetration that were imagined at its inception. The early proponents of ATM envisaged a network with ATM connections directly to end users and traffic carried from end-to-end over ATM. However, with the enormous growth in the use of IP and the high availability of IP interfaces, many now believe that an IP-based infrastructure is a more likely candidate for multiservice networking.

Most current ATM service offerings are semi-permanent – as opposed to switched – service offerings. These could be used for high-speed *Local Area Network* (LAN) interconnections where, for example, users on the interconnected LANs use bandwidth intensive applications such as video.

The work described below is strongly influenced by the traditional telephony perspective rather than data communications perspective which is more dominant today. Hence, the authors think of connections being established and torn down via ATM signalling. This is not the state of the market today and it is unlikely that the market will evolve to this point. However, the techniques described here could be applied in IP-QoS networks, although they may need to be modified to some extent. Hence they are interesting.

Since ATM connections are cell-based, it is often not obvious how to determine whether there are sufficient resources to accept a new connection. The problem of determining whether a new connection can be accommodated without adversely affecting existing connections is the admission control problem. A short discussion of this problem is included here since it is a fundamental aspect of the operation of the network which has a profound impact on how the network resources are used and hence has an impact on how the network is dimensioned.

The admission control problem for ATM focuses on how to decide whether or not a particular connection can be carried on the network. Much work has been done on this problem and a recent overview of contributions in this area is reported in an article by Knightly and Shroff [KS99]. A number of different approaches have been proposed such as techniques based on average and peak rate combinatorics or maximum variance based approaches. One of the more interesting approaches is the so-called effective bandwidth approach in which the requirements of each connection are encapsulated in a figure of merit known as the effective bandwidth.

The effective bandwidth approach was originally proposed by Hui in [Hui88]. The concept is attractive because it means that admission control decisions can be easily made by comparing the available resources against the effective bandwidth. Alternatively, the amount of capacity required for a number of connections is simply the sum of the effective bandwidths. Using this concept then, cell-based ATM networks bear very striking similarities to circuit-switched networks, although in the ATM case, the 'circuits' can have quite arbitrary capacities. Some of the theory of circuit-switched networking – some of which is discussed in section 2.4 – can then be applied [Ros95] to the network design problem.

The remainder of the section is structured as follows. The logical network design problem is discussed next, followed by the physical network design problem. Then a way to solve a combined logical and physical network design problem is discussed.

2.5.1 Logical Network Design

The logical network design problem focuses on dimensioning the VP logical network to meet some performance criteria. Three classes of logical network design problem exist in the literature:

1. generic problems;
2. those in which the emphasis is on packet level issues ;
3. those in which the emphasis is on connection level issues.

The generic problems are ones in which the context of the VP is not considered: the input to the problem is the set of VP demands. The packet level problem is one in which the objective is to determine the logical network configuration that minimises some packet level characteristic. Similarly, in the connection level problem the objective is to determine the logical network configuration that minimises some call level characteristic. Each of these is discussed here.

Generic Logical Network Design

The problem in this case is to route a set of VP demands on a network. The VP demands are given as an input and the objective is to determine the optimal network configuration. This problem was studied by Chlamtac et al in [CFZ94]. There, the objective was to determine a route configuration for which the maximum load on the physical links was minimised. This results in a network in which the load on the network is balanced and the network is most resilient to failure.

Chlamtac et al proposed an algorithm in which the path for each demand is chosen at random from the set of shortest paths for each of the demands. They demonstrate that the solution obtained using this simple approach is provably close to the optimal in a large network.

Optimisation of Packet Level Characteristics

Gerla, Monteiro and Pazos [GMP89] consider a logical packet-switched network design problem with ATM in mind. The problem they wish to solve is to determine the logical network that optimises some packet level characteristics given some physical network. In the problem, the demands are point-to-point demands specified in terms of a set of packet generation rates. Arbitrary paths are possible for the demands on the network: the problem is to determine the set of VPs that optimises some function of the overall packet delays. The set of VPs are defined by determining the routing and the dimensions of the VPs.

They observe that the link rates are typically very high in ATM networks and hence the nodal delays are usually very small. Since these delays are small, a very accurate model would be required to capture the characteristics of the node delays. Such an accurate model is difficult to generate. For this reason, the authors focus on issues associated with packet loss. They note that for some given set of node buffer sizes, the packet losses are reduced if the overall packet delays are reduced. Consequently, even though minimisation of the packet delay is not their primary objective, the use of a delay function achieves their objective which is minimisation of packet loss.

To solve the problem, some node queuing model is required. They consider an M/M/1 node delay model. They note that the resulting solution of the topology and routing problem is quite insensitive to the shape of the delay versus trunk load curve and hence, the exact nature of the node delay model is not very important. Having argued that the network can be modelled by a set of M/M/1 queues, much of the theory developed by Gerla and Kleinrock in their earlier work can be applied. In particular, the overall network delay expected in a network of M/M/1 queues can be used in this situation.

First, a scenario in which the initial logical network topology is given is examined. A flow and bandwidth allocation problem is then formulated. The decision variables in this problem are the set of logical link capacities and the logical link flows. The problem is formulated as an optimisation problem with a concave objective function and a set of linear constraints. The authors suggest the use of a flow deviation approach to solve the problem.

The problem has a number of locally optimal solutions. The steepest descent algorithm finds one such solution, but the solution found is very dependent on the choice of starting point for the steepest descent algorithm. In this case, a common approach to obtain a good solution is to perform the optimisation a number of times using different starting points and choose the best overall solution.

The authors then consider their choice of an initial starting point. In obtaining a good starting point, they first consider how to determine the topology of the logical network. They identify those node pairs that have the highest traffic demands: these node pairs then have a direct logical link. Nodes with smaller inter-node traffic demands have no direct logical link. They then obtain a particular feasible solution by constructing another minimisation function, the objective of which is to minimise the total logical link capacities and flow. Solution of this yields some particular feasible solution – random variants of this are used as initial starting points.

Optimisation of Call Level Variables

Siebenhaar [Sie94] considers the logical network design problem in the context of an ATM network. He considers the call level version of the logical network design problem. In this case, the objective is to maximise the revenue generated by the network. The revenue is assumed to be a weighted sum of the mean amount of traffic carried on all of the VPs in the network.

Siebenhaar assumes that the routing in the network is fixed and that a number of VPs exist between each node pair. He assumes that a number of different VPs can be modelled as a single VP of capacity equal to the sum of the capacities of the individual VPs. He then assumes that the amount of blocking experienced for calls between the node pairs is well approximated by the blocking on the single aggregate VP. This assumption is reasonable for large networks.

The objective then is to determine the dimensions of the VPs in the network that maximises the overall revenue generated by the network.

The problem formulation that he arrives at has a non-linear objective function with some linear and some non-linear constraints. The non-linear constraints are constraints on the blocking on the VP: blocking on the VP is a non-linear function of the capacity of the VP. The non-linear constraints make the problem more difficult to solve. Siebenhaar simplifies the problem by imposing a lower bound on the size of the aggregate VP – it must be sufficiently large to ensure some minimum level of blocking. This is done by choosing some minimum blocking threshold and determining the VP capacity associated with this blocking threshold. This capacity can then act as a lower bound on the VP capacity and the chosen blocking threshold is a lower bound on the blocking on the VP. In this way, the non-linear constraint can be replaced with a linear constraint.

The above solution method can result in infeasible problem formulations. The blocking threshold can be made arbitrarily small, and an infeasible problem will arise. For example, a tiny blocking probability can be chosen such there is insufficient capacity in the entire network to meet the blocking constraint. To solve this problem, Siebenhaar introduces a so-called bisection method in which the blocking threshold is initially set, and if it results in an infeasible solution, then the blocking threshold is raised, the problem is solved again and the process is repeated until a solution is obtained that minimises the objective function as well as the maximum connection blocking probability. To obtain a solution for each problem with linear constraints, a reduced-gradient approach is used in conjunction with a quasi-Newton method.

The author examines the effect of modifying the arrival traffic. Different modifications are tried; both a skew modification in which the overall load remains the same, but is redistributed amongst the nodes and an overload modification in which the traffic in increased uniformly for all the node pairs. It was found that the network performed better with logical network optimisation in all circumstances than without logical network optimisation.

Arvidsson [Arv94a,Arv94b] also considered the logical network design problem in the context of ATM networks. He considers the problem of determining a logical network configuration given some arrival rates for different services and different origin-destination pairs. He proposes a simple greedy approach to solve the problem in which each service and origin-destination pair is considered in isolation. The approach is quite similar to that used by Gopal et al to solve a similar problem which arises in the context of circuit-switched networking in [GKW91] except that it is generalised to the ATM case.

The benefit associated with increasing the amount of traffic a particular VP can carry by one connection is calculated. The costs of augmenting the VP by a unit of capacity are determined and the benefit/loss ratio is determined. The VP with the highest benefit/loss ratio has its capacity increased, and this process continues iteratively until all the available capacity has been assigned. The benefit of using this method is that the cost function can be non-linear and the capacity can be assigned in arbitrary units.

2.5.2 Logical and Physical Network Design

Some authors have considered the problem of designing the logical and physical networks together. The objective in this case is to obtain a set of logical network connections and physical network connections such that the overall network cost is minimised possibly subject to some performance constraints. This differs from the above problem in that the physical link capacities are now decision variables rather than given parameters.

Medhi [Med95] considers the complex problem of dimensioning a multi-hour multi-service network. The input to the problem is a set of nodes and traffic requirements, and a possible physical network topology. The problem is to determine the set of link capacities that result in the minimum overall cost. In this case, the cost is assumed to be linear in terms of capacity. One output of the solution process is the set of logical network topologies that can be used in the resulting physical network to deliver the desired performance.

This overall network design problem is very difficult and must be simplified. The general approach used by Medhi is a kind of decoupling approach in which the connection-level demands are mapped to a set of VP capacities. The problem is then to determine how these should be routed on the physical network. The call-level demands are mapped to VP capacities by first using the grade-of-service constraints to determine the amount of calls that need to be carried on each VP in the network. Then, some model of connection characteristics is used to determine an equivalent capacity for the VP. The approach used in the paper uses the equivalent capacity techniques of Anick, Mitra and Sondhi [AMS82] that is based on two-state fluid flow models and some specified buffer overflow probability.

Having determined a set of VP capacities, the problem is then to determine the way in which the demands are routed for the different busy hours. This is formulated as an integer programming problem. Medhi uses a subgradient[3] optimisation approach to solve the problem. This approach was used to solve smaller problems in reasonable time on relatively old computing power. Specifically, the approach was used to solve a problem with 18 switching nodes and 23 cross-connect nodes in approximately 2 minutes.

One shortcoming of this approach is that the link cost functions are limited to linear cost functions and the solution method is designed specifically for linear link cost functions. Another limitation of this approach is that the network topology is specified as an input to the problem. If the network topology is not known, then this approach cannot be applied directly, although it could most probably be nested inside one of the algorithms used above to determine the network topology.

Rohne et al [RJSV98] propose an approach to design both the logical and physical networks. They take a more practical approach than that of Medhi. They develop a practical model for the costs associated with a particular physical and logical network configuration. The costs include transmission costs, switching costs and connection set-up costs. The objective is to minimise these three sets of costs while meeting the network performance objectives.

The problem inputs are then quite specific traffic data, specific nodal data and quite specific link availability data. The information is used in a heuristic approach to solve the network design problem. The output is a physical and logical network configuration.

The problem is solved in two steps: first, they solve the physical network design problem assuming that there is no segregation of traffic onto different VPs and then

[3] Subgradient optimisation is designed for situations in which the problem can be solved using a dual approach. In this scheme, the direction of ascent is given by the value of the constraints $g(x)$ at each iteration.

they try to lower costs by choosing a logical network topology in the given physical network.

A specific topology is assumed in the physical network design problem and some specific traffic routing is assumed. Given this, the approach to solve the problem is to iterate through all nodes and quantify the traffic on each link terminating at the node. This is then used to dimension the link subject to the performance constraints. This procedure is performed iteratively until the process converges to some solution. A dimensioned physical network results.

Once the physical network is designed, the authors attempt to reduce the cost of the network by introducing VPs. The VPs reduce the network costs by reducing the costs associated with call set-up and switching. As against this, the traffic segregation that typically occurs in VP networks can result in increased transmission capacity. Hence, the reduction in set-up and switching costs must be offset against the increase in transmission costs.

The VP network is designed by iterating through the VPs, determining the gain/loss incurred by cross-connecting the VP at the node. If a cost reduction results from cross-connecting a VP at a node, then this VP is cross-connected.

2.6 Layered Approaches to Network Design

One final approach to the network design problem that is very relevant to the work done here is a layered approach to the network design problem. This can be considered more of a framework in which to think about network design problems than a specific approach to solving a specific network design problem. In this framework, different layers within the network architecture are identified and these are designed separately, with the output of one design problem feeding into another design problem. Lubacz and Tomaszewksi [TL00] have developed an interesting framework in which to consider this approach. There, they envisage the network design to consist of several layers – switching network, logical network, physical network – and they consider each layer to have its own design problem. They then consider each of the network design problems in isolation.

Ash et al follow a somewhat similar layered approach in [ACL94]. Indeed, the notion of decoupling the different layers arises in many of the contributions discussed above, although in most cases, this is just a convenience that simplifies the problem. For example, Medhi [Med97] decouples the connection level problem from the facility network problem by mapping from the demands to a single capacity, although he doesn't explicitly think of this as constituting a layered approach to solve the problem.

Doshi and Havardshana [DH98] describe a systematic approach to network design in which this layering notion is clear. They consider large realistic network design problems in which the demands are many and varied. These demands are then mapped to capacities using some demand mapping modules and the capacities are then used as inputs to the facility network design problem. Different possibilities for the facility network design problem are then considered – the facility demands can be realised in mesh networks or ring networks or some combination of both. Again, this notion is quite close to that considered here. However, Doshi and Havardshana focus exclusively on network design problems in the context of public network design. Also, they think quite strongly in terms of determining concrete capacities for the demands as output of the demand modules.

These ideas are related to this work, although they do not think in quite the same way. Here, objective is to attempt to unify network optimisation problems: to highlight common characteristics of such problems and to exploit these common characteristics in a reusable way. In these approaches, the authors often think of dividing the network design problems and tackling each one using a very different approach.

2.7 Conclusion

Here, an overview of different network design problems and different solution approaches has been given. These range from early studies of facility network design problems to packet-switched network design problems to circuit-switched network design problems to ATM network design problems to more generic layered approaches to solve these problems. It is clear that there are many variations of network design problems. Hence some approaches to characterise network design problems and abstract some of the characteristics of network design problems may be beneficial. This is the central concept of this thesis and is explored in detail in the following chapters.

CHAPTER 3 A FLEXIBLE, ABSTRACT NETWORK OPTIMISATION FRAMEWORK

3.1 Introduction

Here, a flexible, abstract network optimisation framework is described that will be used below to solve some specific network optimisation problems. First, the utility of the network optimisation framework is motivated. Then, the framework itself is described. The framework is divided into layers and each of the layers are described. A key element of the framework is the generic problem which is an abstraction of the specific problems. This is described in the next section. The following section discusses the formal mathematical model of this generic problem. All aspects of the generic network design problem are then defined and examples of how it may be applied to types of specific problems are given. Finally, a high-level description of some algorithms that can be used to solve the generic problem is given.

3.2 Motivation for the Network Optimisation Framework

Typically, a network optimisation problem is solved using an approach consisting of the following steps:

1. Develop a model for the problem;
2. Develop a formal mathematical model from the problem model;
3. Devise an algorithm or employ some standard algorithm to obtain some solution to the formal mathematical problem.

By observing that many network optimisation problems are similar in nature – very many of the problems listed in the previous chapter focussed on determining how to route demands on a network – a generic network optimisation problem can be formulated that can be applied to many specific network design problems. The big advantage of such an abstraction is that it can be reused and hence can result in reducing the time and effort required to obtain a solution to the problem.

3.2.1 A High-level View of the Network Optimisation Framework

The network optimisation framework is illustrated in Figure 3-1. In this approach, the problem is decomposed into the following layers:

- the specific problem model layer;
- the mapping function;
- the generic problem model layer;
- the solution layer.

Each of these is discussed in more detail.

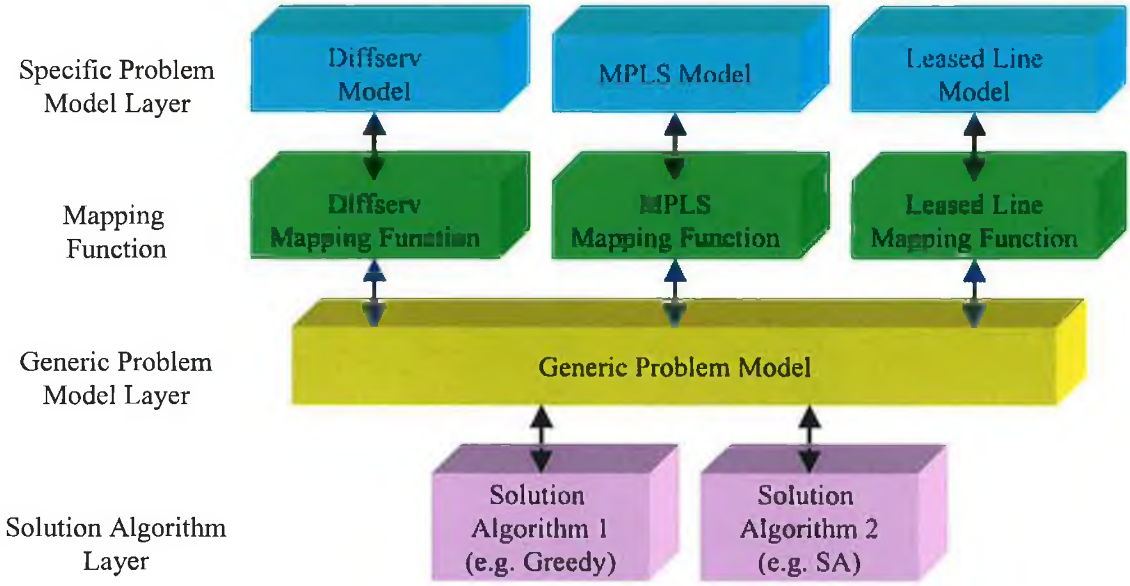

Specific Problem Model Layer
Diffserv Model
MPLS Model
Leased Line Model
Mapping Function
Diffserv Mapping Function
MPLS Mapping Function
Leased Line Mapping Function
Generic Problem Model Layer
Generic Problem Model
Solution Algorithm Layer
Solution Algorithm 1 (e.g. Greedy)
Solution Algorithm 2 (e.g. SA)

Figure 3-1: Illustration of the layered approach to solve the problem.

At the specific problem model layer, the **specific network optimisation problem** or **specific problem** is defined. This is defined in context-specific terms. For example, in an ATM network design problem, the specific network design problem could be defined in terms of the number and sizes of VPs required between destinations; the demands could be specified in terms of numbers of connections, associated bandwidth requirements and associated QoS requirements. Constraints may also exist at the specific problem layer that are artefacts of the capabilities of the technologies used in the problem.

The mapping function performs a mapping from the specific problem to the generic problem and vice versa. This mapping is necessary to apply the solution algorithms. The mapping is a two-way mapping in the sense that it is necessary to map from the specific problem to the generic problem in order to obtain a solution and it is also necessary to map the solution of the generic problem back to the specific problem domain.

The generic problem is an abstraction of network design and configuration problems that can be applied in many circumstances. It is defined in terms of more abstract quantities than the specific problem. A key characteristic of this problem is that it must be **flexible.** If it is not flexible, then it cannot be reused: if it cannot be reused then the single most important motivation for this approach is lost. Having said this, there are constraints on how flexible the problem can be. The more general the problem, the more time is required to solve the problem. Hence, some balance between generality and the associated flexibility and tractability must be found.

The final layer is the solution layer. In general, any algorithm to solve the generic network design problem can be used here. However, the overall approach is made much more powerful if there is a suite of solution algorithms that can all solve the generic problem. The solution algorithms would have differing characteristics and the most appropriate can be chosen to solve the problem. One natural way of differentiating between solution algorithms is to consider them in terms of time versus solution quality: for some applications it could be useful to obtain a solution in a small time even though the solution quality is not excellent, while for other applications solution quality may be the primary concern. Developing this suite of solution algorithms can take some considerable effort. However, this needs to be done only once. Then the suite of algorithms can be applied to a large number of problems.

3.2.2 Advantages of this Network Optimisation Framework

The primary advantage of using this approach is that less effort is required to solve specific network optimisation problems. The conventional process for solving a network optimisation problem described above consists of modelling and solution stages – using the approach described here, this is reduced to a modelling and mapping process. The work required to map from the specific problem to the generic problem is less than that required to develop a formal mathematical model and then develop some solution algorithm for it.

Also, the set of solution algorithms would be like a well-tested library that gets reused over and over. This is in contrast to the situation in which a new network optimisation algorithm is developed for each specific problem: in this case, the solution algorithm must undergo rigorous testing for the results to be valid. This can take considerable time and effort.

This approach also has other benefits. By applying this layered approach to solving the network optimisation problem, the problem modelling process can be **decoupled** from the solution process. If the network optimisation problem is decomposed in this manner, then the work involved in solving network optimisation can also be decomposed in this manner. Groups with expertise in modelling network optimisation problems can focus their efforts on mapping specific problems to the generic one and groups with expertise in solving optimisation problems can concentrate on solving the generic problem.

3.2.3 Caveat

This approach is not applicable in all cases. Some applications have certain strict requirements that obviate the use of this approach. For example, some network optimisation problems may require that solutions be found very quickly, and hence specific algorithms tuned to the particular application may need to be devised. Having said this, there is a significantly large class of problems to which this generic approach can be applied to make it interesting.

3.3 The Generic Network Design Problem

In order to use this approach, it is necessary to devise a clear, well-defined problem model. The specific problem model can then be mapped to this generic problem model. Here, the generic problem model is described in detail.

The generic network optimisation problem model is described in terms of the model inputs and outputs. The model inputs are:

- a graph of the nodes and candidate links,;
- the demands on the network;
- the cost functions.

The outputs of the model are

- the overall cost and
- the network configuration that results in this cost.

Each of these is discussed in more detail below.

3.3.1 The Input Parameters

Graph Containing the Nodes and Candidate Links

The first input to the problem is a directed graph that contains the set of nodes in the generic problem and the set of candidate links that the demands can be carried on. The graph can be broken down into a set of nodes and a set of edges.

The set of nodes in the network may map exactly to the set of nodes in the specific problem. Alternatively, as is discussed below, some extra nodes may be added to the set of nodes in the specific problem by the mapping function.

In both cases, the set of nodes in the generic problem contains the set of nodes in the specific problem. These nodes are simply the nodes through which the demands are routed. The nodes in the specific problem would typically be points at which traffic aggregation occurs. They could be the set of *Points of Presence* (POPs) in an operator's network, and the demands could be incident to each of the POPs. They could be a set of core nodes within a network, and the problem is to determine how to configure the demands within the core network. Alternatively, they could be a set of locations in an enterprise network that generate traffic and the problem is to determine how to interconnect these locations with minimum cost.

Here, it is implicitly assumed that the node locations are fixed. One important element of some network design problems is determining where to place nodes e.g. POPs. The node placement problem differs considerably from the route configuration problem. Typically in the node placement problem the cost of the network varies with the distances between the nodes. There is no location information in the generic problem model presented here; therefore, the node locations are implicitly assumed to be fixed. This network optimisation framework is not sufficiently flexible to be able to incorporate node placement problems.

The edge data in the graph is used to indicate the set of links on which the demands can be routed. The graph may be fully connected, in which case all possible links are candidates for routing the demands. Alternatively, the graph may not be fully connected, in which case the network designer has decided not to use or permit a link between some of the node pairs.

Each edge in the graph represents a bidirectional connection, i.e. traffic can flow in either direction on the link. Note, however, that each edge is directed. This facilitates differentiation between traffic flowing in different directions on the link, which enables asymmetric links to be modelled. If non-directed links are used, then it is not possible to differentiate between the direction of flow of traffic on the link and hence it is not possible to model asymmetric links.

Since directed edges are used to model links, it is natural to introduce the notion of **upstream** and **downstream** traffic. Upstream traffic is defined as traffic that flows in the same direction as the edge; downstream traffic flows in the opposite direction.

A rather obvious alternative to this approach is to completely decouple the upstream and downstream links. This could be done by using directed edges to represent links carrying traffic flowing in one direction. Then a bidirectional link could be represented by two unidirectional links. The problem with this approach is that the cost of a link is often dependent on the traffic carried in both the upstream and downstream directions. For example, the cost of a link may be dependent on the maximum of the upstream and downstream traffic rates. It is difficult to model this if the links are completely decoupled. Hence it is difficult to completely decouple the upstream and downstream links.

Note that no specific link data exists in this problem formulation, i.e. the generic problem model does not contain links with some associated capacity: the edges do not have weights associated with them. However, as is discussed below, it is possible to incorporate, say, link capacities – which would typically be modelled as weights on a graph – into the cost function.

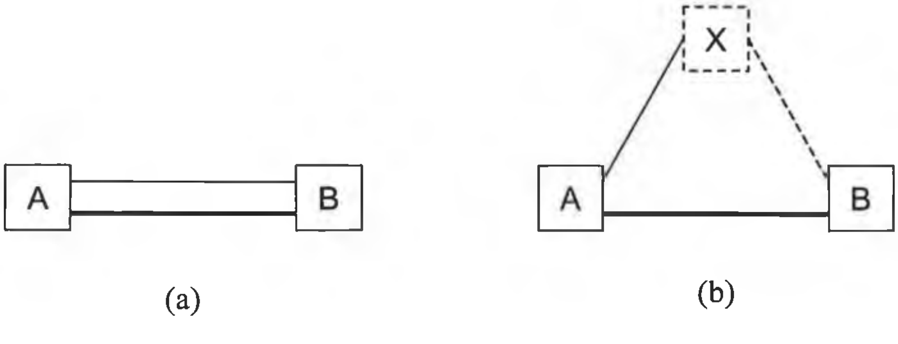


Figure 3-2: Modelling two links between two nodes by introducing an extra node and link. The specific problem model is shown in (a) and the mapping to the generic problem is shown in (b).

The graph of all potential links in this problem formulation is not permitted to have more than one edge between any two nodes. This is because the solution algorithms may apply algorithms to find shortest paths on the graphs: such algorithms typically assume a single edge between nodes. However, there are circumstances in which it may be useful to have two or more candidate links between two nodes. For example, in a network design problem, two candidate technologies may be available to implement a single link, each having different cost/capacity characteristics. Alternatively, in an enterprise network design problem, a choice of a number of service offerings from different operators with different cost/capacity characteristics may be available. Performing an appropriate mapping when mapping from the specific problem model to the generic problem model can cater for this scenario. A straightforward approach to doing this is to insert an extra artificial node into the problem and to insert an extra link with cost identically equal to zero. This is illustrated in Figure 3-2.

In Figure 3-2, both the specific problem model view and generic problem model view are shown for a situation in which the network designer wishes to have two candidate links between two nodes. Node X in Figure 3-2(b) is the artificial node that is introduced by the mapping process. The link between nodes X and B must have a cost function that is identically zero.

The issue of which link to connect to the new node is not considered here. The intention is merely to show that the scenario in which two candidate links at the specific problem model layer can map to a scenario in which only single links exist through the addition of an extra node and link.

The Demands on the Network

The demands describe traffic that is assumed to be incident on the network for the purposes of the problem. A particular demand describes some subset of the total traffic. Different specific problems can have quite different demand characteristics and demand characterisations.

Examples of the ways the demands may be specified at the specific problem level include:

- a set of voice traffic demands characterised by a single source and single destination, an arrival rate and a mean holding time;
- a set of point-to-point data demands characterised by a bitrate;

- a set of data demands, all of which are characterised by a bitrate and a single source, some of which may have multiple destinations to which they can transmit traffic;
- a mixture of voice, video and data traffic in which the voice traffic is characterised by an arrival rate and holding time, the video traffic is characterised by another arrival rate and holding time and the data traffic is characterised by a mean and peak rate. Note that each of these services may require a different QoS from the network.

It is clear from the above examples that the set of demands can be quite diverse. Here the idea is to capture the essential characteristics of the demands for the generic problem.

One natural way of classifying different types of demands that has a very great impact on the problem complexity is by identifying the *scope*[4] of the demand. The scope of a demand loosely characterises how many nodes terminate (either as source or destination) the traffic that constitutes the demand. Three natural examples of the scope of a demand are given as examples of the use of scope in the context of diffserv networks: these are the so-called 1:1, 1:n and 1:any scopes. Both the source and destination nodes are specified for all traffic that constitutes a demand with a 1:1 scope. This is illustrated in Figure 3-3(a). Demands with a 1:n scope consist of traffic having a single specified source and terminating at one of a number of specified destinations. This is illustrated in Figure 3-3(b). Only the source is specified for a demand with 1:any scope; any destination is possible for the traffic that constitutes the demand. This is illustrated in Figure 3-3(c). Other scopes are possible in which the demands may cover some set of source and destination nodes, but these are not considered here.

It is implicitly assumed here that there may be some variation in the demands with more complex scopes: otherwise they can be reduced to 1:1 scopes. For example, if the scope for a particular demand is 1:n, then it is assumed that it is not possible to say with certainty that each destination will receive some known fraction of the traffic. If this was the case, then the demand with 1:n scope would be equivalent to n demands with

[4] The term scope, as applied here, is used in the same sense as in the Differentiated Services model. This terminology was introduced in the diffserv framework document – a document that described the scenario in which diffserv would operate. This was a working document of the IETF working group, which was not updated and consequently, it is not considered by the IETF to be a current document. Since the IETF do not archive contributions, no authoritative reference to this work can be given.

1:1 scope. For demands with 1:any scope there is the added complication that the set of destination nodes is not even known.

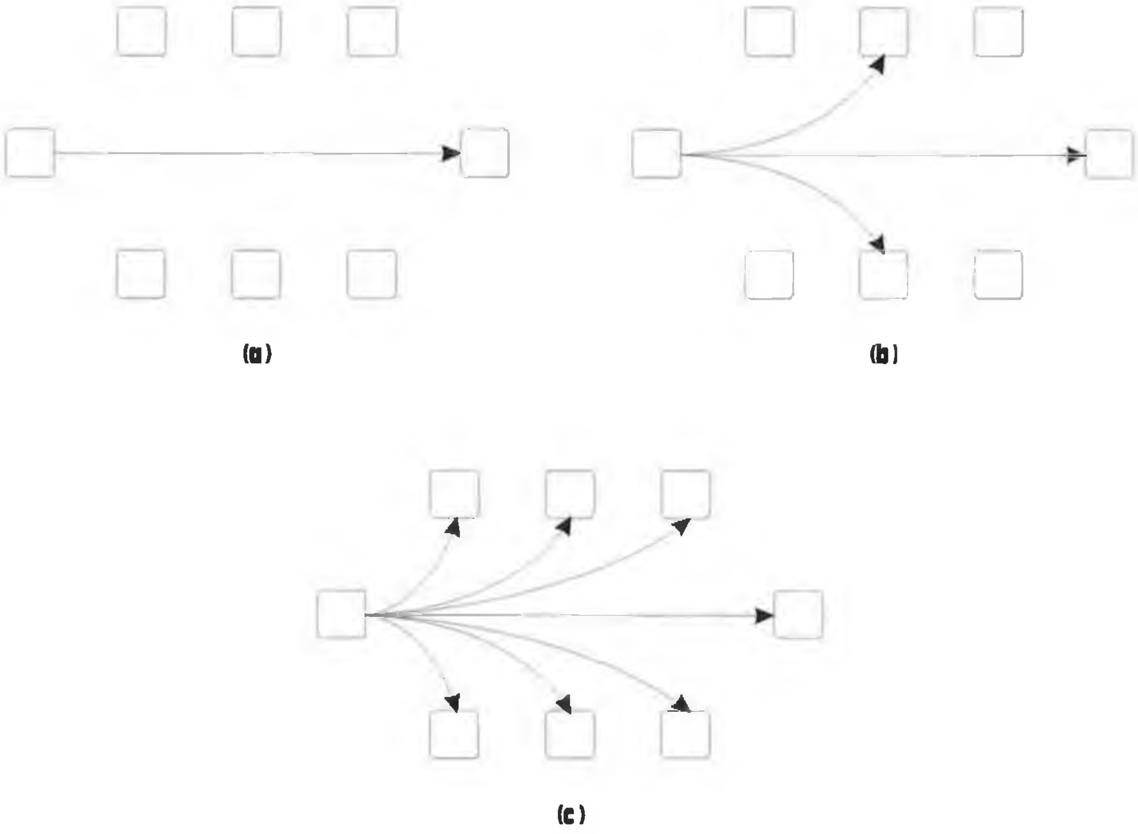


Figure 3-3: Examples of demands with different scopes – (a) 1:1 scope, (b) 1:n scope and (c) 1:any scope.

Here, only demands with 1:1 scope are permitted in the generic model. There are three reasons for this. Firstly, network design problems in which demands with arbitrary scopes are permitted are difficult to solve. In the absence of any extra knowledge, it is impossible to know how the traffic is distributed amongst the destination nodes for the more complex scopes described above.

Secondly, most established technologies only permit 1:1 scopes; the notion of more complex scopes is still a relatively new idea. Older technologies were not able to support more generic scopes and, as a result, non-broadcast communications in general usually had to be point-to-point. Point-to-point communications constitute a means of communication that many people are very comfortable with. Consequently, a considerable amount of traffic on networks for years to come will be of a point-to-point nature. Such traffic is well characterised by demands of 1:1 scope and hence many network optimisation problems will consist of such demands.

Thirdly, while permitting demands with a more general scope would increase the flexibility of the model, this increased flexibility would come at the expense of solution complexity: permitting such demands would greatly increase the complexity of the generic problem. So, even though flexibility is an essential characteristic of the generic model, problem complexity concerns impose limits on the amount of flexibility that can feasibly be incorporated into the model.

Even though the demands are limited to 1:1 scope, there are some factors fuelling an increase in real demands with more complex scopes. Two are considered here. Firstly, newer technologies are enabling more sophisticated communications and enabling point-to-multipoint communications, resulting in applications with more generic scope requirements. Secondly, if customers have large networking requirements, they often prefer simply to specify how much traffic they will generate and inject into the network rather than specifying each point-to-point demand individually. These two factors indicate that demands with more generic scopes must be catered for in future networks, which means that design and configuration tools should be able to accommodate such demands.

It is possible to use the mapping function to map demands with more general scope to demands of 1:1 scope. Then the generic network design model can be applied to problems in which the demands have more general scope. Of course, this mapping is only possible if some assumptions are made on the way that the traffic is split between the node-pairs. Such assumptions could be based on traffic models or, as is more likely, traffic measurements. This approach cannot produce solutions that are as good as those produced using an approach tailored specifically for demands with more general scope, but it does enable this model to be applied **quickly** and **easily** such that some reasonable solution to the problem can be found.

From the examples of the demands given above it is clear that there exist many different types of demands that can be parameterised in different ways. The generic problem must be able to cater for many different types of demands; the characteristics of demands in the generic problem model should not be specific to any specific problem. Rather, some essential characteristic(s) of the demand that can reflect the amount of resources required to carry the demand should be used.

In the generic problem, each demand is characterised by a single parameter. Typically this parameter is a bandwidth parameter, or in some cases it may be a so-called *effective* *bandwidth* (see [Hui88] for the initial development of the effective bandwidth notion, or [Kel96] and references therein for development of the concept and later work).

Mapping from the specific problem demands to the single parameter required in the generic problem can be quite complex. For demands that are characterised by Poisson arrivals and exponentially distributed holding times, the well known Erlang blocking formula [Sys86], coupled with a desired blocking parameter, can be used to determine some equivalent capacity that can carry the traffic while meeting the target blocking probability. In situations in which there is a more diverse set of applications, each of which has some effective bandwidth, and there are some target blocking parameters, techniques such as that used by Bean, Gibbens and Zachary [BGZ94] or the techniques described in [RMV96] can be used to determine some capacity requirement. For situations in which there is a data demand characterised by a peak and a mean parameter, some a priori knowledge could be used to determine an effective bandwidth for the demand. Alternatively, a conservative approach could be simply to choose the peak bandwidth as the required bandwidth for the demand. Use of effective bandwidths in this context will be discussed later.

Often a demand may have a QoS associated with it, meaning that the traffic that constitutes the demand must obtain the specified QoS. For example, packet video connections may have specific delay and loss requirements of the network while other data traffic using the same network may not have such stringent requirements. As with demand parameterisation, QoS is a difficult concept to incorporate into a generic design model in a generic way. As was seen in the previous chapter, different networking technologies have different types of QoS measure. For packet traffic QoS is typically measured in terms of packet delay or loss; for connection-oriented traffic QoS is typically measured in terms of blocking probabilities. For connection-oriented traffic carried over a packet network some combination of both measures would be used to reflect the QoS offered by the network as is the case in ATM.

It is assumed that the QoS requirements of each demand are incorporated into the mapping from the demand parameters to the single parameter used in the generic design model. For example, a single bandwidth parameter can be used to model the amount of voice traffic to be carried between two nodes. This can be calculated based on the QoS requirements between the nodes: if the internodal traffic requires high QoS, i.e. low blocking, then more resources should be reserved between the node pair. Similarly, the effective bandwidth approach can incorporate QoS requirements into the demand. If the QoS required is (in some sense) high, then the effective bandwidth will also be high.

In summary, it is assumed that the demands provided as input to this network design problem are unidirectional point-to-point demands characterised by a single parameter that encapsulates the magnitude of the demand coupled with the QoS requirements of the demand.

The Cost Function

In general, the cost function must be a function of the way the demands are routed on the network, i.e. the cost function must be a function of the route configuration. This is the most general form of cost function. Allowing the cost function to be explicitly dependent on the route configuration is not entirely natural: costs are not usually associated with specific routes or route configurations in network optimisation problems. Indeed, if they did, then the network design problem would be solved simply by choosing the lowest cost route for each demand.

Here, the cost function is comprised of two components: a link cost component and a node cost component. The overall cost is simply the sum of these components. Both the link costs and the node costs are dependent on the way the demands are routed on the network and hence the overall cost function is dependent on the way the demands are routed on the network. This is a more natural way to construct the cost function.

Some particular situations could be envisaged that are not catered for by such cost functions. For example, in some network optimisation problems it may be desirable to weight one or more routes for a demand such that they are more or less likely to be chosen. This is not possible using the cost functions described here. However, usually when a network designer wishes to achieve this, the motivation is to reduce the likelihood of a demand being routed through a particular node or link. This can be achieved by choosing the link/node cost function appropriately: if the node/link cost is high then it will become unattractive and hence there is a smaller likelihood that traffic will be routed on it.

Here, a higher level view can be taken: a cost function can be chosen for a particular node or link that makes it either attractive or unattractive, and in this manner, demands can be routed to/from some particular nodes/links.

The link cost component of the overall cost function reflects the link costs associated with routing the given demands in a particular manner. In network design problems, it is usual to have a cost associated with a link that is independent of the costs associated with other links in the problem. Consequently, the total link costs can be decoupled into individual link costs. Moreover, the individual link cost functions are typically increasing functions of the aggregate capacity carried on the link.

Since a separate link cost function exists for each link in the problem, it is not difficult to incorporate differences between different links into a problem. For example, different links in the problem may be implemented using different technologies. In this case, the cost function associated with different links could be quite different. It is also simple to model situations in which, say the cost associated with the link has some dependency on the distance between the link endpoints.

In the graph that is used to specify the set of nodes and candidate links, a directed edge is used to indicate that a particular edge can be used to carry traffic. Despite the directed nature of the edge used to represent the link, traffic can be carried in either direction on the link. A single cost function is used to model the link; this cost function is a function of both the upstream and downstream traffic carried on the link.

This choice of link cost model caters for situations in which the upstream and downstream traffics can and cannot be decoupled. For example, in a situation in which installation of a new symmetric bidirectional link is under consideration, the upstream and downstream link costs cannot be decoupled. The cost of the new link is a function of the greater of the upstream and downstream traffics. Consequently, the cost function must be a function of the maximum of the upstream and downstream traffics. Conversely, if the problem is to determine how to balance load on a network, then the upstream and downstream traffics can be completely independent, and the cost function can be decoupled into a cost function for the upstream traffic and a cost function for the downstream traffic.

The link cost functions can be very general, but usually the cost will increase with increases in the upstream and downstream traffic carried on the link. They can be non-linear functions of the upstream and downstream traffic. Examples of the type of cost functions that may be used are shown in Figure 3-4.

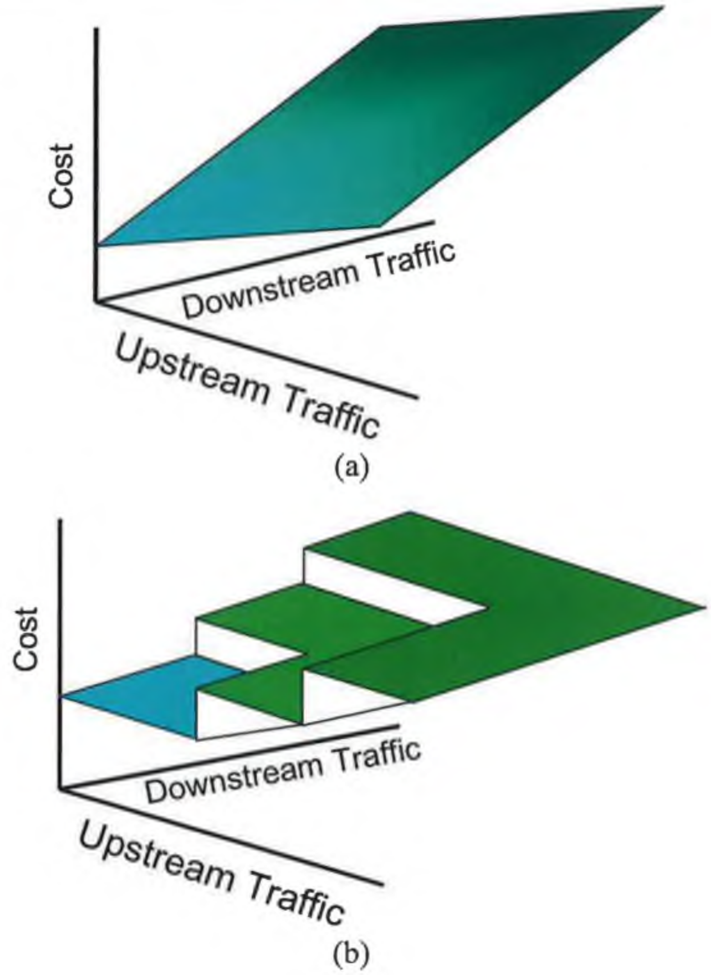


Figure 3-4: Examples of different link cost functions. In (a) the cost function is linear in both upstream and downstream traffic; in (b) the cost increases with capacity in a stepwise fashion.

In some cases, the network designer is interested in designing a real network in which the links are physical links and the costs associated with each link are the costs associated with installing a link of the given capacity between the given nodes. In other cases, the network designer may wish to determine where to add extra capacity to a network to carry the given set of demands. In this case the cost functions used in the generic problem are the costs associated with adding extra capacity on each link. These cost functions must incorporate the available link capacities into the cost function. They will have a value of zero for routing capacity on the link of less than or equal to the capacity available on the link. In other cases, the network designer may be interested in traffic-engineering related problems in which the problem is to determine how best to route a set of demands on an existing network. In this case, the link costs are not so obvious. Some function that increases with decreasing available capacity can be used to obtain a solution in which the total used capacity is minimised and the load balanced on the network.

As with the link costs, the node costs can be modelled as a set of individual node costs. The node costs are also independent of each other and hence it makes sense to decouple them: the total node costs are then the sum of the individual node costs.

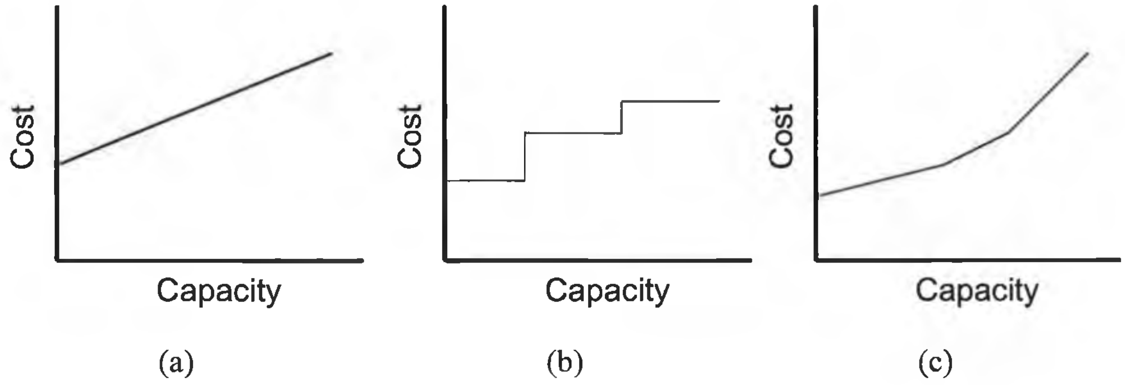


Figure 3-5: Examples of different node cost functions. The capacity is the aggregate capacity switched through the node. In (a) the cost function is linear in capacity; in (b) the cost increases with capacity in a stepwise fashion and in (c) the cost function is a piecewise linear approximation to an exponential function.

The node cost for each node depends on the aggregate traffic routed through the node. Typically, the node cost increases as the amount of traffic switched through the node increases. The aggregate traffic routed through each node can be determined from the route configuration and knowledge of the demands. Examples of the node costs are shown in Figure 3-5.

The node cost functions may be chosen in a similar manner to the link cost functions. In scenarios in which the objective is to construct a network the node costs can represent the actual costs of installing a node. In this case, a number of different switch configurations may be possible for a single node, each having a different switching core with different switching capacity. As the required capacity increases, higher performance switch configurations may be required, resulting in increased cost. Hence a stepwise incremental cost function may be used in this case to model the node costs. Alternatively, if a node is in place, and the objective is to ensure that a limit is imposed on the amount of traffic switched through a node, then the node cost could increase exponentially as the node capacity reaches its limit.

3.3.2 The Output Parameters

In general, the network optimisation problems considered here focus on how to route the demands on the network to minimise some cost function. The specific technologies involved and the problem objectives may differ greatly, but the problem ultimately reduces to this. The generic network optimisation problem considered here is no different.

The outputs of the network optimisation problem model are then:

- *an overall cost*: this is the minimum value of the objective function obtained by the solution algorithm,
- *a network configuration*: this is the route configuration of the demands on the network that results in the above cost.

While the solution to the generic problem consists primarily of a route configuration, the network designer may be more concerned with issues other than how the demands are routed on the network. For example, the network designer may be interested in where extra capacity should be added on an existing network or how much free capacity will exist on a network when some set of traffic demands are incident to the network. These quantities can be derived from the solution to the generic problem. Hence the solution to the specific problem may differ in character from that of the generic problem, but in all cases it can be derived from network configuration and solution cost data.

In the most general solution to the generic problem the demands may be split and fractions of the demands may be carried over different routes. In general, this can result in cheaper solutions. However, it is dependent on the technology being able to support arbitrary demand splitting and it also greatly increases the complexity of the generic problem. For these reasons it is assumed that the demands cannot be split.

In some cases, it is very beneficial to split the demands. For example, in situations in which the demands are large relative to the capacities of the connections, and the technology is able to support splitting of demands, splitting demands can result in significantly reduced costs. Such situations are often identifiable at the specific problem layer and the mapping function can split large demands into a number of smaller ones to reduce the overall costs.

3.4 Mathematical Problem Formulation of the Generic Problem

The mathematical problem can then be formulated. Denote the following given quantities as follows:

- N : the set of nodes in the problem;
- Λ : the set of edges in the graph. These are the set of candidate links on which the demands can be routed;
- $\Gamma(\mathrm{N},\Lambda)$: the graph relating the nodes and links;
- $(\psi_\lambda,\zeta_\lambda)$: the origin and destination nodes for link $\lambda \in \Lambda$;
- Δ : the set of unidirectional demands in the problem;
- η_i : the capacity required for demand $\delta_i \in \Delta, i \in 1\ldots|\Delta|$;
- $\delta_i = (\alpha_i, \beta_i, \eta_i)$: the triplet which defines demand $\delta_i, i \in 1\ldots|\Delta|$, where α_i is the source node for demand δ_i and β_i is the destination node for demand δ_i ;
- $\phi_\lambda(s,t)$: the cost function for link $\lambda \in \Lambda$; this is the cost of carrying capacity s in the upstream direction and capacity t in downstream direction;
- $\gamma_v(u)$: the cost function for node $v \in \mathrm{N}$; this is the cost associated with switching capacity u through node v ;

The following are then defined:

- P : a route configuration;
- ρ_{δ_i} : the route used by demand $\delta_i \in \Delta$ in route configuration P
- $\mathrm{I}_{\delta_i}^{\lambda}(\mathrm{P})$: the link-path incidence matrix for route configuration P. This has the value 0 if ρ_{δ_i} does not contain link λ , it has a value 1 if the demand is routed from ψ_λ to ζ_λ and has a value of -1 if the demand is routed from ψ_λ to ζ_λ .

The overall cost function is defined as follows:

$$\Phi(\mathrm{P}) = \sum_{\lambda \in \Lambda} \phi_\lambda(s_\lambda(\mathrm{P}), t_\lambda(\mathrm{P})) + \sum_{v \in \mathrm{N}} \gamma_v(u_v(\mathrm{P}))$$

where:

- $s_\lambda(\mathrm{P}) = \sum_{\delta_i | \mathrm{I}_{\delta_i}^{\lambda}(\mathrm{P})=1} \eta_i$ – the traffic carried in the upstream direction on link $\lambda \in \Lambda$
- $t_\lambda(\mathrm{P}) = \sum_{\delta_i | \mathrm{I}_{\delta_i}^{\lambda}(\mathrm{P})=-1} \eta_i$ – the traffic carried in the downstream direction on link $\lambda \in \Lambda$
- $u_v(\mathrm{P}) = \sum_i \eta_{\delta_i} \left(\sum_{\lambda | \psi_\lambda = v} \mathrm{I}_{\delta_i}^{\lambda}(\mathrm{P}) - \sum_{\lambda | \zeta_\lambda = v} \mathrm{I}_{\delta_i}^{\lambda}(\mathrm{P}) \right) + \sum_{\delta_i | \alpha_i = v} \eta_i + \sum_{\delta_i | \beta_i = v} \eta_i$ – the total traffic switched through node $v \in \mathrm{N}$.

The optimisation problem can then be formulated as follows:

$$\text{Find } \mathrm{P}^* = \min_{\mathrm{P}} \Phi(\mathrm{P})$$

This problem formulation does not include any constraints. In particular, there are no link or node constraints, i.e. constraints ensuring that the amount of traffic carried on a particular link or switched through a particular node does not exceed its capacity. Not including such constraints in the formulation means that it is more flexible and can be applied in more situations. For example, this formulation can be applied in green field network design problems. Alternatively, it could be applied in situations in which the problem is to determine how much extra capacity to add to a network to ensure that the forecast traffic demands can be carried.

Note that there is a finite, albeit large, set of route configurations. Consequently, one or more minimum cost route configurations must exist. This is independent of the nature of the particular link and node cost functions specified as inputs to the problem.

3.4.1 Problem Complexity

The complexity of this problem is dependent on both the nature of the cost function and the size of the state space. The overall cost function is the sum of a set of quite arbitrary link and node cost functions. In general, these will be non-decreasing functions, although they may not be. Typically, the state space is very large. Moreover, the state space typically contains many local minima: this further compounds the difficulty of the problem. These factors combine to make the problem very difficult to solve.

Since each state in the state space maps to a single route configuration, the size of the state space is equal to the number of route configurations in the problem. The rate at which the state space grows can be illustrated as follows. The size of the state space is dependent on the number of demands in the problem and the number of routes for each demand. In general, the number of routes available to each demand can be large and can vary with each demand. For the purposes of illustration, it is assumed that the number of routes available to each demand is a constant, k. If there are D demands in the problem, then the number of route configurations in the problem is k^D. The number of demands is typically dependent on the number of nodes in the problem. If demands exist between each node pair, then the number of demands is $o(N^2)$. The number of routes is then $o(k^{N^2})$.

Nodes	Demands	Routes ($k=3$)	Routes($k=5$)	Routes($k=8$)
5	20	3.49×10^{9}	9.54×10^{13}	1.15×10^{18}
10	90	8.87×10^{42}	8.08×10^{62}	1.90×10^{81}
15	210	1.57×10^{100}	6.08×10^{146}	4.46×10^{189}
20	380	2.02×10^{181}	4.06×10^{265}	1.49×10^{343}
25	600	1.87×10^{286}	2.41×10^{419}	7.14×10^{541}
30	870	1.25×10^{415}	1.27×10^{608}	4.88×10^{785}

Table 3-1: Approximate number of states in the state space as a function of node size. Here it is assumed that there are $N(N-1)$ demands in an N-node problem.

Table 3-1 shows how rapidly the state space grows. Even for 30-node problems, which are not particularly large problems, the number of states in the state space is enormous. In reality, the number of routes is dependent on the number of nodes in the problem. Problems consisting of more nodes usually contain more routes between the nodes. Consequently the estimates used in Table 3-1 are conservative: the actual sizes of the state spaces are usually larger than the numbers quoted above.

Costing all of the states in the state space is not feasible. This is can be seen as follows. Assume there are only 3 routes per demand and a computer was available that could cost 10^{10} states per second[5]. For the 30 node problem it would take approximately 10^{405} seconds or approximately 10^{399} years. Even tenfold increases in available processing

[5] Current high-specification microprocessors operate at 1000MHz. Single processor systems based on such microprocessors certainly cannot cost states at this rate. Multiprocessor systems may be able to cost states at this rate if specific code was written to take advantage of the multiprocessor architecture.

power make little difference on the time required to perform exhaustive searches of the state space. Clearly then it is unrealistic to search all the states in the state space: an alternative approach must be used.

This problem is an example of a combinatorial optimisation problem. Optimisation problems on discrete state spaces fall into this category of problem. These problems can often be very difficult to solve. Difficult combinatorial optimisation problems can fall into the class of so-called *NP-complete* or *NP-hard* problems (see [GJ79] for a discussion of complexity of combinatorial problems). No known algorithms exist that can solve NP-complete and NP-hard problems in polynomial time; the time taken to solve such problems can increase exponentially with problem size. In general, it is not possible to find the optimal solution to such problems in any reasonable time. Hence some heuristic algorithms are typically used to obtain some reasonable solution. Some such algorithms are discussed below.

3.4.2 Examples of the Use of the Generic Model

Two examples of how this generic problem formulation could be used in two different problems are given. In the first example, the problem is to design a network from a green-field scenario that can accommodate the given demands. In the second example, the problem is to determine whether a planned network can meet some forecast demands.

These problems are not considered in detail here, but rather, the objective is to illustrate how the generic problem can be used to solve these types of problems. Specifically, the technologies used to implement the network, the nature of the demands, the cost functions used etc. are not considered. A more detailed exposition of how the specific problem model can be formulated and mapped to the generic problem model is given in later chapters.

The Green Field Network Design Problem

In this problem no network exists. The designer is faced with the problem of determining the minimum cost network that will be able to carry the given traffic demands.

The set of available links can be chosen according to the needs of the network designer. If the network designer is willing to permit a link between every node-pair, then a fully connected input graph can be used. However, if the network designer does not wish to permit certain links the input graph of candidate links will not be fully connected.

The following data can be input to the generic problem. The demands are specified somehow in capacities: how these are calculated is not important here. The links are bidirectional in this problem. The cost of each link can be exactly the cost of installing a bidirectional link of capacity that is the greatest of the upstream and downstream traffics. This could consist of the cost of installing plant plus the costs of terminating the line. The node costs could be the cost of installing a node that can carry the capacity switched through the node. The bulk of this cost could be the cost of the switching system, which would be dependent on the capacity switched through the node, but could also include costs of other node subsystems such as management systems. The overall cost is the sum of the link and node costs.

Once this data is input to the generic problem, a solution can be obtained. This solution will consist of a route configuration and an associated cost. The network designer is only interested in the network capacities and the overall cost. The mapping function can process the output of the generic problem to obtain the link capacities and the route configuration data can then be discarded. This data can then be returned to the network designer.

The Network Planning/Forecast Problem

This problem is a little different from the problem above. Here, the problem is to determine whether or not a particular set of demands can be carried on a given network. The intended application is a forecasting application in which the demands are forecast demands and the network may be either an existing network or a planned network.

There may be many ways in which the demands could be routed on the network. Alternatively, it may not be possible to route the demands on the network: sufficient resources may not exist on the network. The mapping function should be able to determine if this is the case.

A natural approach to configuring the network in this context is to attempt to maximise the spare capacity on the network while attempting to keep the load balanced. Using this objective when configuring the network will have the effect of utilising the network resources efficiently[6] while making the network robust to changes in the demands. However, there is a trade-off here between efficient use of network resources and balancing the load. Maximising the efficiency of the network can often mean maximising the use of cheaper nodes or links. This is not consistent with the load balancing objective.

This trade-off manifests itself in the choice of link and node cost functions. If a link cost function is chosen that increases greatly if the amount of spare capacity on each link becomes small, then the load balancing and efficiency objectives can be met. The rate at which this cost function increases for each of the links determines the trade-off between maximising the spare capacity and ensuring that residual capacity exists on each link. Similarly, if a node cost function is chosen such that the cost increases dramatically when the capacity switched through the node nears the node limits, load balancing can be achieved with respect to the nodes. In some situations, this may not be important – the nodes may be over-engineered and may support capacities much greater than those switched in the network. In this case the node costs could be made constant. Hence the node costs would be independent of the capacities switched through the node.

There is the possibility that the demands cannot be accommodated on the network. The generic problem formulation does not permit constraints on the used capacity on each link. Hence, it is not possible to make states in which the traffic carried on a link or switched through a node exceeds the available capacity infeasible. If a very large cost is assigned to a link if the capacity is exceeded, then the solution algorithm will naturally tend to avoid these solutions. Similarly, if a large cost is assigned to a node if the node capacity is exceeded, solutions in which the node capacity is exceeded will also be avoided. Also, if the link or node cost is sufficiently large, the mapping function can recognise that the algorithm was unable to find a solution that enables the given set of demands to be carried on the network.

3.5 Generic Problem Solution Approaches

As illustrated above, the state space for these problems is very large. In general, the cost function can be quite complex and this can result in a state space containing many local minima. These two factors make it very difficult to find a specific algorithm that can be

[6] Here, efficiency is used in the sense of the minimal use of overall network resources, i.e. maximising the spare capacity on the network.

guaranteed to find a globally optimal solution for any realistic problem. Since the generic problem can be NP-complete or NP-hard, no known algorithm exists that can find solutions to reasonable size problems in reasonable time. Note that this is not a characteristic of the overall approach considered here: even if the specific problems were considered in isolation rather than as part of this generic network design approach no known algorithm would be guaranteed to find the optimal solution. Heuristic algorithms that can find reasonably good solutions to the problem in reasonable time must be used.

Here, the notion is to have a suite or 'toolbox' of heuristic algorithms that can be applied to find solutions to the problem. The toolbox can consist of different algorithms having different operating characteristics. In particular, there can be a trade-off between the solution quality and the execution time. For example, some algorithms may solve a problem relatively quickly but obtain a relatively poor solution, while other algorithms may solve the problem more slowly obtaining a higher quality solution.

A good description of heuristic approaches that can be used to solve combinatorial problems can be found in [Ree95]. Some of these heuristics are so-called local search heuristics. The fundamental philosophy behind local search algorithms is to iterate through states in some subset of the state space until some reasonable solution is found. Local search heuristics are typically applied in cases in which the state spaces are very large and it is not feasible to search all of the states. The idea is to limit the search to some subset of the state space that contains 'reasonable' solutions. Hence it is a local search rather than a global search. The greedy algorithm, simulated annealing algorithm and tabu search algorithms described below are examples of local search algorithms.

When considering local search heuristics it is necessary to define a *neighbour relation* between states. This implicitly introduces the concept of a *neighbourhood* of a state – the neighbourhood of a state is the set of states that are neighbours of that state. Typically, neighbouring states are very similar. Local search algorithms typically iterate through the states moving from neighbour to neighbour until some terminating condition is reached.

The choice of neighbourhood can affect the performance of the algorithm. Choosing a neighbour relation that results in large neighbourhoods can greatly increase the execution times of some algorithms. For example, algorithms that evaluate the cost of all the neighbours in a neighbourhood will take much longer if the size of the neighbourhood is greatly increased. There are drawbacks to choosing small neighbourhoods: since the number of moves possible in any state is small it may take a long time to navigate through all the neighbourhoods to a good solution.

Two examples of local searches heuristics are described next. These are followed by a brief description of a further two examples of local search heuristics and a short note on reducing the size of the state space by limiting the amount of routes available to each demand. This is followed by a description of the particular choice of state space and neighbourhood chosen for this work.

3.5.1 Greedy Algorithm

The greedy algorithm is a straightforward local search algorithm. This is the local search analogue of the steepest descent techniques that can be used to find locally optimal solutions to problems in continuous domains. The state at iteration $i+1$ is obtained from the state at iteration i by iterating through all the neighbours of the state at iteration i and choosing the neighbour with the lowest cost. The algorithm terminates when a state is found that has a lower cost than that of all its neighbours. A more formal description of the greedy algorithm is given in Algorithm 1.

```
Step 1    Choose some initial starting point, $x_0$. Set $i = 0$
Step 2    Determine $x_{\text{low}}$, the lowest cost neighbour of $x_i$, by iterating through the neighbourhood of $x_i$.
Step 3    If $c(x_i) < c(x_{\text{low}})$ where $c(x)$ is the cost of state $x$ then terminate.
Step 4    Set $x_{i+1} = x_{\text{low}}$, increase $i$, go to step 2.
```

Algorithm 1: Greedy algorithm.

Like the steepest descent techniques, this local search algorithm is very sensitive to the initial starting point. If a bad initial starting point is chosen, then the solution can also be bad. This point is illustrated in Figure 3-6. If point ‘a’ is chosen as the starting point for the algorithm, then the m_1 solution will be found, but if the search algorithm is started at point ‘b’, then the m_2 solution will be found.

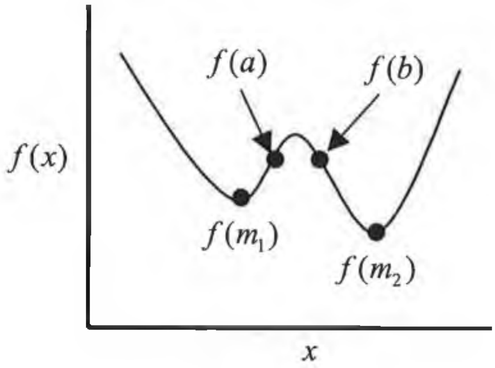


Figure 3-6: Example of sensitivity of greedy and steepest descent algorithms to initial starting point. If the algorithm is started with starting point $x = a$, then the solution at $x = m_1$ will be found; if started at $x = b$ then the solution at $x = m_2$ will be found.

To obtain good solutions, it is necessary to consider the choice of the initial starting point. In many problems, it is difficult to identify characteristics of a good solution that can be used to concentrate the algorithm on some part of the state space. Often a random starting point is chosen. Often the algorithm is performed a number of times with a number of different starting points in some attempt to broaden the search to different parts of the state space.

In the generic problem considered here, a few natural and yet reasonable starting points are considered. Choosing a route configuration entirely at random would result in a poor choice of initial starting points. This could result in many long routes in the route configuration. Long routes are not bad per se; for example, it is reasonable to have long routes if the result is a reduction in costs, and the long route uses residual capacity on some links. However, choosing routes at random may not result in such long routes; the random process is quite likely to choose long and inefficient routes to carry some of the demands. The initial starting point chosen here is one in which all the shortest paths are found on the connection graph. These are then used as the routes for the demands. Depending on the problem, other starting points should be chosen based on the assumption that more or less of the links are used in the solution and the shortest paths on the graph chosen for the routes.

The time taken to find solutions using this algorithm is dependent on the size of the neighbourhoods. If the neighbourhoods are very large, then an exhaustive search of the neighbourhood can take a long time.

3.5.2 Simulated Annealing Algorithm

The simulated annealing algorithm was initially proposed by Kirkpatrick et al [KGV83] The algorithm was inspired by the simulating the annealing process in metals. In the annealing process, a metal is cooled from a hot (high-energy) state to a cooler (low-energy) state. The state of the metal maps to the state space in the optimisation problem and the energy of the metal in a particular state maps to the cost of a state. The annealing process is a somewhat random one, but is characterised by a very definite downward trend.

```
Step 1    Choose some initial starting point, $x_0$. Set $i = 0$. Set the initial cooling
          parameter $T = T_{\text{init}}$
Step 2    Choose candidate state from the neighbourhood of $x_i$ at random; call it
          $x_{\text{cand}}$
Step 3    Determine $\delta = c(x_i) - c(x_{\text{cand}})$
Step 4    If $\delta > 0$ set $x_{i+1} = x_{\text{cand}}$ , increment $i$ and go to step 7
Step 5    Determine $p$, the probability of accepting the state as the next state using
          $p = f(\delta, T)$
Step 6    Choose a random variable, $r$, from the distribution $U(0,1]$. If $r > p$ go to
          step 2
Step 7    If necessary, reduce the cooling parameter, $T$, according to the cooling
          schedule. If $T > T_{\text{final}}$ go to 2, otherwise end.
```

Algorithm 2: General form of Simulated Annealing algorithm.

In the simulated annealing algorithm as applied to optimisation problems, the algorithm moves from state to state in a similar random fashion. A neighbour of the current state is chosen at random. If this costs less than the current state, then it is accepted as the next state. This ensures that there is a downward trend over time. However, if it costs more than the current state then it is accepted as the next state with some probability. Clearly, this introduces the probabilistic nature of the algorithm. It also enables the algorithm to 'climb' out of troughs in the state space that would result in poor local minima, e.g. the m_1 minimum in Figure 3-6 above. The algorithm is more formally described in Algorithm 2.

The key parameter that controls whether or not the more costly solution is accepted as the next state is the *cooling parameter* that is determined by the *cooling schedule.* If the cooling parameter is high, then the probability of choosing a higher cost solution is higher. Conversely, if the cooling parameter is low, the probability of choosing a lower cost solution is low. As the algorithm progresses the cooling parameter decreases according to the cooling schedule; the probability of choosing a higher cost solution also decreases. The logic is that during the initial part of the execution of the algorithm higher cost solutions are frequently permitted, but at the final stages of the algorithm, higher cost solutions are very infrequently permitted. During the initial stages of the algorithm execution, the simulated annealing algorithm will resemble a somewhat random process jumping from one state to the next with a very small downward trend. As the algorithm progresses, the downward trend becomes more pronounced since the probability of moving to a higher cost state decreases. In the terminal stages of the algorithm, the algorithm behaves in a similar manner to the greedy algorithm. The algorithm usually terminates when the cooling parameter reaches some pre-specified limit. This is illustrated in Figures 7 and 8. It is clear that as the cooling parameter decreases the cost also decreases.

If an aggressive cooling schedule is used, i.e. one in which the cooling parameter is reduced quickly, then the algorithm behaves in a similar manner to the greedy algorithm. This algorithm should terminate more quickly than an algorithm in which a less aggressive cooling schedule is used.

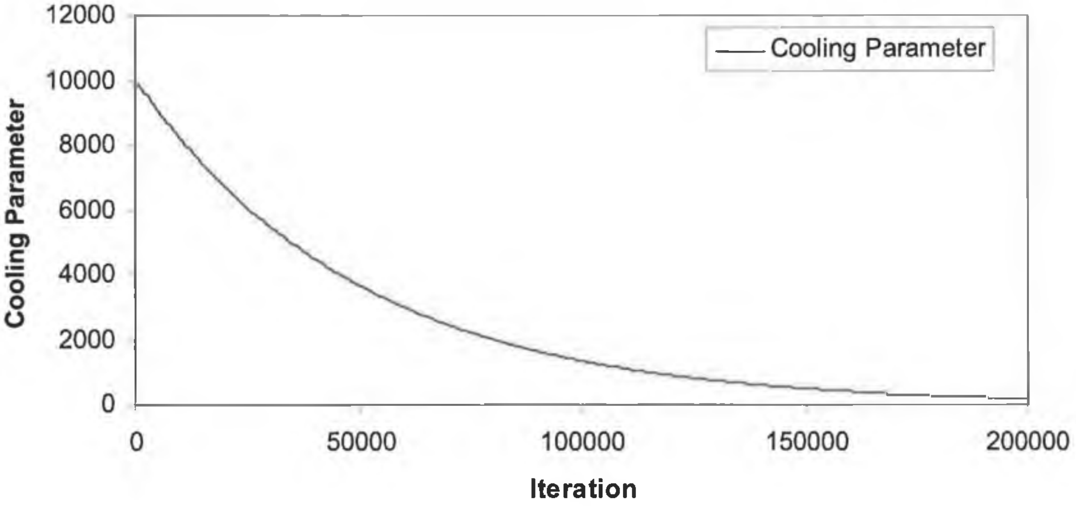


Figure 3-7: Cooling schedule used in a particular simulated annealing experiment.

Typically, in the simulated annealing algorithm the probability of choosing a higher cost solution is also dependent on the increase in cost. States that are marginally more expensive are more likely to be accepted as the next state than solutions that are considerably more expensive.

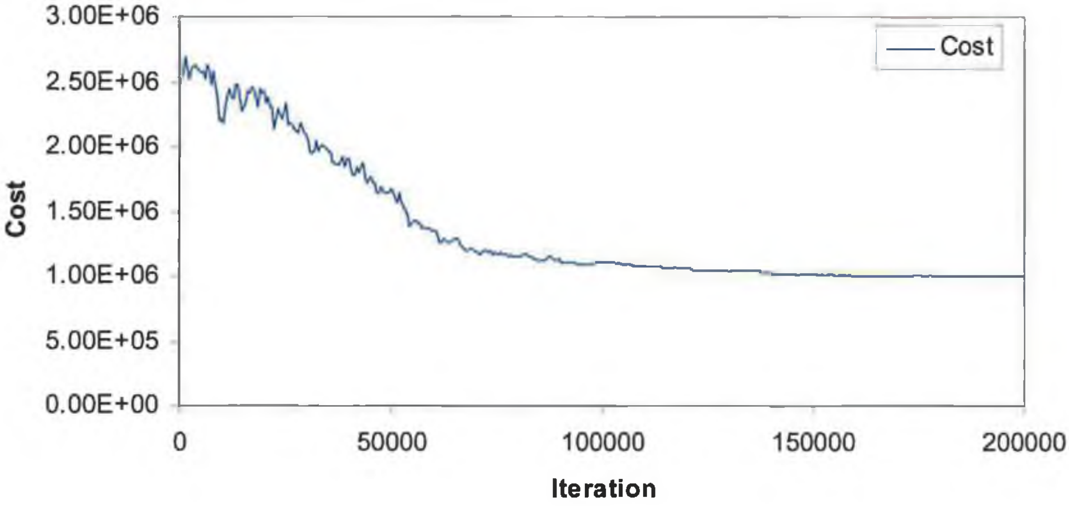


Figure 3-8: Variation of cost with number of iterations in a particular simulated annealing experiment.

The simulated annealing algorithm is shown in a general form in Algorithm 2. The cooling schedule and the way of calculating p are not explicitly stated. Typically, the cooling parameter decreases exponentially with the number of iterations, although many other decreasing functions could be used to control the cooling schedule. Similarly, a number of different functions can be used to determine the probability that a higher cost state is chosen as the next state. Again, a typical choice is one in which the probability of accepting the higher cost state is a negative exponential function depending on the cost difference between the two states. The probability is often calculated using $p = Ae^{B\delta T}$, where A and B are positive constants. Since p is only calculated when δ is negative and T always is positive, p is guaranteed to be between 0 and 1. The dependency on δ in this function ensures that the greater the increase in cost resulting from this state transition the less likely it is to be chosen as the next state.

3.5.3 Other Approaches

There are two other frequently used heuristics that can be applied to combinatorial problems – *tabu search algorithms* and *genetic algorithms*. Each of these are described briefly here. Techniques in which the paths available to route each of the demands are limited to some particular set of 'good' paths are also discussed.

Tabu Search

Tabu search algorithms attempt to incorporate memory into the solution process to exploit knowledge of previously visited states and previous state transitions. This is in contrast to the algorithms described above which are more primitive in the sense that they do not learn from previous moves. The objective of incorporating such knowledge into the solution process is to learn from previous efforts and guide the solution process through difficult parts of the state space so as to increase the likelihood of finding a good solution.

Tabu searches are so-called because they operate by making certain moves tabu or invalid at each iteration. For example, some neighbours in the neighbourhood could be tabu at each iteration. Moves that are tabu are typically moves that may cause the process to revert to a state that was visited earlier or a state that is similar to one that was visited earlier. By making a set of previous moves tabu the search process can be guided into parts of the search space that have not been visited before.

Tabu searches have been found to have good results for a number of problems. However, they are still quite a new heuristic approach to solving problems. Also, there are many variations on the tabu searches that can be used. Tabu search algorithms are not considered any further here. The intention here was to note it as another algorithm that can be added to the toolbox.

Genetic Algorithms

Genetic algorithms (GAs) form another class of algorithms that can be added to the toolbox of algorithms. GAs were inspired by selective breeding processes that take place in biology that can be used to give certain desirable characteristics to a population. The optimisation analogue of this is to 'breed' certain 'good' solutions to result in higher quality solutions.

The approach used by GAs in general is to choose some set of states from the state space. These states are then 'cross-bred' with the possibility of allowing mutation to introduce some extra randomness in the process. Bias is given to the more optimal states[7] in the cross-breeding process. This should result in a next generation that is, in general, more optimal than the preceding one.

While GAs have obtained useful results for many problems, they are not so directly applicable here for two reasons. Firstly, in the generic network problem described above, it is not obvious how to choose a set of basis states. These could be chosen entirely at random, but this would undoubtedly result in a very poor set of basis states and it would take quite some time to breed good solutions.

Secondly, GAs are particularly suitable to situations in which the states can be represented using binary variables, and such problems have been studied using GAs with some success. The problem under study here is not one that can be easily mapped to one containing a set of 0/1 decision variables.

The above two issues do not mean that GAs are not applicable to the generic problem. However, some effort would be required to apply GAs to this approach, and even then, it is not clear if the results would be good. This could certainly be an area for further research.

Path Pre-selection

In the above algorithms, it has been implicitly assumed that any non-cyclical route can be used to carry any demand. However, in many problems, many routes are bad candidates for routing a particular demand. Long routes, for example, are typically inefficient. Long routes containing one or more low capacity links are particularly undesirable. There may be other reasons why particular routes are undesirable, e.g. some routes may not be able to offer QoS sufficient for some demands.

Approaches in which the set of routes for a particular demand are limited to a set of pre-determined paths could also be considered. Determining the paths in advance can greatly reduce the size of the state space by eliminating large amounts of bad solutions. However, the size of the state space will still be very large – still far too large to search exhaustively.

If this approach is used, the problem of how to choose paths for each demand must be solved. This is not a trivial problem, and many authors have studied such routing

[7] More optimal in the sense of lower/higher cost depending on whether a minimum or maximum cost is sought.

problems. However, there are many common sense approaches that can be used to choose a reasonable set of paths for each demand. For example, the k shortest paths for each demand could be used, where k is some user specified parameter. In the case in which a number of paths are of equal length, then those paths containing the largest links could be chosen.

Determining the paths in advance can take some time, particularly if the problem is large: there is an overhead associated with choosing good paths. However, if the solution quality is in general better if specific paths are selected in advance, then it may be worth the overhead associated with choosing the paths.

Such path pre-selection is not considered any further here. However, it is noted as an area that appears to hold promise for future research.

3.5.4 State-space, Neighbourhoods and Algorithms used to Solve the Generic Problem

In this work, local search heuristics are used to solve the generic problem. As noted above, these local search heuristics require definition of a state space and a neighbourhood. Here, the state space and neighbourhood that are used in solving the generic problem are defined. Also, the particular variants of the local search heuristics that are used to solve the problem are described.

The state space chosen here is a way of representing all possible route configurations. Each state in the state space represents a particular routing for all of the demands on the network. Examples of the way that states are represented are shown in Figure 3-9.

Two states in the state space are considered to be neighbours if all of the demands are routed on exactly the same routes except one. Further, the demand that is routed differently is differs only by the insertion or removal or swapping of a node in a path. An example of neighbouring route configurations is shown in Figure 3-9. These route configurations apply to demands that are routed over the network shown in Figure 3-10.

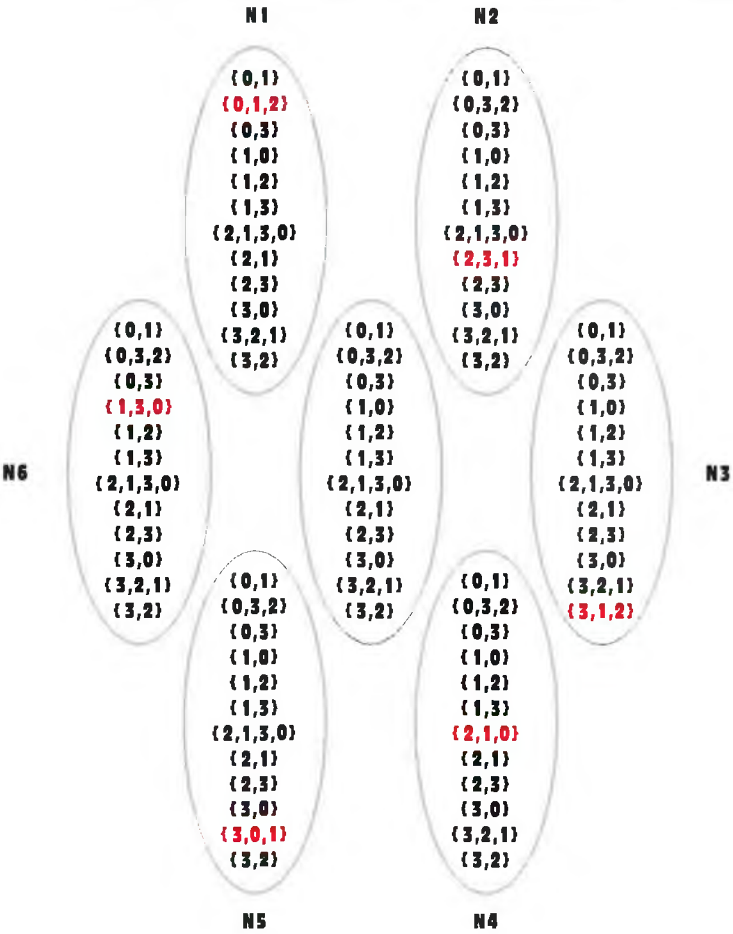


Figure 3-9: State with some neighbours. The state at the centre is the central state and the other states are neighbours of this state. This is not an exhaustive list of neighbours.

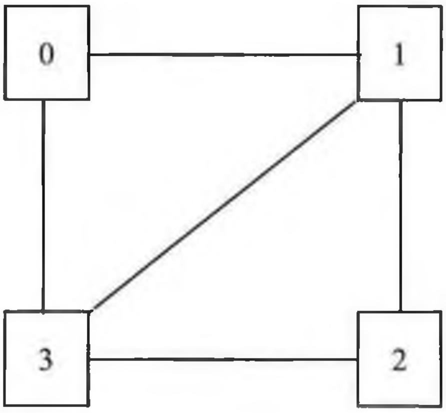


Figure 3-10: Network on which the routings in Figure 3-9 are based.

In Figure 3-9, all the demands in the neighbouring states are routed in the same way except one. The demand that is routed differently is shown in red. The different types of neighbours can be seen in Figure 3-9: some of the neighbours are obtained by inserting a node on this route, others are obtained by removing a node from the route and yet others are obtained by swapping a node on the route. In Figure 3-9, neighbours N2, N3 and N6 are obtained by inserting a node on a route; neighbours N1 and N5 are obtained by swapping a node on a route for another node and neighbour N4 is obtained by removing a node from a route.

One consequence of this choice of neighbourhood is that the majority of neighbours of any state will be states which are obtained by inserting a node into a path. This is because nodes can be inserted in a number of positions on the route, and often there are a few possible nodes that can be inserted.

The difference in the numbers of neighbours that are obtained via node insertion, node removal and node swapping is illustrated in Figure 3-11. There, the rerouting of a particular demand is depicted. The rerouting is dependent on the topology of the network – something that is omitted here. From Figure 3-11, it can be seen that there are 2 nodes that could possibly be inserted between nodes A and B and there are a further three nodes that could possibly be inserted between nodes B and C. This results in five alternate ways of routing the demand in which a node is inserted in the path. Only one node – node B – can be removed from the path since nodes A and C are terminations. There are two options for swapping node B with another node. In total, then, there are 8 neighbours of the state in which this particular demand is rerouted. Of these, the majority are routes that entail insertion of a node, a considerably smaller amount entail swapping nodes on the route and the smallest number of neighbours are obtained by removing nodes from the route. Clearly, the number of neighbours obtained from inserting a node on a route is the greatest. This point is more important for longer routes: for longer routes, there are more possibilities for inserting nodes and more nodes can be inserted. This means that as the length of the route increases, the fraction of ways to reroute the demand that entail insertion of a node increase. This has implications for solving the problem and the simulated annealing algorithm in particular.

One further point is worth highlighting here: the size of the neighbourhoods can be very large. For example, in the above analysis, 8 neighbours arise from rerouting of a single demand. Since there may be hundreds or thousands of demands in the network and it may be possible to reroute many of them, the number of neighbours of each state can easily become very large.

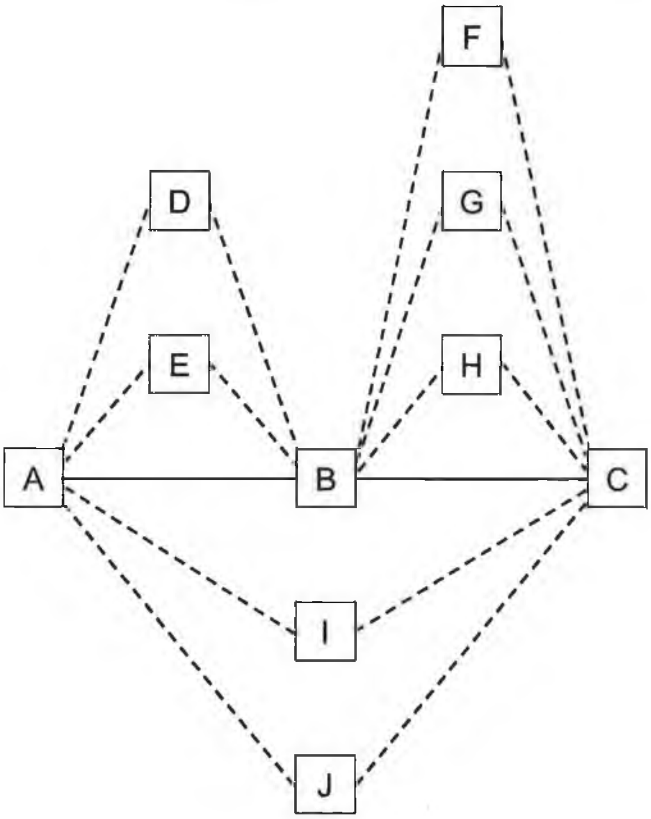


Figure 3-11: Illustration of the differences in the numbers of neighbours from the perspective of a single route.

The above state space and neighbourhood is used to run local search optimisation algorithms on the generic problem. In this work, two such algorithms are used: the greedy algorithm and the simulated annealing algorithm. Both of these are described at an abstract level above. Here, some more details are included on the implementations used here.

The greedy algorithm used here starts from a maximally connected network – all the candidate links are present in the network. A shortest path routing algorithm is performed on this network to obtain an initial routing for all of the demands. This is used as the starting point for the greedy algorithm.

During the operation of the algorithm, all the neighbours of the current state are costed. This is done by iterating through the routing of all of the demands, determining how many ways an individual demand can be rerouted and costing each of these reroutings. Thus, all the neighbours are costed and the lowest cost neighbour is chosen as the current state for the next iteration. The algorithm continues until the current state has a cost lower than that of all its neighbours.

The details of the simulated annealing algorithm do not differ very substantially from the algorithm described in section 3.5.2 above. The same initial starting point is chosen for the simulated annealing algorithm as is chosen for the greedy algorithm. A state is chosen at random and costed: it is accepted as the next state if it has a lower cost than the current state. Alternatively, it is accepted with a probability which is dependent on the resulting increase in cost and the cooling temperature.

One aspect of this simulated annealing algorithm which is not obvious is that if all of the demand is removed from a link, then the link is considered inefficient and is removed as a candidate link from the problem. This was done in order to reduce the number of links used in the solution, on the premise that reducing the number of links would have the effect of reducing the overall cost.

In this simulated annealing algorithm, the temperature is cooled at each iteration. The temperature is cooled in a geometric fashion.

The fact that a large majority of the neighbours of a particular state entail insertion of a node has implications for the simulated annealing algorithm in particular. Since this algorithm chooses a neighbour of the current state at random, the neighbour chosen will most probably be one that entails insertion of a node. As will be seen below, this has implications for the performance of the algorithm.

3.6 Conclusions

A flexible, abstract network optimisation framework has been described. The motivation for the framework has been given; viz. to reduce the time required to obtain solutions to specific network optimisation problems. The layered nature of the framework was described and the functions of the different layers were identified.

A generic problem is the crux of this framework. The generic problem was described in detail here: the sets of inputs and outputs of the problem was first described, followed by a mathematical formulation of the generic network optimisation problem. Some short example applications of the framework were then described.

Lastly, a number of solution approaches that can be used to solve the generic network optimisation problem were discussed. These solution techniques are techniques that are quite generally applicable and arise in the combinatorial optimisation domain. The use of some of these solution techniques will be investigated below.

CHAPTER 4 ENTERPRISE NETWORK DESIGN PROBLEM

4.1 Introduction

Here, a specific network design problem is discussed. The problem is motivated and the application of the network optimisation framework to obtain a solution to the problem is demonstrated. The specific problem considered here is one that arises in the context of enterprise network design.

The purpose of an enterprise network is to facilitate internal communications within the enterprise. As such, the bulk of the traffic carried on the network consists of the intra-enterprise traffic generated within the organisation. However, some of the traffic on the enterprise network may be destined for locations outside the organisation: some application traffic could be carried on the network to some point close to the destination and then it could be switched onto another network. This would be done to effect cost savings. This is the case for so-called 'break-in/break-out' voice traffic on some private networks – the traffic is routed to the location on the enterprise network that is closest to the (off-net) destination and is switched to the public network at that point. So, some traffic on the enterprise network may not be intra-enterprise traffic per se.

Enterprise networks often grow in an unplanned, ad hoc way. Extra resources are added to the network as and when necessary. There are a number of reasons for this, including:

- network usage data may not be readily available;
- the network is not the core business of the enterprise and hence it may not have the experience or expertise to plan and optimise the network;
- often the enterprise does not even have a good inventory of its network resources.

However, as noted in chapter 2, Lloyd-Evans [Llo96] estimates that 10-20% savings can be achieved by optimising packet-switched networks. Since the majority of enterprise networks are packet-switched networks, this is likely to be applicable in this case. Consequently, enterprise networks are an interesting candidate for optimisation.

Here, it is assumed that the enterprise wishes to maximise the use of the enterprise network resources. Consequently, cases in which there is significant traffic aggregation are considered; i.e. the enterprise uses the enterprise network to carry diverse traffic types. This is in contrast to a situation in which the enterprise may have separate networks for different applications. Traffic aggregation in this manner may result in cost savings since only one network needs to be operated and maintained.

The specific problem considered here is to determine how to add capacity to an existing network at minimal cost to accommodate the demands of the network users. As the problem is stated, it is implied that there is some existing network and the objective of the problem is to determine how to add capacity to it. However, the problem also encompasses the case in which there is no existing network – the green-field network design problem. This can be considered to be the special case in which the existing network consists of zero-capacity links. The solution to the problem is a route configuration – how the demands are routed on the network – and a set of costs associated with this configuration. This problem is discussed at length throughout the remainder of the chapter.

The chapter is structured as follows. A detailed description of the problem is given in the next section. This is followed by a formulation of the specific problem model. This is mapped to the generic problem in the following section and issues that arise in the mapping function are discussed. Some example problems are then given to illustrate the use of the framework to solve this problem and then the chapter is concluded.

4.2 Problem Description

Here, a more detailed description of the problem is given. This entails a discussion of the demands on the network – the demands of the users of the network. This is followed by a discussion on how the network may be realised, which is followed by some comments on the costs of implementing the network. A short note on network evolution considerations is included at the end of this section.

4.2.1 Enterprise User Demands

The network must be designed to support the applications the organisation uses; hence it is important to consider what applications the enterprise uses when designing the network. The applications used in enterprise networks are discussed next.

Applications Used on Enterprise Networks

While every organisation is different and uses different applications, many applications can be broadly categorised using the following categories:

- *Interactive voice* – voice communications have traditionally been an integral component of private networks. Voice is still an essential form of communications, particularly in those cases when the communication needs to be one to one and interactive, and, as such, will continue to form a constituent of any large organisation's private network traffic.
- *Interactive video* – interactive video refers mainly to one of two applications: videoconferencing applications or one-to-one videotelephony type applications. The former usually have a number of parties involved and usually take place in a particular videoconference suite, while the latter are usually one-to-one communications and take place between user desktops.
- *File download* – file download applications are characterised by downloading a particular file from a server; what is done with this file is not important. Examples of file download applications include FTP and can also include applications such as video-on-demand.
- *Collaborative working applications* – collaborative working applications are those in which a number of parties are simultaneously working together on the same material. An example of a collaborative working application could be a document editing suite in which many users can simultaneously and remotely comment on a particular document during an editing meeting.
- *Transaction based applications* – these form a very important classification of applications, since there are very many mission critical applications that fall into this category. Many applications based on database querying can be considered to be transaction based and web downloads can also be considered to be transaction based.

Rapid growth in computing power and communications equipment mean that more sophisticated applications are becoming possible and that the above classifications may require some modifications. For example, applications which incorporate elements of a number of the above classifications can be envisaged. For now, however, they form a useful, although not exhaustive classification of applications that run on enterprise networks.

The network designer needs to ascertain which applications the private network will support and the demand generated by these applications. The network designer also needs to know what QoS is required by the different applications and must design the network such that the appropriate QoS is delivered to the applications. Also, it may be necessary to take into account interactions between different applications – the performance of some mission critical applications could possibly be affected by traffic from other applications if the no precautions are taken to ensure that mission critical traffic is protected.

In general, the application level quantities can be difficult to measure. The application level demand is often not easy to measure. Also, the data generated by the application can be difficult to model: this is especially true of video traffic (see [RMV96,BCMM94] for more discussion on this). It may also be difficult to quantify the QoS required by the applications – especially in terms of network level parameters such as delay and loss. Also, if the network is to be designed to accommodate future growth in the network, then the problem of forecasting growth in the use of different applications arises. These issues are beyond the scope of this work.

For the purposes of this study, only voice and data traffic are considered. Voice traffic is considered to originate from a voice terminal and data traffic is any traffic that originates from a computer terminal. In this case, the data traffic can be quite heterogeneous: it can constitute traffic from many different applications. Voice and data traffic have historically been considered different and consequently this differentiation is not unnatural. As more sophisticated applications with different requirements of the network are developed, this simple classification will no longer suffice. For this problem, however, it is assumed to suffice.

The question then is how to realise the network to accommodate the voice and data traffic demands.

4.2.2 Network Realisation

The network can be realised in many ways. Here, issues relating to the way that the network is realised are discussed. This includes addressing such issues as the network architecture being used, the network components used and the services used to implement the network to meet the demands of the users.

Network Architecture

In most cases, the most cost-effective network design is one of a hierarchical nature. Even without much planning private networks have tended to evolve to a hierarchical architecture for cost reasons.

An example of an hierarchical network design is shown in Figure 4-1. In this example three levels exist in the hierarchy – headquarters, the regional offices and the local offices. The local offices are typically small offices and generate small amounts of traffic. These are homed on one or more of the regional offices depending on how important connectivity is and what options are available to cope with failure of the communications between the local and regional offices. The interconnects between the regional offices then form the backbone network: these interconnects are typically high capacity. There may be high connectivity in the backbone network for efficiency and reliability reasons.

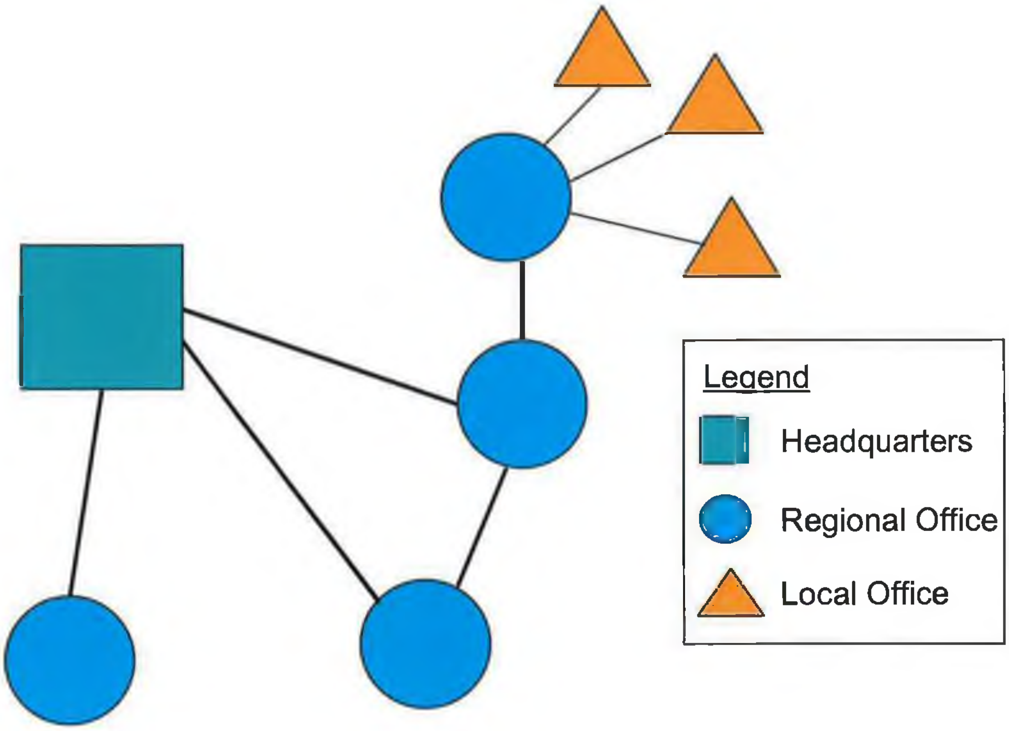


Figure 4-1: Example of hierarchical network.

Since this architecture is typical of private networks, the networks used in this study are assumed to be of this hierarchical nature. Designing such networks is not a trivial task.

The most difficult problem to solve when designing such networks is how to determine the network hierarchy; specifically, which nodes are homed on which concentrator nodes. Once this is solved, the mesh network design problem for the highest level nodes must be solved.

Since determining the network hierarchy is not a simple task, some comments are merited here. It is not always obvious which nodes in the network should be the regional offices. Often the regional office is not much different than the local offices. For example, in the case of a bank, the regional office may be a large branch that happens to have space for the network equipment. There may not be any on-site staff to operate this equipment; this could be performed remotely via the network. Sometimes, it makes sense to have the regional concentrator located in some location that may be a building that has the specific purpose of housing the concentration equipment i.e. it doesn't have to be a branch office. When choosing the concentration points there may be other factors that influence the decision, such as the availability of service, or the proximity to the nearest operator's engineering office so, for example, if there were some problem with the network, then a maintenance team could be on-site within some short period of time.

If there are few constraints on the choice of the concentration points, then determination of the concentrators in particular and the network hierarchy in general can be formulated as an optimisation problem, the objective of which is to determine the set of node locations that minimise the overall network cost. This is a complex problem in itself and is not considered here.

The emphasis in this work is on the mesh network design problem. This is partly because the generic problem in the network optimisation framework can only solve mesh network optimisation problems, but there are other reasons for studying such a problem. The mesh network can form the backbone of the enterprise network. Also, the mesh network can consist of high capacity links and as such can be costly: hence, there is an opportunity for savings to be made. In large enterprises, the backbone network can be large and hence the design problem can be challenging.

Network Components

Many different network components are used in private networks. The type of components that constitute the network is dependent on the type of traffic the network must carry. Here, data and voice traffic are considered. It is assumed that each node has voice and data communications requirements, and that it needs to communicate with some or all of the other nodes in the network.

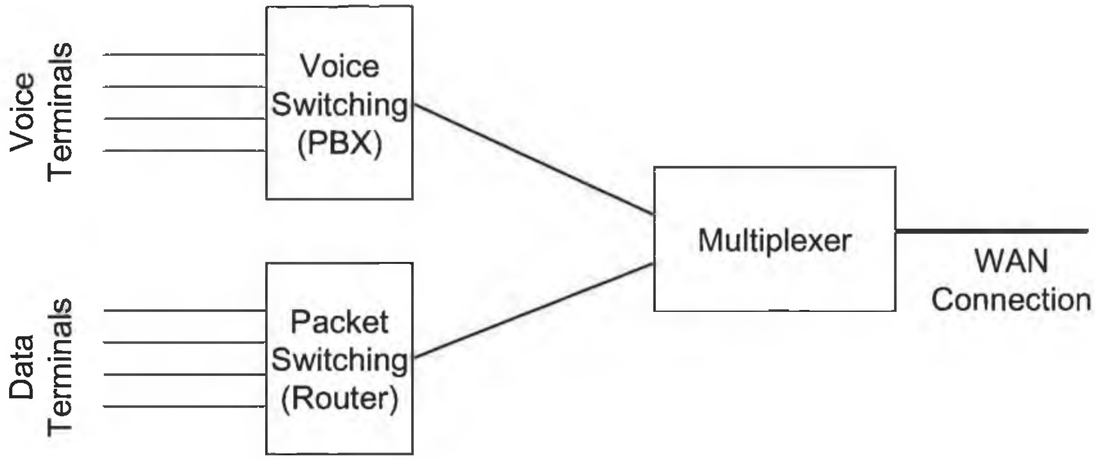


Figure 4-2: Logical components required to access WAN in customer premises.

A number of different functionalities are required in this scenario: functionality to interface with voice terminals and switch voice calls, functionality to interface with data terminals/LANs and perform packet routing functions and functionality to control access to the *Wide Area Network* (WAN) resources. These functionalities can be implemented in different network components, or alternatively, a single network component can implement all of these functionalities. Whether or not all of these functionalities can be implemented in a single unit is dependent on the size of the installation: in larger installations these functionalities would typically be realised in different systems. In any case, the specific systems used to realise these functionalities are not considered here. Rather, it is assumed that these functionalities are realised somehow. An illustration of the logical components required at each customer premises is shown in Figure 4-2.

The functionality of the multiplexer is an important issue in the design of such networks. There are two fundamentally different types of multiplexer that can be used in this way; the first operates using *Time Division Multiplexing* (TDM), while the second uses packet switching.

The TDM approach is the more established approach for such multiplexers and there are many TDM based multiplexers on the market. Companies such as Timeplex, Tellabs and Lucent Technologies have been selling TDM based multiplexers for many years. The TDM based approach involves division of the bandwidth on the wide area link into a number of fixed bitrate channels. All of the traffic using the channel is given one of the channels; even traffic which is inherently variable such as data traffic is given some fixed rate channel. Depending on the amount of data traffic being generated, the channel may be fully utilised or highly under utilised. TDM based solutions are only applicable in situations in which the WAN is realised using fixed bitrate channels, as opposed to packet-based interfaces to WAN connections.

The packet-based approach is quite different. Using this approach, the incoming traffic is broken down into packets at the ingress to the multiplexer, and then routed to the appropriate output port. In a packet-based multiplexer, all the data is transported through the multiplexer in packet format. It is then transmitted over the WAN in packet format. The WAN can be realised using leased lines or, alternatively, a packet based service such as ATM or *Frame Relay* (FR) can be used to realise the WAN.

Using packet based communications is, in principle, more efficient, since the network resources are only used when there is information to be transmitted[8]. This is in contrast to channel based solutions in which bandwidth is always reserved for specific applications. Consider, for example, a single link which is part of a private network; this could be a leased line. If the capacity on this link is divided amongst the different applications contending for the link resources using TDM, then there is a very hard division of the link capacity: some of the link is **always** reserved for voice traffic, even though there may be no voice traffic using the link. This is in contrast to a scenario based on packet switching in which capacity unused by one application can be used by another application. Thus, packet switching is inherently more efficient.

Network Technologies

A number of different technologies can be used to implement the network. The private network can be implemented using leased lines, FR or ATM or some mixture of these technologies; also, some of the voice traffic could be carried over the *Public Switched Telephony Network* (PSTN) or a *Voice-based Virtual Private Network* (V-VPN)[9]. Each of these technologies is discussed below.

[8] However, this does not necessarily mean that packet-based communication is cheaper; this is dependent on the tariffing scheme.

[9] This is usually simply termed a VPN. However, it is called a V-VPN here to differentiate from the type of VPN used in chapter 5. The V-VPN is a sophisticated bulk voice service offered by telecom operators.

Private leased lines are the most widely-used technology for implementation of a private network. They generate huge amounts of revenue for operators. They are well established and customers are very comfortable and familiar with the use of leased lines to meet their communications needs. A leased line acts as a simple transparent interconnect of some specified, fixed capacity between two locations. The customer can then choose to use this interconnect in whatever fashion it sees fit. In particular, the customer could choose to implement the WAN using packet switching or use a channel based approach to divide the capacity on the WAN between different applications.

FR and ATM services differ fundamentally from leased line services in that they are packet based. This means that the WAN must be implemented on a packet-switched basis – it is not possible to completely segregate the capacity for the different services using a channel based approach.

FR services are characterised by a *Committed Information Rate* (CIR). The CIR is the rate that the operator will ensure to the customer – if the customer does not exceed this rate, then the traffic is assured delivery. However, if the customer exceeds the CIR, the operator will try to deliver the excess traffic.

In many cases, customers use FR services with no CIR – they do not require any throughput commitment from the operator. The FR service that they obtain from the operator is a best-effort service. Since FR networks typically have a substantial amount of resources, the QoS perceived by such users is usually quite reasonable. Some users, however, do require some throughput assurances to ensure that their applications receive the desired QoS. The users in this work are assumed to fall into the latter category and obtain FR service with a specific CIR.

ATM services are somewhat different. A number of different ATM service types exist. In particular, ATM supports both *Constant Bitrate* (CBR) and *Variable Bitrate* (VBR) services. The former are characterised by a single peak rate; the latter are a little different since the customer can (roughly) specify peak and mean rates for the connection and the customer is allowed (within some limits) to transmit at arbitrary rates between these peak and mean rates. ATM CBR services are the ATM analogue of leased lines. Unlike the FR services, the customer will only obtain service up to this

It has functionality to implement a private numbering plan as well as advanced voice-based functionality including call forwarding, call back, voice mail etc.

rate. ATM VBR services are a somewhat more complex to deal with and hence they are not considered here for implementation of the private network interconnects.

Leased lines, FR and ATM offer permanent connectivity between customer sites. For this reason, they are very suitable for data communications. They can also be used to carry voice traffic. However, voice traffic imposes some stringent QoS constraints on the network. Consequently, when implementing the network some care must be taken to ensure that the desired QoS is delivered to the voice traffic. In the TDM leased line case, capacity is reserved for the voice traffic. Thus, QoS is assured to the voice traffic. In the packet based scenario, this is not necessarily the case. In the packet based scenario implemented using leased lines, data packets could cause voice packets to be queued resulting in delays for the voice packets and a degradation in perceived voice quality. A similar effect could be observed in the case of a network realised using ATM CBR connections. In both of these cases, the solution is to prioritise voice packets in the network.

The FR case is a little more complex, since the amount of delay introduced in the network is unknown. Also, the amount of resources available to the customer is unknown. Here, it is assumed that the customer does not exceed the CIR. In this way, all the traffic can be assured of getting to the destination and the voice traffic in particular will be assured of getting to the destination. The FR scenario could also be realised more effectively if the FR network used the priority bit in the FR header to differentiate between low priority and high-priority traffic. If the customer marked traffic appropriately to indicate which traffic was most important – the voice traffic in this case – then the operator could use these markings and prioritise the high priority traffic accordingly.

The PSTN can be used to accommodate the voice traffic. In this case, the organisation would pay for all inter-office calls on a usage basis. However, for a large enterprise network, it is quite likely that there will be a very substantial amount of inter-office traffic and the resulting costs could be quite high. Also, the costs incurred by the use of the PSTN are quite unpredictable.

One way to reduce these costs is to use V-VPN services offered by operators. Such services usually result in cost savings for large customers because they are considered to be bulk users of the service and they receive (in some sense) bulk discounts. V-VPN services also have the advantage of providing advanced call related features as part of the V-VPN package – voicemail, callback, call forwarding, etc. – via the public network and consequently, the organisation does not need to purchase and maintain equipment to support such services. V-VPN services still result in variable costs, although the variation is smaller than that of the PSTN.

Using a V-VPN for voice traffic and a leased line/FR/ATM network for data traffic is a very clear segregation in services. Here, however, the emphasis is on traffic aggregation rather than traffic segregation to achieve savings. Consequently, the use of V-VPNs to carry voice traffic is considered no further here.

Given that the emphasis is on traffic aggregation then, the objective is to design the private network such that the overall network costs are minimised. Any of the above interconnect services could be used. The interconnects may not have sufficient capacity to handle all of the voice traffic, especially when there is a peak in voice traffic. Consequently, it make sense to permit voice traffic to overflow onto the public network, i.e., the private network is used for data traffic and most of the voice traffic and the PSTN is used for voice traffic when there are insufficient resources on the private network.

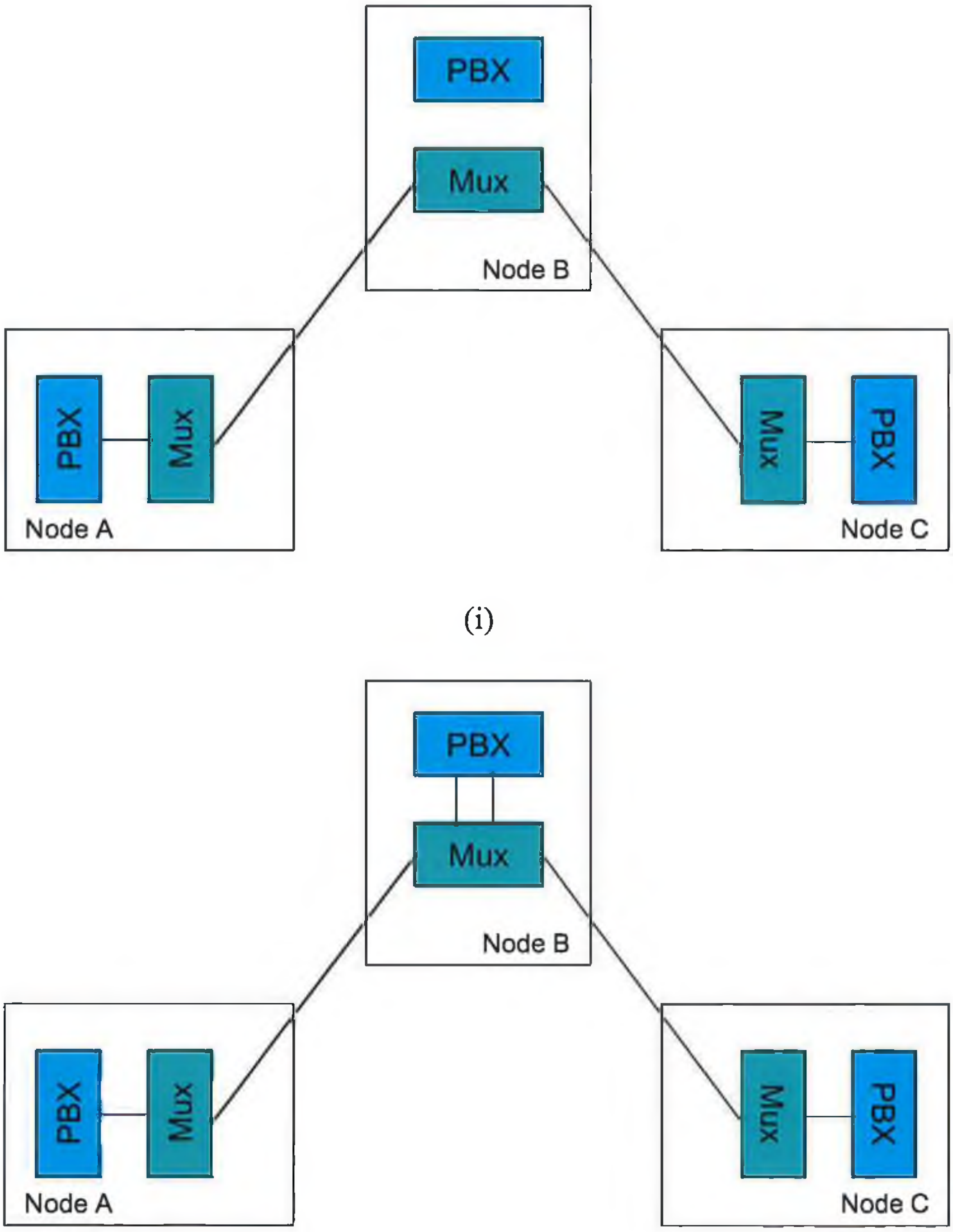
Network Configuration

The mesh network design problem considered here reduces to one of considering how the demands should be routed on the network. This is essentially a network configuration problem. Issues pertaining to configuring the network are discussed here.

Two separate network implementations are considered here: a channel based network implementation and a packet based network implementation. In the former, there is a clear separation between resources reserved for voice traffic and those reserved for data traffic; some channels are reserved for voice traffic and some are reserved explicitly for data traffic. In the latter, all the resources are shared. Both of these scenarios are considered in more detail below.

In both implementations, all of the data traffic is multiplexed on the WAN; specific WAN capacity is not reserved for **data** communications between specific node pairs. In today's networks, IP traffic is very much the dominant data traffic type. Hence, it is not unreasonable to assume that all of the data traffic on the network is IP traffic. IP can also be used to encapsulate other traffic types [RFC2661] so other traffic types can be carried on the network inside IP packets.

If all of the data traffic on the network is IP traffic, then IP routing mechanisms must be used to effect a particular network configuration. OSPF is the most common intra-domain routing protocol and it is assumed that this is used here. As discussed in chapter 2 above, OSPF does not permit arbitrary choice of routes for the demands although it does provide some level of flexibility. Here, it is assumed that this level of flexibility is sufficient to implement the desired network route configuration.



(i)

(ii)

Figure 4-3: Difference between two ways of switching voice calls. In (i) the call is switched through the multiplexer; it is not switched in the PBX at the intermediate node B. Conversely, in (ii) the call is switched in the PBX at in node B.

If the voice traffic is carried on separate channels, two options for configuring the network are possible. In the first, the voice channels are terminated at adjacent *Private Branch Exchanges* (PBXs); in the second, the voice channels are terminated at non-adjacent PBXs – the multiplexing equipment performs a function similar to a cross-connect and the call is not routed through the PBX at the transit node. Both of these situations are illustrated in Figure 4-3.

These two approaches differ in the way that resources are shared in the network. In the first approach, depicted in Figure 4-3 (i), resources are reserved exclusively for traffic between A and C on links AB and BC. The traffic is not switched at the PBX at node B. In this approach a mesh of resources reserved for exclusive use by a particular node pair – a logical network – can be implemented. The logical network is implemented using the multiplexers. In the second approach, depicted in Figure 4-3 (ii), no resources are reserved for exclusive use by any node pair. In this case, traffic between nodes A and C is switched through the PBX at B. More generally, all traffic is switched at intermediate PBXs as appropriate. No logical network is implemented in this case.

The two approaches can differ in terms of cost and efficiency. For small networks, the latter implementation is more efficient, since voice channels are at a premium and the best use of these channels is made possible if there is full sharing between the voice channels. The alternative scenario can be more economical in larger networks. In this case, switching does not need to be performed at intermediate nodes and consequently, the cost of the intermediate nodes is less. Specifically, the intermediate nodes need less voice call terminations which results in a lower cost node. Also, the total amount of switching to be performed at the node is less and consequently, the overall cost of the node may be less.

If the voice and data traffic are decoupled in this manner, they can be routed differently: a different route can be used for the voice traffic than that of the data traffic. This can permit the network to be configured more efficiently. However, it is assumed below that the voice and data traffic are routed together.

The performance of the network in which the voice channels are terminated at intermediate PBXs will be better than that of a network in which they are not terminated at the intermediate PBXs. Consequently, if it is assumed that the voice channels are not terminated at intermediate PBXs in the design process, then the performance of the resulting design will be better than predicted by the design approach. This assumption also makes the design problem more tractable: otherwise some of the techniques described in the section 2.4 would have to be applied to determine how many voice channels are required on each link. Hence, it is reasonable to assume this in the design process, although it may result in networks that are of slightly higher cost than required.

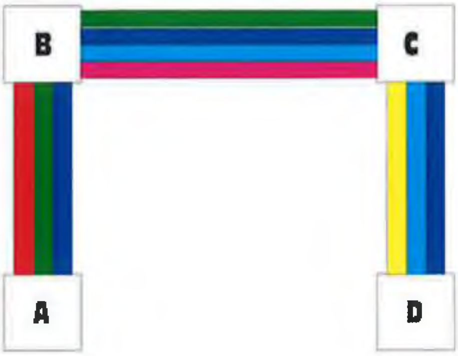


Figure 4-4: Illustration of how the capacity of the interconnects may be divided between the traffic for each of the node pairs.

If the voice traffic is packetised and transported in the same manner as the packet data, then the WAN capacity does not need to be segregated at all. In this case, the multiplexer is not needed; the router can be connected directly to the WAN link. The data networking routing protocols can be used to perform routing in this network. This scenario is illustrated in Figure 4-4.

If this approach is used, then the voice traffic must be prioritised over the data traffic. This simple prioritisation can be done using, say, the *Type of Service* (TOS) header field in the IP packet header. This enables routers to differentiate between the different traffic types and to give priority to the voice traffic to facilitate timely delivery of voice traffic.

4.2.3 Network Costs

The network costs fall into one of three categories:

1. Capital expenditure – these are once-off payments typically paid to purchase equipment;
2. Fixed recurring costs – these are costs that must be made periodically. Examples of this type of cost include service subscription/service access costs;
3. Usage based costs – these are variable costs that are incurred based on usage of a service. If usage is high, then costs will be high and if usage is low, then costs will be low.

Capital expenditure cost can be incurred when purchasing or upgrading equipment. In this problem, the capital expenditure costs would only be associated with upgrade or purchase of new node equipment.

Fixed, recurring costs are costs that are incurred on a periodic basis. These are often incurred for subscription/rental of services. Examples of such costs would be annual line rental costs or annual subscription to FR/ATM services.

Usage based costs are costs that are incurred through usage of a particular service. For a voice based service, usage based costs would be costs incurred for making an individual call. For data-based services, usage based costs could relate to the amount of data that traverses an interconnect, say[10]. Due to their nature, there is some variability in usage based costs and it is sometimes difficult to predict them very accurately.

Note that these categorisations are not rigid. Capital expenditure may be financed by loans, or equipment may be purchased under some kind of hire-purchase arrangement. In these cases, the capital expenditure could be viewed as a recurring cost. Similarly, recurring costs could incorporate some usage component: for example, a customer may pay for V-VPN service which includes some amount of usage over the billing period.

Cost functions for these different services can be quite complex. This is the case for the interconnect services – leased lines, FR or ATM. In general, they are dependent on the capacity and may also have a distance dependence. The costs for the interconnect services can be quite non-linear but they always increase with increasing capacity. Stepwise cost functions are good examples of the type of cost function that is common for such services.

Interconnect services may have an access component that further complicates the cost function. In many cases, service may not be available at the nearest exchange. Consequently, the customer may have to obtain some permanent connectivity to the nearest POP. This may be implemented using leased lines. Thus, the costs of implementing interconnectivity between two nodes may consist of service access costs as well as service subscription costs, further increasing the complexity of the costs.

[10] Usage based costs for data services could also be incurred for accessing content-based data services. These type of services are not considered here.

Predicting the usage based costs can also be non-trivial. Take, for example, the PSTN traffic. The usage based costs are based on the amount of usage of the service: for voice telephony, this can depend on the amount of calls made and the duration of each of the calls. There can be some error in predicting these.

The objective here is to determine the lowest cost network configuration. Implicitly, the notion of comparing costs is assumed. As can be seen here, there are different types of costs that operate over different timescales. Some care must be taken when comparing them. For example, it is difficult to objectively compare a solution which has a high capital expenditure and a low recurring cost with one that has a higher capital expenditure and a lower recurring cost. This is quite a standard problem, which can be solved by normalising the costs with respect to some time interval.

As is seen above, the cost functions for the services used to implement the network are quite complex in general. Hence, it makes sense to use a flexible framework in which the cost functions can be quite arbitrary.

4.3 Specific Problem Model

The specific problem model is discussed next. First, the problem is discussed in terms of the problem inputs and outputs. Then the problem is formulated more rigorously.

The inputs to the problem are:

- the set of nodes;
- the set of voice and data demands;
- the capacity required to carry a voice call on the private network;
- the set of candidate links;
- the set of link cost functions;
- the set of costs functions for the public network traffic.

The voice demands are specified in terms of the number of call arrivals over some time period. The data demands are specified in terms of capacities. The cost functions are both specified in terms of cost over some time period.

The voice traffic can be carried on the private network or can be shed onto the public network. Not all of the voice traffic will be carried on either the public network or the private network – some of the voice traffic will be carried on the public network and some of the voice traffic will be carried on the private network. The capacity required to accommodate a voice call can be used to determine how much capacity a number of voice trunks requires on the private network.

An alternative formulation of this problem could consist of data traffic specified in terms of application level demands and application level QoS measures which could then be translated into some capacities for the data demands. This problem is not essentially different, but would involve a different mapping function – the mapping function would contain functionality to map the application level characteristics into single parameter for the generic problem.

The problem outputs are:

- the network configuration;
- the amount of traffic that is carried on the public network;
- the amount of traffic carried on the private network;
- the set of services that constitute the private network and
- the overall cost.

The specific problem model is constructed such that it can be applied to situations in which the private network is implemented using all of the technologies mentioned above. From the perspective of the specific model, all of the technologies that can be used to implement the interconnect appear the same. The difference between the technologies manifests itself in the link cost function.

4.3.1 Formal Problem Model

Define the following:

- N : the set of nodes in the network;
- L : the set of candidate links;
- $G(N,L)$: the graph of the network of possible links;
- c_l : the capacity of link $l \in L$;
- D : the set of demands;

- (o_i, p_i, h_i, m_i): set characterising demand $d_i, i \in 1 \ldots |D|$ – o_i and p_i are the source and destinations nodes respectively; h_i is the intensity of voice traffic associated with the demand and m_i is the capacity of the data traffic;
- v: the capacity required to carry a voice connection on the private network;
- R: a routing for the demands on the network;
- Q: the set of voice channel reservations for each node pair;
- q_d: the amount of voice channels reserved for demand $d \in D$;
- $I_d^l(R)$: indicator function indicating whether or demand $d \in D$ is carried on link $l \in L$ – the function has a value of 1 if the demand is carried on the link or 0 otherwise;
- $x_l(R,Q) = \left\lceil \sum_i I_{d_i}^l(R)(m_i + q_{d_i} v) \right\rceil - c_l$: the extra capacity that must be added to link l under routing R with voice channels specified by Q;
- $f_{\text{pri}}^l(x)$: the cost of adding x units of capacity to link $l \in L$;
- $f_{\text{pub}}^d(q_d)$: the cost of carrying the public voice traffic for voice demand $d \in D$;
- $F_{\text{pr}}(R,Q) = \sum_{l \in L} f_{\text{pri}}^l(x_l(R,Q))$: the cost of the private network;
- $F_{\text{pu}}(Q) = \sum_{d \in D} f_{\text{pub}}^d(q_d)$: the cost of the use of the public network resources;
- $F(R,Q) = F_{\text{pr}}(R,Q) + F_{\text{pu}}(Q)$: the overall cost.

The problem is to find

$$F^* = \min_{R,Q} F(R,Q)$$

In general, the problem of determining the minimum costs is difficult to solve. The problem complexity is dependent on the nature of the cost functions for the public and private network resources. Also, the amount of overflow traffic is dependent on the arrival rate of the voice traffic and the amount of calls that can be accommodated on a particular link. The latter is determined using the non-linear Erlang blocking function:

an extra non-linearity is introduced into the problem. So, even for straightforward cost functions, the problem is difficult to solve.

A report on solution techniques that were used to solve this problem is given in [MBC99] and related work was described in [MBC98]. Girard also formulates a very similar problem in [Gir90], but there the scope is limited to design problems in which the costs of both the private network and the overflow traffic are linear. An alternative approach is used here, which uses the generic network design framework of chapter 3.

4.4 Mapping to the Generic Problem

In the generic problem, there is a set of demands characterised by a single value and a set of individual link cost functions. This differs a little from the specific problem in that the specific problem demands have both a voice and a data component and there are cost functions associated with the traffic carried on the private network and voice traffic shed onto the public network. The mapping function must reduce the parameters in the specific problem to appropriate parameters for the generic problem. Also, the demands and link capacities in the specific function are bidirectional, while their generic problem counterparts are unidirectional. The mapping function must also take this into account when performing the mapping.

4.4.1 Determining the Demands

The approach used here to split the voice demands between the public and the private network is a simple one. The fraction of voice traffic shed onto the public network is defined. This implicitly defines the fraction of traffic carried on the private network; the amount of voice channels required on the private network can be calculated from this. The required number of voice channels can be solved by determining q as the solution to the equation

$$q = \left| E^{-1}(h,b) \right|$$

where h is the intensity of the voice demand, b is the fraction of traffic to be carried on public network and $E^{-1}(\cdot)$ is the inverse of the Erlang blocking function. The problem can be solved by performing an inverse Erlang calculation, but in this case, a more direct, iterative process is used to find the solution. At each iteration, the blocking probability is determined for the current number of voice channels. The number of voice channels is increased until the blocking probability is sufficiently low. Then the required number of voice channels has been reached. Typically, the number of voice channels required is small hence the number of iterations is small, so this approach is feasible.

One problem with this approach is that it is highly unlikely that the optimal solution to the specific problem can be found using this approach. It is very unlikely that the amount of blocking associated with each demand will be approximately equal for all of the demands. However, the approach can find a reasonable solution and hence it is useful.

An alternative approach which is arguably more likely to find a better solution is to assume a direct link is used between each node pair and determine the optimal amount of voice traffic that is carried on this direct link. This idea is a little more interesting, and for this reason, is expanded on here.

Using this approach, the size of the demand for each node pair is calculated by solving the following optimisation problem for each node pair.

$$\text{Find } z = \min_{q} f_{\text{pr}}^{l'}(m_i + qv) + f_{\text{pu}}^{d'}(q)$$

where l' and d' are the link and the demand between the node pair respectively. Thus, the number of voice trunks between each node pair can be determined.

The solution to this problem can be found in a number of ways. As with the other approach, if the intensity of the traffic is low, it is possible to iterate through a number of different values of the voice trunks on the direct link and choose that number that results in the lowest cost.

It would be interesting to explore this idea further. In particular, it would be interesting to see the impact it has on the results obtained.

Once the number of voice channels for each demand has been determined, the set of demands to be input to the generic problem can be constructed. For each demand, the triplet $d_i' = (o_i, p_i, r_i)$ can be generated, where $r_i = m_i + q_{d_i} v$, and the set of demands $D' = \bigcup_i d_i'$ can be constructed.

Note that this fixes the amount of overflow traffic that is generated by the network. Once this step is performed, the problem focuses on how to determine the private network: the costs associated with the public network traffic are defined once the overflow traffic is determined.

4.4.2 Determining the Link Cost Functions

The link cost functions must be mapped from those in the specific problem to those in the generic problem. In the specific problem, both the demands and the links are assumed bidirectional, while in the generic problem, all quantities are assumed unidirectional. Also, the link cost functions in the specific problem are functions of the **excess** capacity required on each link – the cost of the extra capacity that must be added to the link to accommodate the demand. The link cost functions in the generic problem reflect the costs of carrying the traffic on the link rather than the extra capacity required on the link to accommodate the demand. Both of these issues must be addressed in this mapping.

First the issue relating to the extra capacity is dealt with. This can be accommodated quite easily by performing a mapping on the dependent variable. The cost function – for bidirectional traffic – can then be written as:

$$f_{\mathrm{pr}}^{l}{}'(x)=\begin{cases} f_{\mathrm{pr}}^{l}(x-c_l) & x>c_l \\ 0 & x\le c_l \end{cases}$$

The modified link cost function is dependent on the capacity carried on the link rather than the excess capacity.

The function can then be extended to be bidirectional. As noted above, the demands are mapped from a set of undirected demands to a set of directed demands for the generic problem. Consequently, the demands can flow on either direction on each link. The link cost function, then, must take this into account in a way that is consistent with the specific problem. The approach used here is to make the cost of the directed link equal to the cost of the sum of the capacities flowing in either direction on the link, i.e.

$$f(s,t)=f(s+t)$$

The link cost function for the specific problem can be extended to a function suitable for the generic problem as follows:

$$f_{\mathrm{pr}}^{l}{}''(s,t)=f_{\mathrm{pr}}^{l}{}'(s+t)$$

This construction still permits arbitrary functions in the specific problem which can be mapped to arbitrary functions in the generic problem.

4.4.3 Formal Mapping from the Specific Problem to the Generic Problem

The mapping can then be formalised as follows:

- $N \rightarrow \mathrm{N}$;
- $L \rightarrow \Lambda$;
- $G(N, L) \rightarrow \Gamma(\mathrm{N}, \Lambda)$;
- $D' \rightarrow \Delta$;
- $f_{\mathrm{pr}}^{l''}(s,t) \rightarrow \phi_{\lambda}(s,t)$ and
- $\gamma_{v}(\cdot) = 0$ for all $v \in \mathrm{N}$

Thus, the specific problem is mapped to the generic problem and it is then possible to obtain a solution to the generic problem in order to solve the specific problem.

4.5 Examples and Solutions

The use of the network optimisation framework in solving the enterprise network design problem is demonstrated here. A number of different problems were generated and solved using this approach; also, different algorithms were used to solve the problems. The results are presented, analysed and discussed here. Two questions must be asked when assessing this approach to solve the problem: how long does it take to obtain some solution and what is the quality of the resulting solution. These two questions are addressed separately here.

The test problems were generated using a random problem generator. All of the problems were green field network design problems – no problems containing existing networks were generated. The problems can be divided into three broad categories of problem – these are characterised by the nature of the link cost function used for the private network. Problems with three different link cost characteristics were generated: a linear link cost function, a piecewise linear link cost function and a stepwise incremental link cost function. These different cost functions are illustrated in Figure 4-5. Solutions to these problems were obtained and analysed.

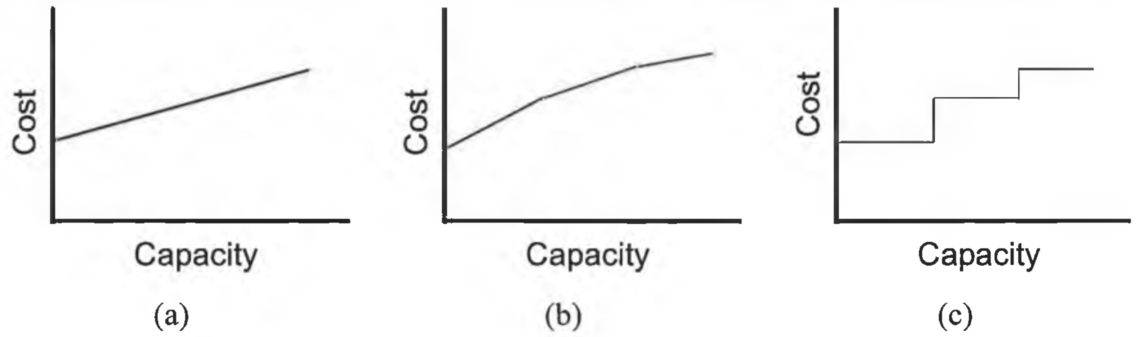


Figure 4-5: Link cost functions used in these examples – (a) linear link cost function, (b) piecewise linear link cost function and (c) stepwise incremental link cost function.

The random generator used to determine test functions is described first. Then the two questions posed above are addressed.

4.5.1 The Random Problem Generator

A random problem generator tool was developed to enable different random green field network design problems to be generated and used to validate this approach to obtaining some solution. The random problem generator that was designed was quite flexible and enabled a number of different variants of the problem to be generated at random. Problems with different numbers of nodes, different link cost functions and different switched cost functions can be generated. The generator has support for distance dependent costs and also supports a notion of a population associated with a location and demands can then be correlated with populations.

In the random problem generator, the nodes are chosen first. The number of nodes, n, and $J_{\max}$, the width and height of the square area on which the nodes are randomly placed are first specified. The node co-ordinates are then chosen from the uniform distribution $[0, J_{\max}]$. This enables the distance between the nodes to be calculated. Each of the nodes also has a relative population associated with it. This is chosen from the uniform distribution [0,1] and can be used to generate a set of demands in which the traffic generated by a particular node pair is related to the populations of both of the node pairs.

The demands are chosen next. Demands between all of the node pairs are generated and the demands between each node pair consist of a voice component specified in erlangs and a data component specified as a bitrate. The minimum and maximum for each of the voice and data demands are specified. Two possible ways of choosing the demands are then possible: they can be chosen entirely at random, or the demands can be correlated with the node populations. The first approach is straightforward and only requires a simple explanation – the voice and data demands are chosen from a uniform distribution with specified limits – but the second approach is somewhat more complex and a more detailed explanation is necessary.

In the second approach, the internodal demands are based on the populations of the two nodes. The geometric mean of the two node populations is determined. Since the node populations are between 0 and 1, the geometric mean of the populations is also between 0 and 1. The value for the demand that is then chosen is by linear interpolation between the minimum and maximum values for each of the demands using the geometric mean as the interpolation parameter. For example, if the population values are e_1 and e_2 and the minimum and maximum values of the demands are $m_{\min}$ and $m_{\max}$ respectively, then, using this approach, the value of the demand between the two populations is:

$$m = m_{\min} + \sqrt{e_1 e_2}\left(m_{\max} - m_{\min}\right).$$

Some random perturbation is then added to the resulting demand: some random fraction between +/-10% of the value of the demand is added. This is done to make the resulting problem slightly less regular.

The costs are specified next. The costs can be broken down into the link costs and the switched call costs. The link call costs are discussed first, followed by a short discussion on the switched call costs.

The problem generator permits three different types of link cost function to be generated: linear, piecewise linear and stepwise incremental. It also enables the link cost functions to be generated such that the link costs are distance dependent. This is optional and the case in which the link costs are not dependent on distance is discussed first.

If the link costs are not dependent on the distance, then the user can choose to have linear, piecewise linear or stepwise incremental link costs. All the link costs must be of the same type in this problem generator: it is not possible to have some linear link cost functions and some piecewise linear link cost functions. Note that this is a characteristic of the problem generator – the generic problem solvers have no problem accommodating very different link cost functions. The link cost functions that are generated permit a capacity to be mapped to a cost, and a link cost function must exist for all of the potential links in the problem. In the distance independent case, a set of parameters are chosen from which all of the link parameters are chosen at random. The linear parameters specified are the minimum and maximum link installation cost and the minimum and maximum marginal increase in capacity. This effectively defines an envelope on the link cost functions that are chosen as shown in Figure 4-6.

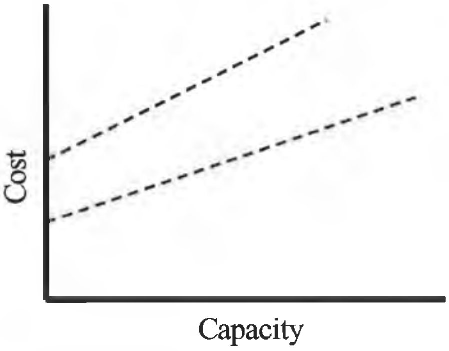


Figure 4-6: Envelope for linear link cost function. Any link cost function can be chosen which is bounded by the two lines chosen in the figure.

The piecewise linear and stepwise incremental cost functions are substantially different from the linear cost function, but since they require the same set of parameters to define the functions, they are described together here. Both of these functions are specified using a finite set of defining points – the points that define the function. These points are points at which the nature of the function changes somewhat. In the case of the piecewise linear function, these points are the points at which the slope changes and in the case of the stepwise incremental functions, these are the points at which the function value increases in a stepwise manner. The function value is obtained differently for each of the functions. For the piecewise linear function, the function value is obtained by interpolating between the defining points and extrapolating from the last one. For the stepwise incremental points, the function value is determined by finding the defining point with the maximum capacity less than the current capacity. The value of the function at this point is equal to the value at the current capacity.

The piecewise linear and stepwise functions are generated by choosing a fixed interval between the points that characterise the function. A minimum and maximum value for the initial value of the function is specified by the user and a value is chosen uniformly from this range. Each of the points are then generated. This is done in the same way for both the piecewise linear and stepwise functions, although the way that they are used when calculating the value of the function differs slightly in each case. Here, the discussion is in the context of the piecewise linear cost function, but exactly the same applies for choosing the points for the stepwise incremental cost function points. It is assumed that the slope of all the segments in the piecewise linear function decrease with increasing capacity. The slope for each segment can then obtained by multiplying the slope of the previous segment by some number between 0 and 1. In the generator, a range for the slope of the first segment is specified and then a range for the multipliers from which the slopes of the subsequent segments can be obtained is specified. Using this data, the set of points can be generated and both the piecewise linear and stepwise incremental cost functions defined.

The case in which link cost functions are distance dependent is a little more complex. The problem generator has support for link cost functions that vary linearly, in a piecewise linear fashion, or in a stepwise fashion with distance. The general approach used to incorporate distance into the link costs is to construct a kind of base cost-capacity function and a distance multipliers function. The overall function then is the product of the distance multiplier function evaluated at the appropriate distance and the base function.

A number of different variants for constructing distance dependent link cost functions are then possible. Any combination of linear, piecewise linear or stepwise incremental base functions and linear, piecewise linear and stepwise incremental distance multipliers is possible.

The switched costs then remain to be determined. Three possibilities exist for these: the switched costs can be fixed, random or distance dependent. In the first case, the switched costs are constant over all node pairs. In the second case, some range of values is specified and the cost per switched call between the node pairs is chosen at random from this range. Each node pair can have a different switched cost. The third case is one in which the switched call costs are dependent on the distance between the nodes. A single type of distance dependence is permitted here, which models existing distance dependent tariffs most accurately. This distance dependence is a stepwise incremental cost function. The parameters for this function are chosen in the same way that the parameters for the stepwise incremental cost function are chosen above.

This problem generator enables many types of problems to be generated, although not all types are used here.

4.5.2 Time Taken to Obtain Solutions to Problems

The problem generator is used to generate a substantial number of problems and the times taken to solve these problems are recorded and analysed here. The problems are solved using the generic problem solvers mentioned in Chapter 3. The objective is to compare the time taken to solve the different problems using the different problem solvers.

The problems have different link cost characteristics – some of the problems have linear link costs, some of them have piecewise linear link costs and some of them have stepwise incremental cost functions as described above. None of the problems solved have an existing network – all of the problems are green-field network design problems. The problems vary in size from 10 nodes to 50 nodes.

In the problems considered here, the amount of time taken to solve the entire problem comprises of the time taken to perform the mapping function plus the time taken to solve the generic problem. The time taken to perform the mapping function is small in comparison to the time taken to solve the generic problem. Hence the former is assumed to be negligible here for the purposes of comparing the time required to obtain solutions.

Results for the problems with a linear link cost function are discussed first, followed by discussion of the results for the piecewise linear and stepwise incremental cost functions.

The parameters chosen for the linear problems used in these experiments are shown in Table 4-1. These parameters were used to generate problems varying in size from 10 to 50 nodes in increments of 5 nodes. In these problems, the costs are distance independent and the demands generated are not correlated with the population associated with the node.

Five problems of each problem size were generated using the problem generator and the parameters listed above. Then they were mapped to the generic problem using software written to perform this function. As discussed above, the demands for the generic problem were generated by specifying the fraction of the voice traffic that gets shed onto the public network. In all of these cases, the fraction of traffic shed onto the public network was 5%. This was used to determine the number of trunks required to carry voice traffic between the node pairs. The link cost functions used in the generic problem were exactly the link cost functions defined in the specific problem.

Parameter	Value
Maximum internodal distance	500
Maximum Voice Demand (Erlangs)	10
Minimum Voice Demand (Erlangs)	3
Maximum Data demand (Mb/s)	2
Minimum Data demand (Mb/s)	0.5
Data demands correlated with node size	false
Demands chosen from uniform distribution	true
Demands chosen from normal distribution	false
Link cost data	linear
Distance dependent	false
Minimum link installation cost	5000
Maximum link installation cost	10000
Minimum marginal cost $(\text{Mb/s})^{-1}$	50
Maximum marginal cost $(\text{Mb/s})^{-1}$	100
Distance independent switched costs; fixed call costs	0.1

Table 4-1: Parameters used in the random network generator tool to generate problems with linear cost functions.

The two approaches described in chapter 3 were used to solve the problems described here. For the greedy approach to solve the problem, a fully connected initial starting point was chosen. For the simulated annealing algorithms, results were obtained for different parameter sets: specifically, results were obtained for different initial cooling temperature and cooling rates. The results obtained by varying these parameters are shown in the figures below.

The generic problem solver was run on a 270Mhz Sun UltraSparc 5 machine running the Solaris 2.6 operating system. The machine in question was running in multi-user, multi-processing mode. Consequently, any number of processes could have been running concurrently with the generic problem solvers. Solaris provides a rudimentary mechanism to obtain information on how much time was used by a process. The operating system can determine how many seconds of processor time a particular process consumes. This is mostly independent of the load on the system and the number of users of the system. This time is used for comparison purposes here.

A very comprehensive set of results was obtained for the problems with linear link cost functions. All of the problems were solved using the greedy and simulated annealing approaches. The results are shown in Figure 4-7 to Figure 4-10. Each point on the graph is an average of the solutions to the five problems with the same characteristics.

The results show that the amount of time required to solve the problem increases exponentially with problem size for the problem with linear link cost functions. This is the case for both the greedy and simulated annealing algorithms. Note that the longest solution times can be about 40 minutes (~2700 seconds). This is prohibitively long for any kind of interactive application, but is adequate for applications in which it is more important to obtain a good quality solution at the expense of processing time.

The time required to solve the problem increases exponentially with the problem size because the size of the neighbourhood increases exponentially with the number of nodes. Both the algorithms are sensitive to the size of the neighbourhood. This is obvious in the case of the greedy algorithm, since the algorithm involves costing all of the neighbours of a particular state. If the number of neighbours of each state is increasing dramatically, then the time taken to search all of the neighbours will also increase dramatically. It is less obvious in the case of the simulated annealing algorithm.

In the simulated annealing algorithm, the number of iterations is governed by the cooling schedule. The simulated annealing algorithm used in this work is straightforward: the current temperature is decreased by some fixed, specified proportion at each iteration. Thus, the temperature at iteration i, T_i, can be written as

$$T_i = T_0 \kappa^i$$

where T_0 is the initial temperature and κ is the cooling parameter. Here, the stopping condition is reached if the current temperature is less than 1.0. If the initial temperature and the cooling parameter are known then the number of iterations is fixed. Importantly, the number of iterations is independent of the problem size. Consequently, the increase in time required to obtain a solution results from an increase in the amount of time required to perform each iteration.

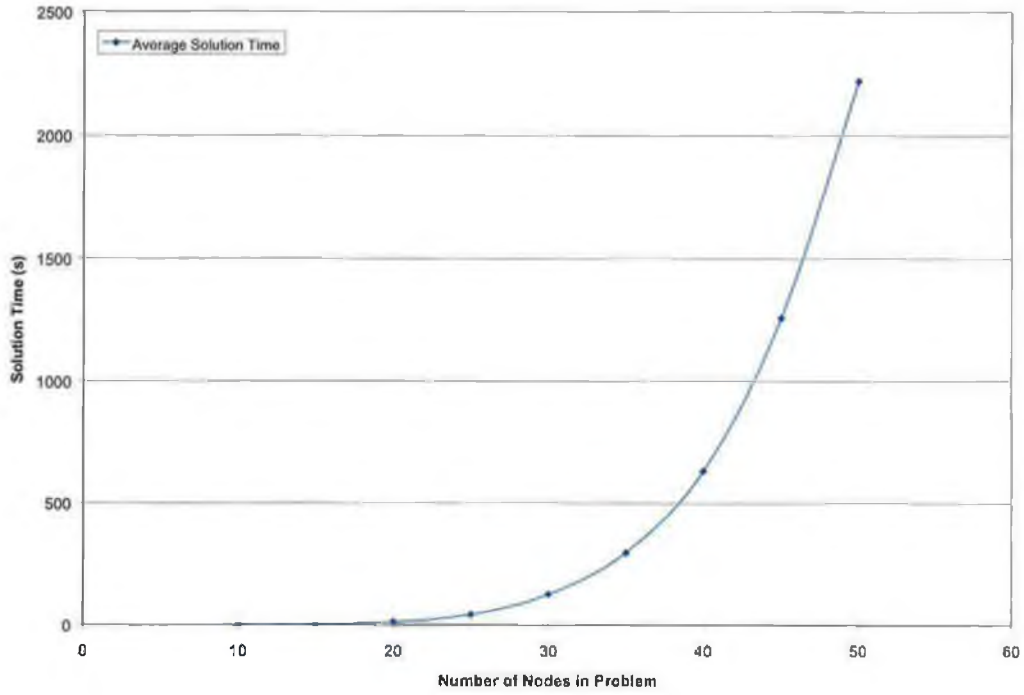


Figure 4-7: Time required to obtain solution to problem with linear cost function using the greedy algorithm.

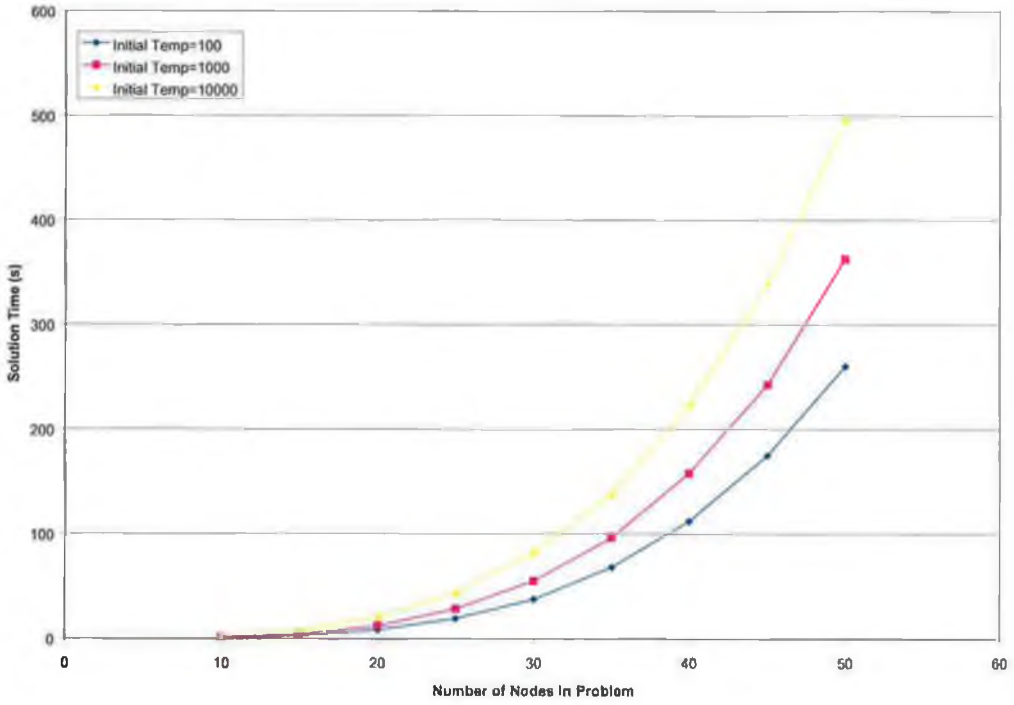


Figure 4-8: Time required to obtain solution to problem with linear cost function when simulated annealing cooling parameter is 0.98.

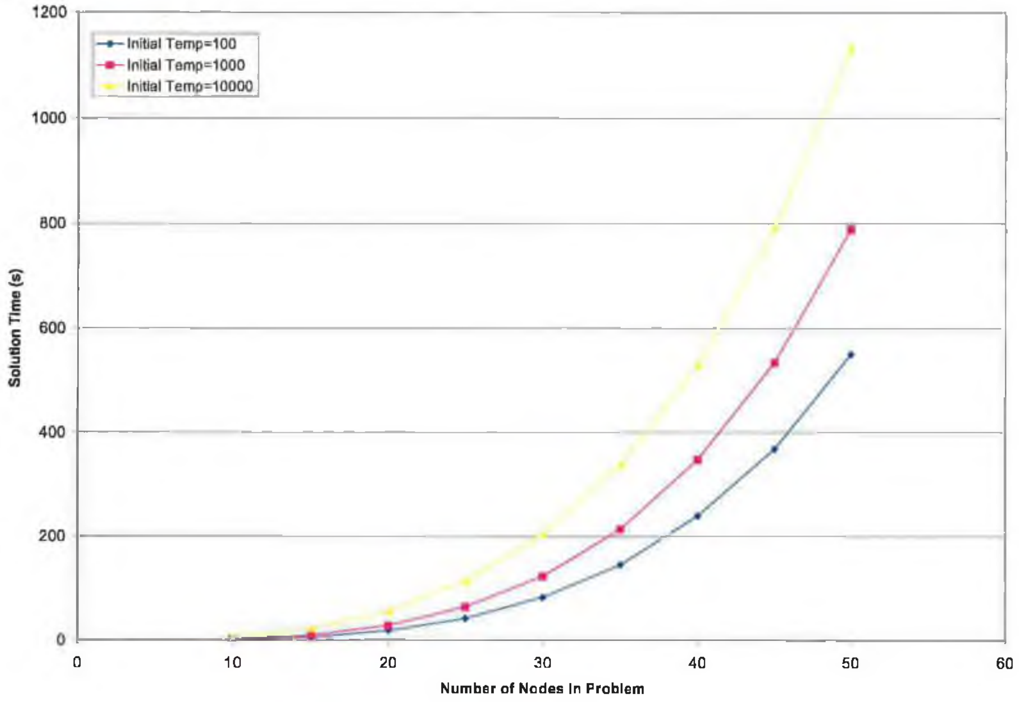


Figure 4-9: Time required to obtain solution to problem with linear cost function when simulated annealing cooling parameter is 0.99.

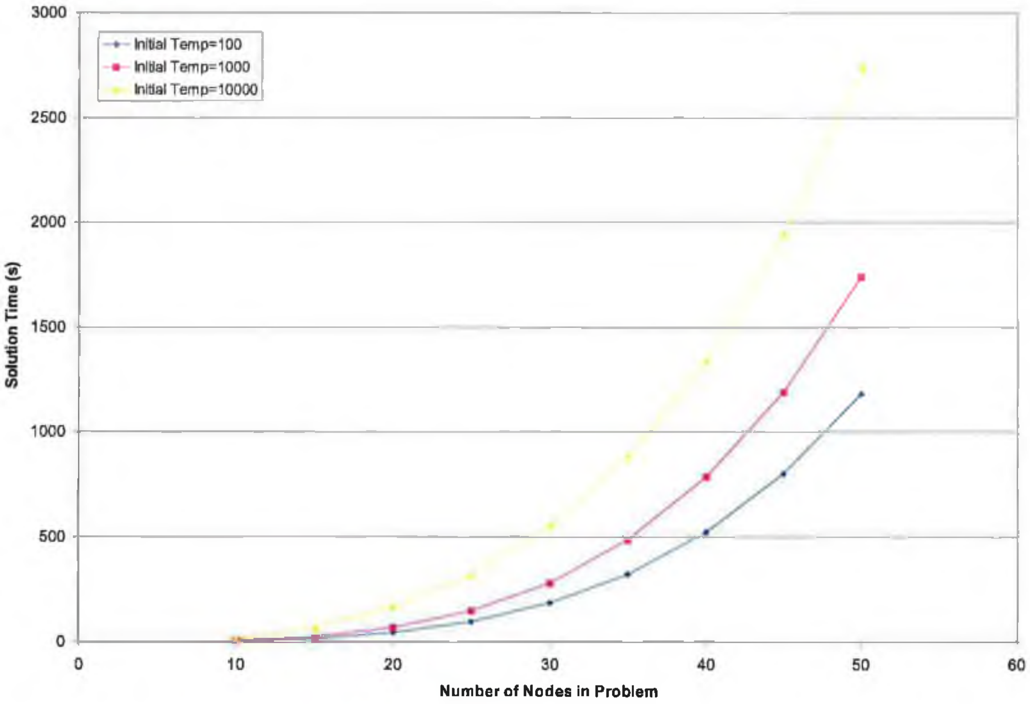


Figure 4-10: Time required to obtain solution to problem with linear cost function when simulated annealing cooling parameter is 0.995.

The increase in time required to perform each iteration can be explained by noting that the simulated annealing algorithm operates by choosing a neighbour at random. In this implementation, this works by counting the number of neighbours of the current state and choosing one of these. Since the size of the neighbourhoods increases exponentially with the number of nodes, the time required to perform the counting operation increases exponentially with the problem size.

Some approach which is independent of the size of the neighbourhood is desirable. This may operate more quickly and would certainly be more scalable.

Similar experiments were run for the scenarios in which the cost functions were piecewise linear and stepwise incremental to see if these differ significantly (see Figure 4-11 to Figure 4-18). The results obtained for these scenarios were not as comprehensive as those obtained for the linear case. The time taken to obtain a solution in this case is a little longer than the linear cost function case. This is because it takes longer to calculate the link cost function. Since this is performed very many times during the optimisation, a small difference in time taken to perform this calculation can result in a significant difference in the time taken to perform the overall computation.

Here, it is clear that scenarios containing more than 50 nodes require substantial amounts of time – almost 2 hours in some cases – hence, unless it is possible to run the software, say, overnight, or possibly over a few days, the algorithms used here will not be able to find solutions for problems of more than 10's of nodes.

Note also that the computing power used when obtaining these solutions is not very modern. Current high-end processors can operate at 1.5GHz[11]. Some experiments were performed to determine the difference in speed between a Pentium® III based system operating at 600MHz and the Sparc-based system that was used to perform these experiments. The experiments showed that the Pentium® system performed approximately 5 times faster than the Sparc system. The latest microprocessors could probably operate 10 times faster. Having such processing power available would increase the maximum size of problem that is solvable within some specific time, but since the time required increases exponentially, the increase in processing power would only permit a relatively small increase in the maximum possible problem size.

[11] Vendors sell Pentium® 4 based systems in which the processors operate at this speed.

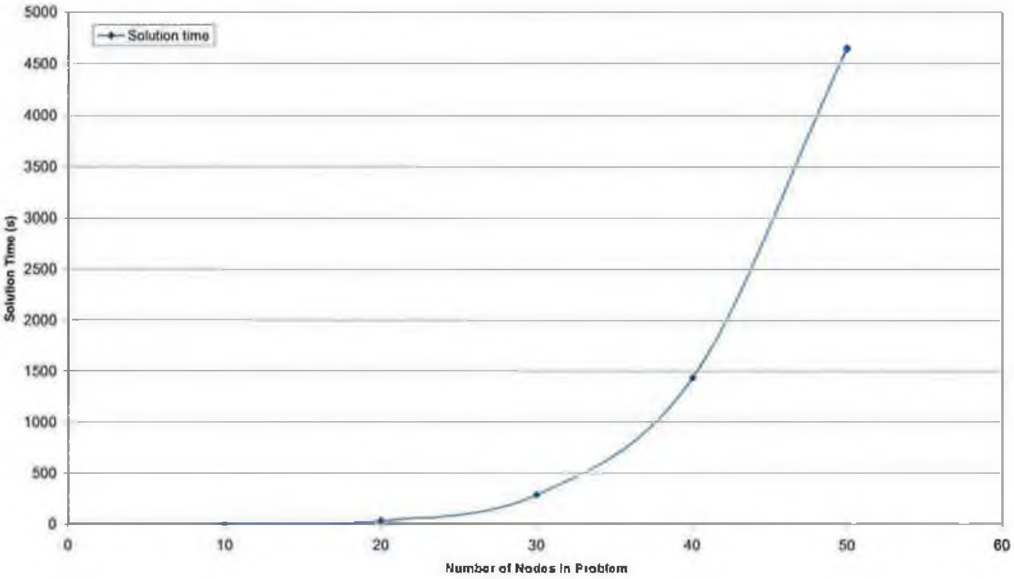


Figure 4-11: Time required to obtain solution to problem with piecewise linear cost function using greedy algorithm.

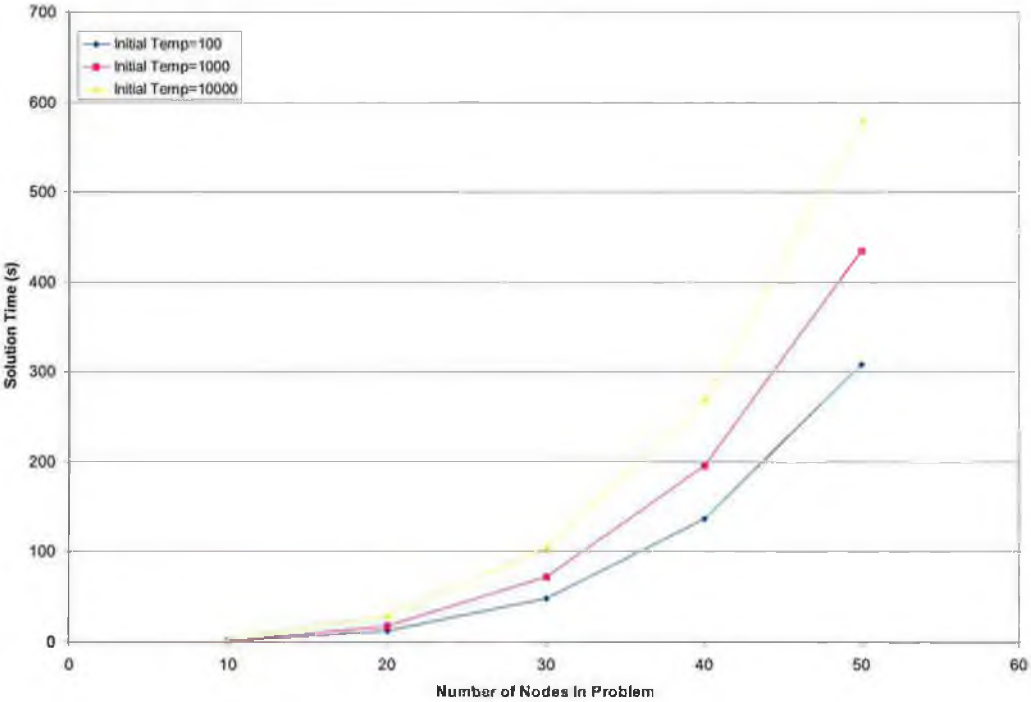


Figure 4-12: Time required to obtain solution to problem with piecewise linear cost function when simulated annealing cooling parameter is 0.98.

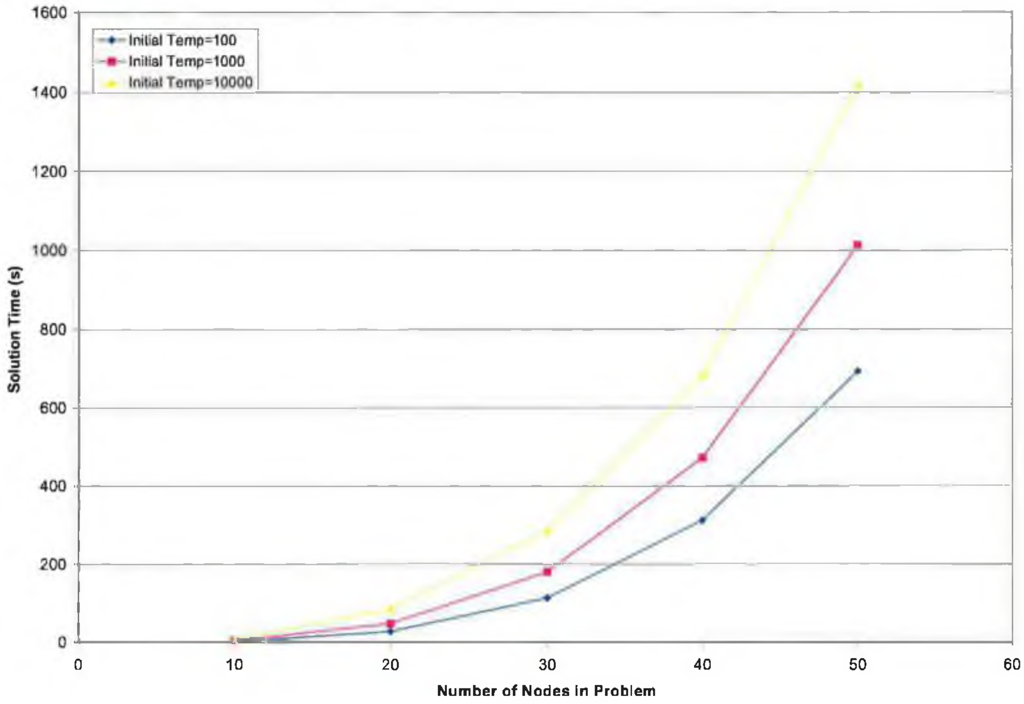


Figure 4-13: Time required to obtain solution to problem with piecewise linear cost function when simulated annealing cooling parameter is 0.99.

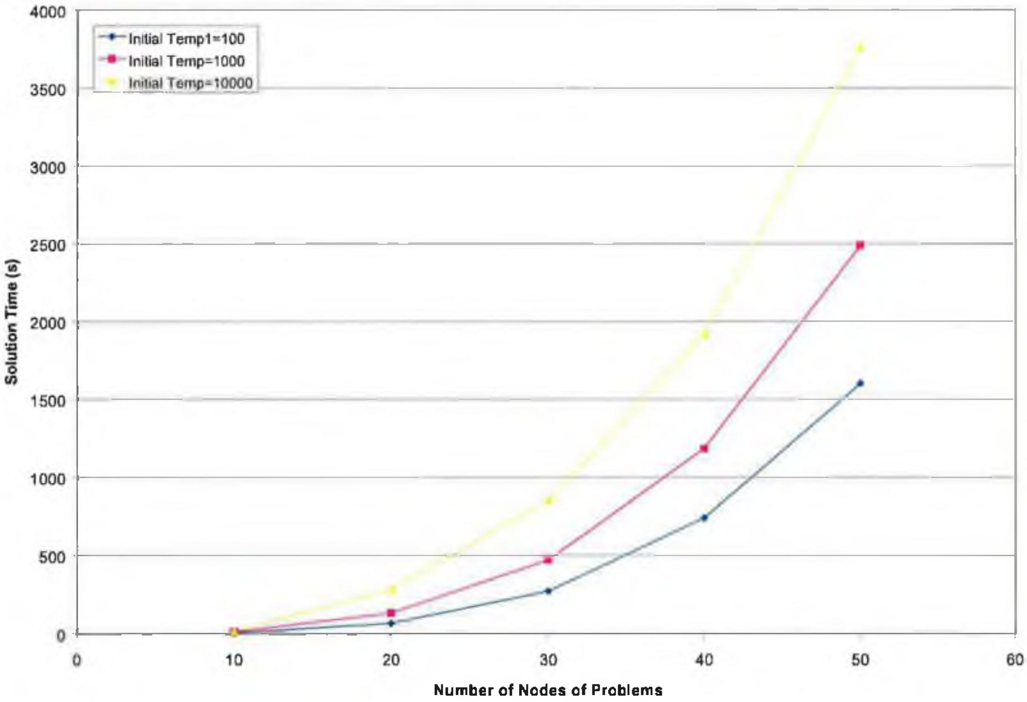


Figure 4-14: Time required to obtain solution to problem with piecewise linear cost function when simulated annealing cooling parameter is 0.995.

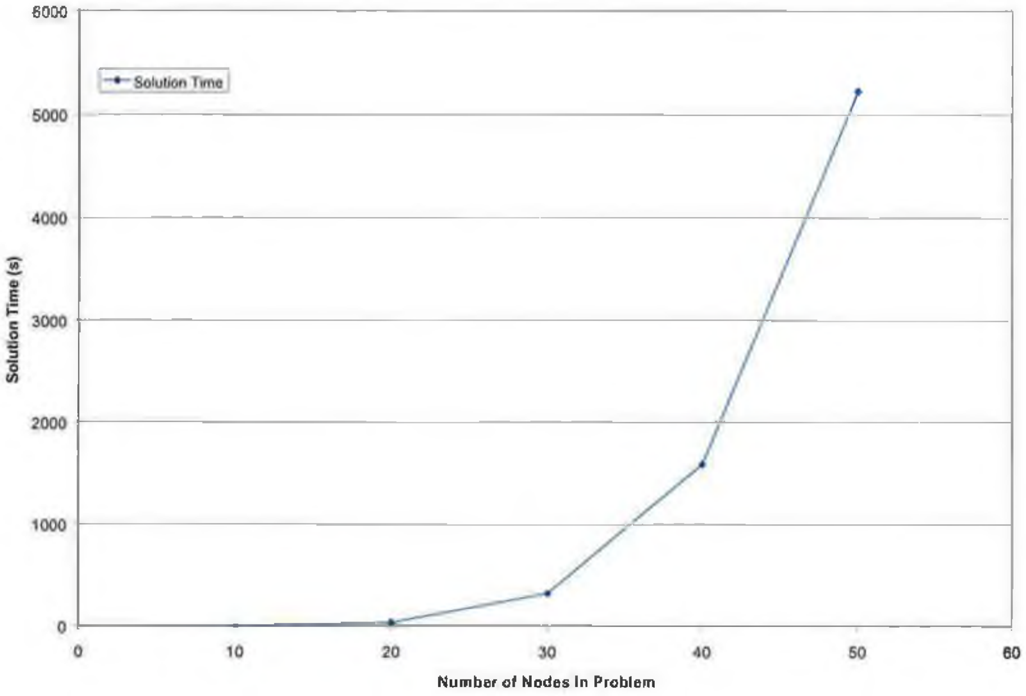


Figure 4-15: Time required to obtain solution to problem with stepwise incremental cost function using greedy algorithm.

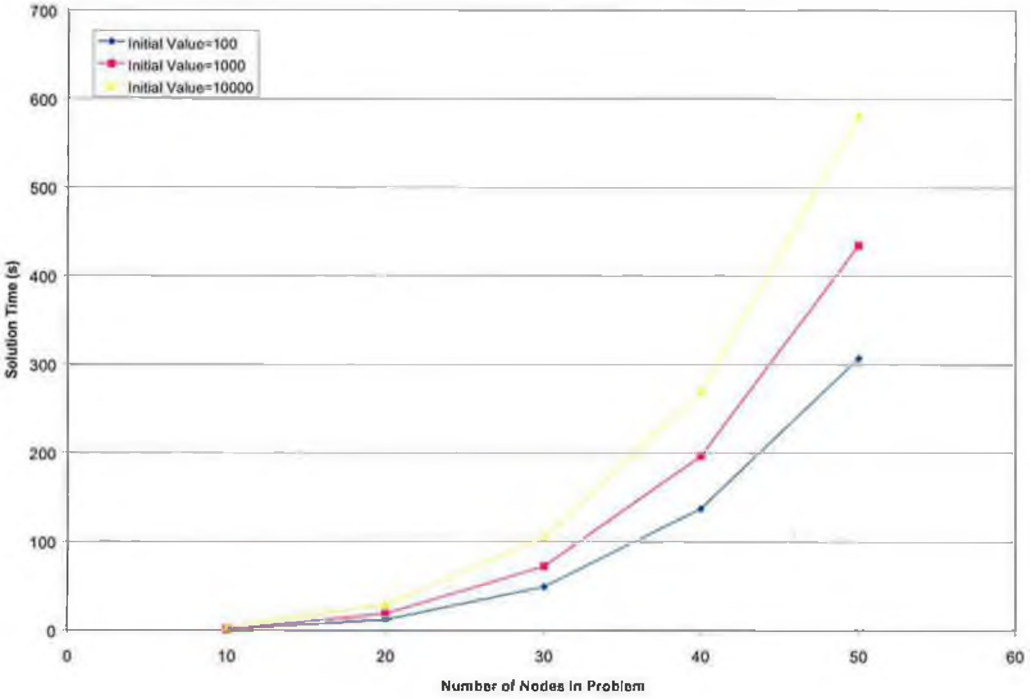


Figure 4-16: Times required to obtain solution to problem with stepwise incremental cost function when simulated annealing cooling parameter is 0.98.

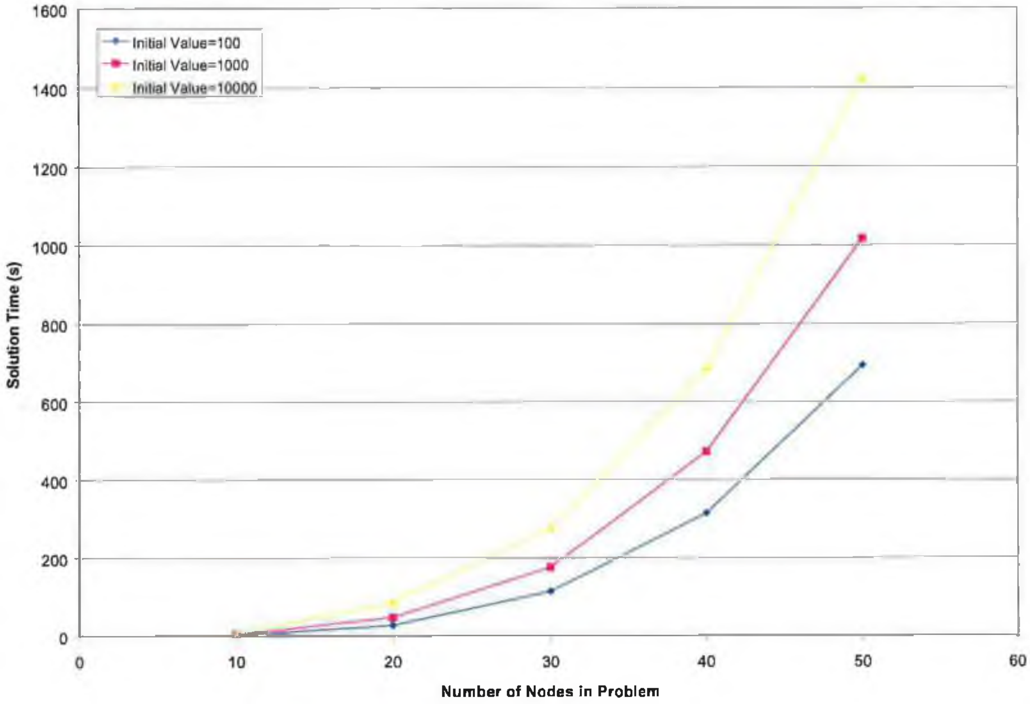


Figure 4-17: Times required to obtain solution to problem with stepwise incremental cost function when simulated annealing cooling parameter is 0.99.

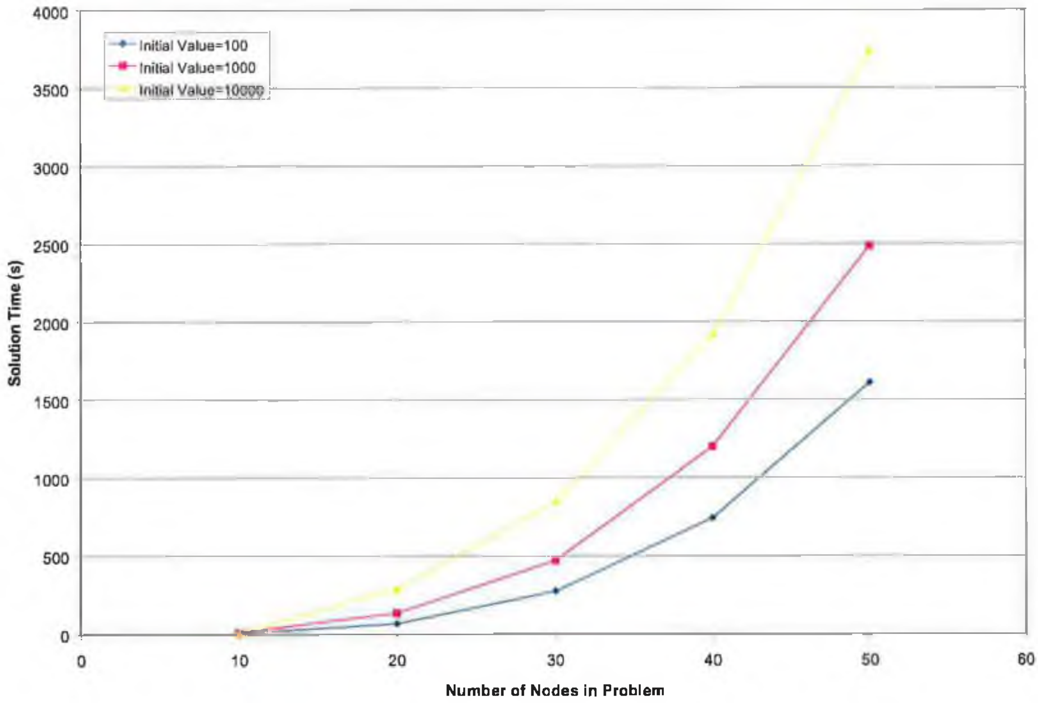


Figure 4-18: Time required to obtain solution to problem with stepwise incremental cost function when simulated annealing cooling parameter is 0.995.

Some further observations are useful. In the above experiments, the greedy algorithm takes the most time. The solution time for the simulated annealing algorithm when the cooling parameter is 0.98 are approximately 10% of that of the greedy algorithm. The simulated annealing run with cooling parameter of 0.995 and initial temperature of 10000 results in a running time comparable to that of the greedy algorithm. The other parameter choices result in running times somewhere in between these two extremes. Obviously, the initial temperature could be made larger or the cooling parameter could be made closer to 1 resulting in even longer running times: the simulated annealing running times can of course exceed those of the greedy algorithm if the parameters are chosen appropriately.

4.5.3 Quality of Solutions Obtained Using this Approach

Considering the running time alone is not sufficient: the quality of the solution in terms of the resulting cost must also be considered. If the approach takes a long time to find a poor quality solution, then the approach is not very useful. Here, the quality of the resulting solutions is assessed.

Choosing Simulated Annealing Parameters

A number of parameters need to be specified when running the simulated annealing algorithm. In the instance of the algorithm used here, three parameters are required: the initial temperature used in the cooling schedule; the final temperature used in the cooling schedule and the cooling parameter which controls the rate at which the process is cooled. Each of these parameters can affect both the time required to obtain a solution and the quality of the resulting solution.

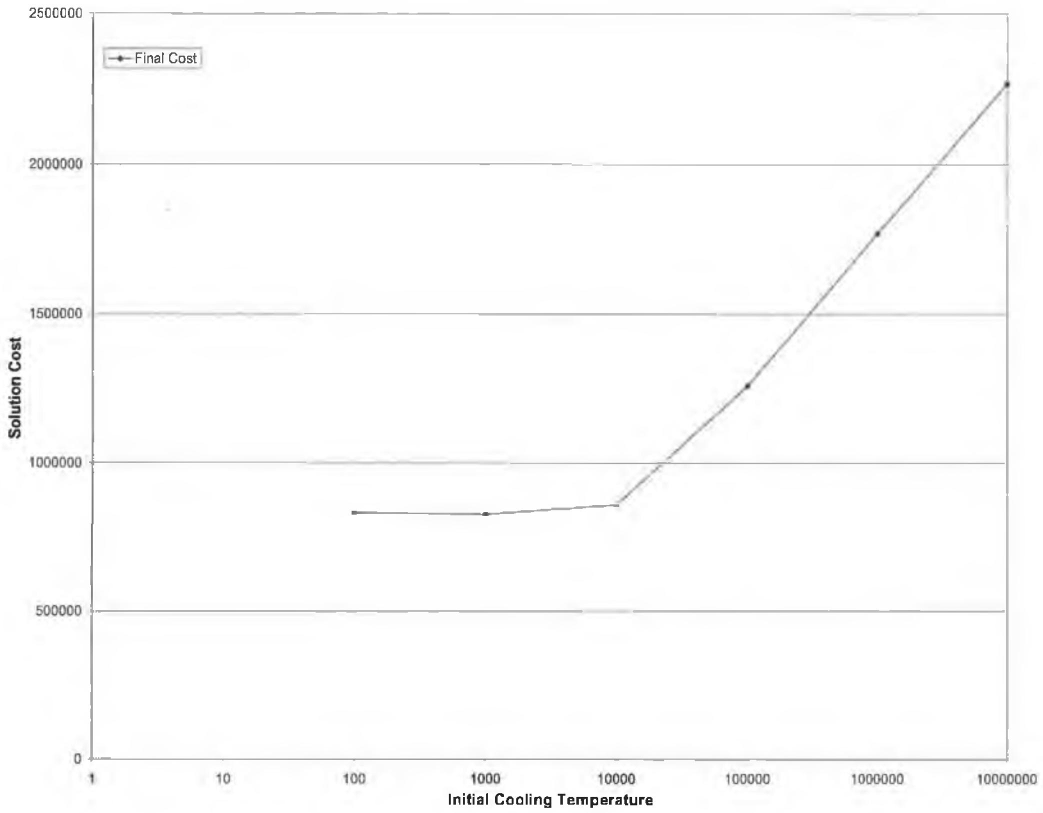


Figure 4-19: Variation of result with initial temperature of simulated annealing algorithm for a randomly generated 10-node problem.

The initial temperature is considered first. The initial cooling temperature was varied substantially for one problem – this problem was a 10-node problem. The cooling parameter and the final temperature remained fixed. The simulated annealing algorithm was run once each time using a single random seed. The SA algorithm could have been run a number of times and the results averaged to obtain more representative results: however, the trend is very obvious, even when only one run of the SA algorithm is used. It is clear that as the initial temperature increases beyond some point, the quality of the solution obtained gets worse as can be seen in Figure 4-19 for this small problem.

The final solution quality gets substantially worse as the initial cooling temperature increases. At the highest level, this can be explained by noting that the algorithm permits many moves that result in poorer quality solutions in the initial period of the cooling schedule. The algorithm then experiences difficulty in reversing all of these poor moves when the cooling schedule is nearing its end. Consequently, the final solution retains some of the poor characteristics that were introduced in the initial stages of the cooling schedule.

The situation in this case is made more complex due to the fact that many of the neighbours of a particular state are poor neighbours: many of the neighbours can result in very unsuitable routings for demands. For example, some of the neighbours can involve rerouting demands via expensive links. Moreover, if such moves are accepted in the initial part of the cooling schedule, a link may be removed as discussed in chapter 3. Removing a link in this case makes it impossible to revert to the previous situation. It is conceivable that the removed link could be critical to a low-cost solution: in this case, the simulated annealing algorithm would not be able to arrive at such a low cost solution.

Note also that for a fixed cooling rate, the time required to obtain the solution is related to the initial temperature: if the initial temperature is increased, the time required to obtain a solution is also increased. Since a large initial temperature results in both a poorer quality solution and an increased amount of time to obtain the solution, it makes sense not to consider using very large initial cooling temperatures for these small problems.

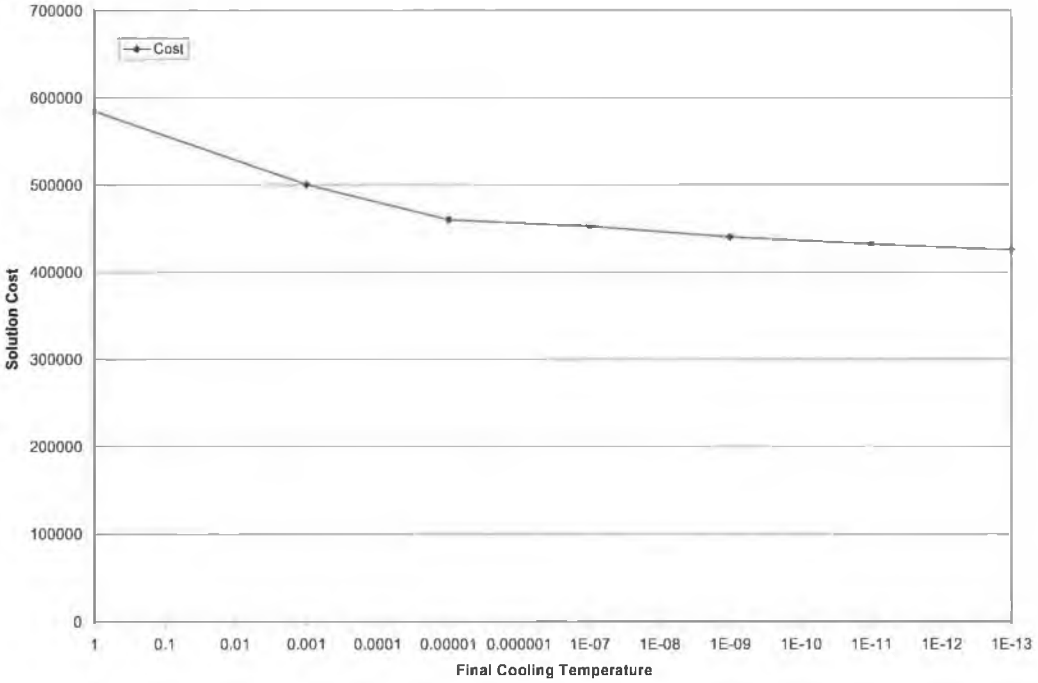


Figure 4-20: Variation of solution quality with final temperature of cooling schedule.

The final temperature of the cooling schedule – when the cooling schedule reaches this temperature, the process terminates – is also important. It has an effect on both the time taken to obtain a solution and the resultant solution quality. This can be seen in Figure 4-20. The solution quality improves as the final temperature decreases. However, there is a time cost associated with this, which is shown in Figure 4-21. The amount of iterations increases linearly as the log of the final temperature decreases. This arises directly from the fact that the cooling temperature at each interval is equal to that of the last interval multiplied by the cooling parameter. The question then is whether the increase in the time required to obtain a solution is warranted and this is dependent on how much time is available and what the application is. Note however, that the linear improvements resulting from decreasing the final temperature shown in Figure 4-19 will break down and decreasing the cooling temperature further will not result in any more cost savings. This will occur as the solution becomes closer and closer to a local minimum.

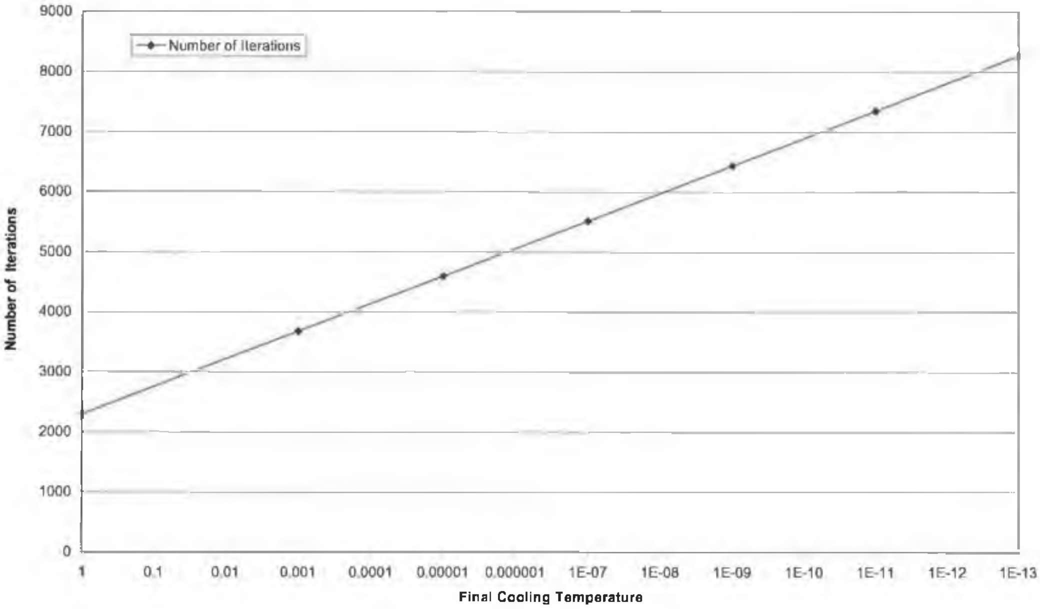


Figure 4-21: Variation of number of iterations with final temperature. The initial cooling temperature is 100000 and the cooling parameter is 0.995.

The cooling parameter also has a substantial effect on both the time taken to obtain a result and the quality of the result obtained. The dependence of the solution quality on the cooling parameter is more difficult to identify than either the initial or final cooling temperature, but some investigation of the effect of the cooling parameter on the solution quality follows.

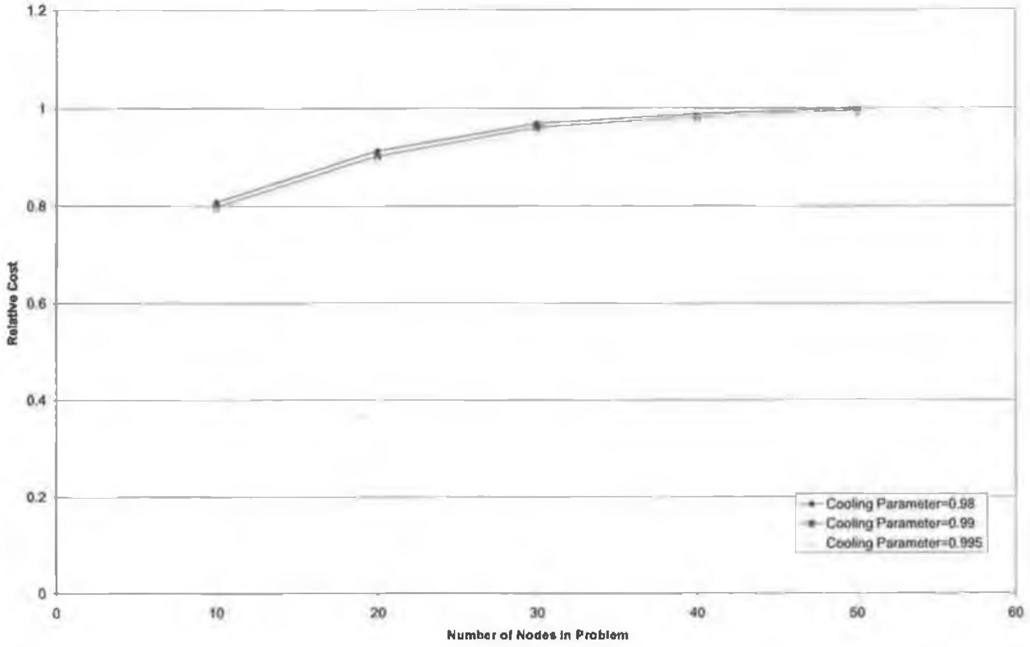


Figure 4-22: Variation of costs for different values of the cooling parameter for the problems with linear costs. The initial temperature is 1000. The results are relative to some norm.

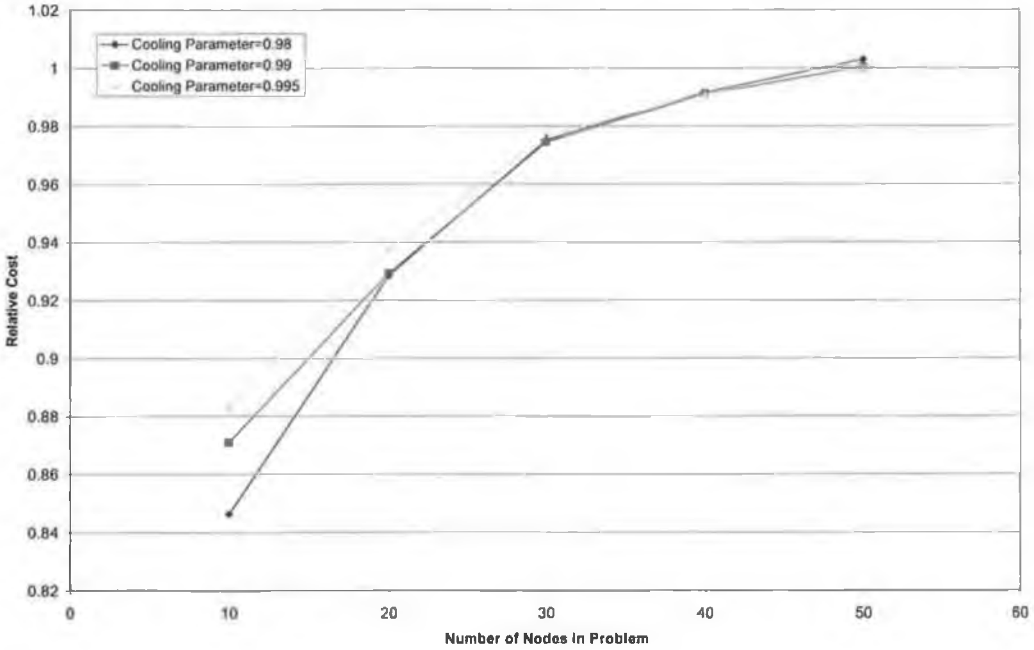


Figure 4-23: Variation of costs for different values of the cooling parameter for the problems with linear costs. The initial temperature is 10000. The results are relative to some norm.

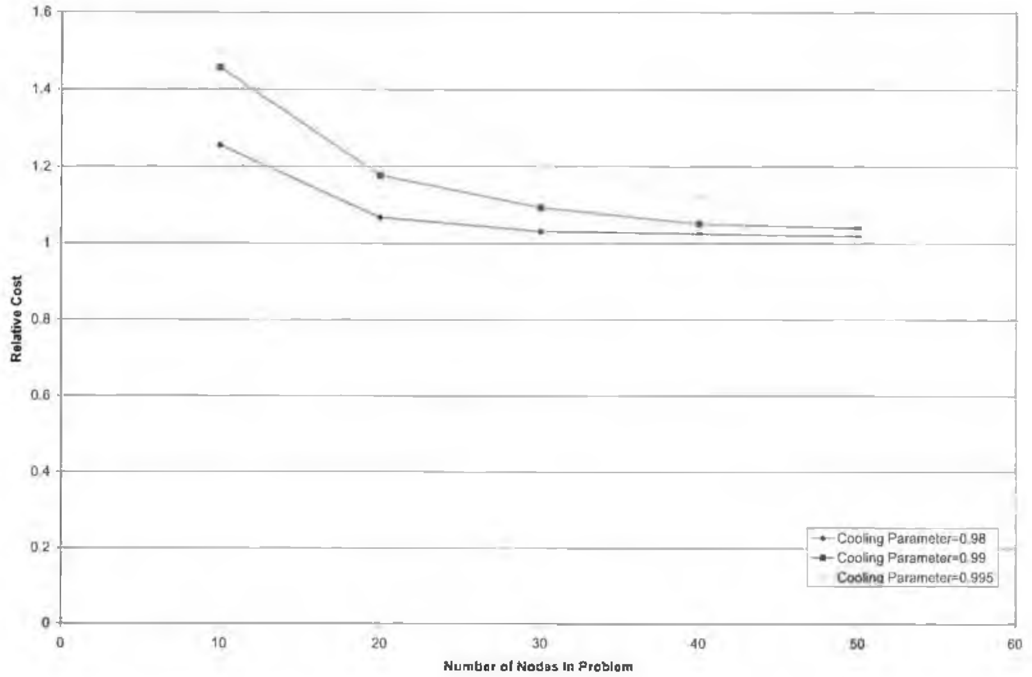


Figure 4-24: Variation of costs for different values of the cooling parameter for the problems with linear costs. The initial temperature is 100000. The results are relative to some norm.

Examples of how the solution quality varies with the cooling parameter are shown in Figure 4-22 to Figure 4-24. The results shown in these figures were obtained by using the simulated annealing algorithm to solve a particular problem. The results were averaged and the relative costs using the different approaches are shown in Figure 4-22 to Figure 4-24. From the figures, it can be seen that for a higher initial temperature, a higher cooling parameter causes the quality of the solution to be worse; conversely, for a lower initial temperature, a higher cooling parameter causes the solution quality to be better. In both cases, this can be explained by the fact that the cooling parameter lengthens the overall cooling process, but more discussion of each case is warranted.

In the case in which the initial temperature is high, lengthening the cooling process means that more time is spent in this high temperature part of the cooling schedule. Consequently, more poor quality moves are permitted and the solution can stray far from an optimal in this part of the cooling schedule. Furthermore, the particular implementation of the simulated annealing algorithm does not permit links to be inserted once they are removed: this makes it impossible to revert to a previous state if the transition to the current state results in this link removal. This can mean that poor quality moves are accepted in the early part of the cooling schedule, and they cannot be undone later on. This gives rise to poorer quality solutions in the case in which the initial temperature is high and the cooling parameter is high.

When the initial temperature is low, moves that are accepted are mostly those which cause a reduction in the overall cost of the solution; the probability of accepting a move resulting in an increase in cost is small. Hence, using a larger cooling parameter in this case, causes the cooling process to be lengthened and increases the number of moves attempted. Since the accepted moves are predominantly those which cause a reduction in cost, using a larger cooling parameter in this case has the effect of causing more moves resulting in a lower cost to be tried and hence the resulting solution to be of lower cost.

There is a region in between these two extremes for which the use of the larger cooling parameter can result in increased costs for smaller problems and decreased costs for larger problems. This arises because the bad irreversible moves alluded to above have more of an impact in the smaller problems than the larger problems.

Similar results were observed for problems with piecewise linear and stepwise cost functions.

The above results show that it is difficult to conclude what is the best cooling parameter in each case. It is dependent on the problem size and the initial cooling temperature. Perhaps if the irreversible nature of the cooling process was modified, the process would be more predictable and the use of a larger cooling parameter would uniformly result in improved solution quality on average. However, this was not explored here because the solutions obtained using the greedy algorithm were so much better as is discussed below.

Comparison of Results Obtained Using Greedy and Simulated Annealing Algorithms

The simulated annealing algorithm was run for a number of different problems and the results were compared with the results obtained using the greedy algorithm. A set of simulated annealing parameters was chosen and was applied to many of the different problems. The sets of simulated annealing parameters chosen are shown in Table 4-2.

Problem Identifier	Initial Temperature	Cooling Parameter
SA1	1000	0.98
SA2	1000	0.99
SA3	1000	0.995
SA4	10000	0.98
SA5	10000	0.99
SA6	10000	0.995
SA7	100000	0.98
SA8	100000	0.99
SA9	100000	0.995
SA10	1000000	0.98
SA11	1000000	0.99
SA12	1000000	0.995

Table 4-2: Parameters used in each of the simulated annealing solvers.

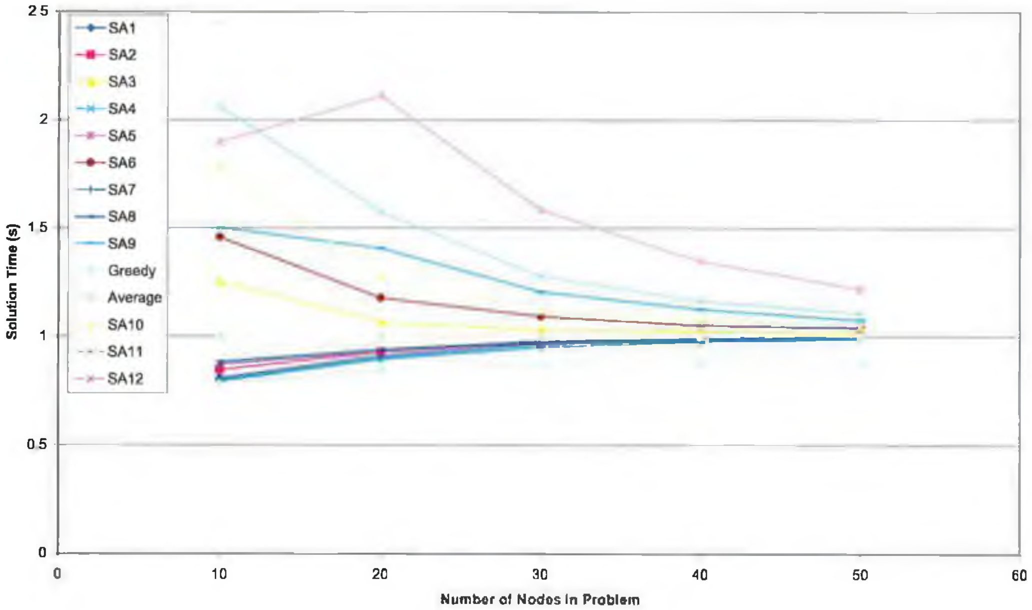


Figure 4-25: Comparison of results obtained using different algorithms and parameters for problems with linear cost functions.

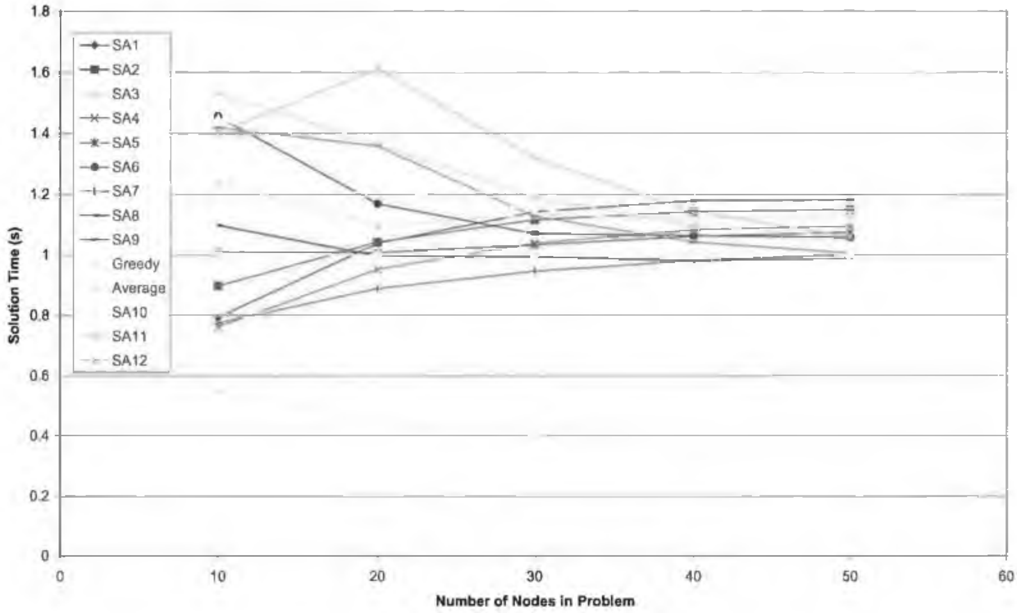


Figure 4-26: Comparison of results obtained using different algorithms and parameters for problems with piecewise linear cost functions.

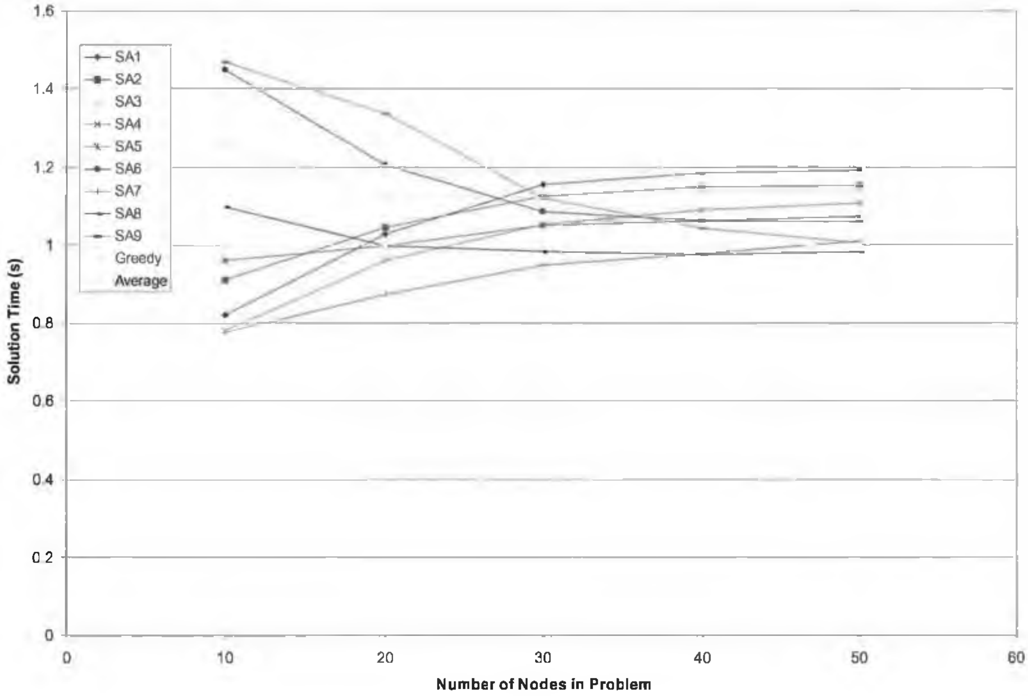


Figure 4-27: Comparison of results obtained using different algorithms and parameters for problems with stepwise cost functions.

The results are shown in Figure 4-25, Figure 4-26 and Figure 4-27 for the problems with linear, piecewise linear and stepwise incremental cost functions respectively. In all cases, it is clear that the greedy algorithm performs significantly better. In the linear cost function case, the difference is least but it is still quite substantial. The difference is considerably greater in both the piecewise linear and stepwise incremental cases.

The difference between the results obtained using the greedy algorithm and the simulated annealing algorithm can be explained using some of the arguments in section 4.5.2 above. Here, three different sets of solutions obtained using the simulated annealing algorithm are identified and considered separately.

In the case in which the initial temperature is low, the algorithm performs in a manner somewhat similar to the greedy algorithm: only moves that result in a lower cost state are accepted. However, in the simulated annealing algorithm, instead of performing an exhaustive search of the state space, neighbours are chosen repeatedly at random, until lower cost neighbours are obtained. This approach is less likely to lead to the local minimum than the more systematic greedy approach. For this reason, the results obtained using the greedy approach are better in this case.

In the first case in which the initial temperature is quite high, bad moves will be permitted in the earlier part of the cooling schedule. This can result in the algorithm entering a bad state. This, coupled with the fact that reinstallation of links is not possible results in a poor solution. This can be seen in Figure 4-25 to Figure 4-27 (this is shown in plots SA7, SA8 and SA9).

If the initial temperature is made very high and the cooling parameter is also high, the solution quality can improve as the cost is made number of nodes increases as is shown in Figure 4-25 and Figure 4-26 (plot SA12 in particular). This does not result in a very good solution, but the point is to note that arbitrarily increasing the initial temperature does not always result in poorer and poorer solutions. This relative improvement in the solution probably comes about in larger problems because the amount of links that are removed is reduced. Also, the number of routes that will remain unaffected by the optimisation algorithm will be larger.

Overall, it can be seen that the simulated annealing algorithm generally results in poor quality solutions.

The difference in the results obtained in the piecewise linear and stepwise incremental cases can be explained by the economies of scale that arise in these cases. In these cases, there are significant advantages to routing as much of the traffic as possible on a small set of links. For example, rerouting a demand on a network with stepwise incremental cost functions may result in no increase in capacity (and hence cost) on the links comprising the new route. This is in contrast to the linear case, where such a rerouting would result in a cost being incurred on the alternate route.

The greedy algorithm attempts to do concentrate as much of the capacity as possible on a small number of links, availing of the aforementioned economies of scale. With the greedy algorithm, hub locations naturally arise and the other nodes in the network are 'homed' on these hub locations. When solving the problem using the simulated annealing approach, the nature of the algorithm means that it is less likely that these hubs will develop – the resulting topology will be much more random, and the overall cost will be higher. Hence the cost difference between the results obtained using the simulated annealing approach and the greedy approach in the piecewise linear and stepwise incremental link cost function cases.

4.5.4 Trade-off Between Solution Quality and Time Required to Obtain Solution

When solving the problems above, it was clear that the greedy algorithm obtained the best solution all the time. However, there may be some cases in which it is desirable to obtain some solution in a reasonably short space of time, rather than obtaining the best solution in a longer time. Hence, it makes sense to consider the trade-off between the time required to obtain the solution and the quality of the resulting solution.

In section 4.5.2 it was observed that time required to obtain solutions increases exponentially with the problem size: this is true of both the greedy approach and the simulated annealing approach. However, for a problem of a given size, it can take considerably longer to obtain a result using the greedy algorithm than it can to obtain a result using a simulated annealing algorithm.

Choosing a large initial cooling temperature and/or a large cooling parameter in the simulated annealing algorithm resulted in a considerable time taken to obtain a solution. Also, the resulting solution was worse than that of the greedy algorithm. Consequently, in this case, it is beneficial to choose parameters for the simulated annealing algorithm that result in a short execution time. This means that the simulated annealing algorithm is a more likely candidate for situations in which the time required to obtain a solution is more limited.

Number of Nodes	Relative Greedy Cost	Greedy Solution Time	Relative SA Cost	SA Solution Time
10	0.78346	0	0.807971	0
20	0.843912	12.8	0.912859	8.08
30	0.863901	126	0.969034	37.52
40	0.878137	631.8	0.987921	111.8
50	0.876892	2221	1.000717	260.08

Table 4-3: Time vs. Cost comparison of solutions for the problems with linear cost functions. The SA algorithm results are for the SA1 parameters above.

The solutions obtained using the simulated annealing algorithm using a small cooling parameter resulted in very small percentage differences in solution quality (<1%). Consequently, the benefits of running the algorithm multiple times with different random seeds to obtain different solutions are questionable – the algorithm can be run once and the solution obtained. Here, the use of the simulated annealing algorithm run once for a specified set of parameters is compared with the use of the greedy algorithm.

Number of Nodes	Relative Greedy Cost	Greedy Solution Time	Relative SA Cost	SA Solution Time
10	0.5487868	0	0.7929614	1
20	0.4471776	30.4	1.0380581	11.6
30	0.3997993	287.6	1.1421577	48.2
40	0.3451297	1436.4	1.1788172	136.2
50	0.3147747	4653	1.1816319	307.8

Table 4-4: Cost vs. quality comparison for problems with the piecewise linear cost function. The SA algorithm results are for the SA1 parameters above.

In the case in which the linear cost function is used, the discrepancy between the results obtained using the simulated annealing algorithm and those obtained using the greedy algorithm are not very great for the sizes of the problems studied. The results are shown in Table 4-3. Since the simulated annealing algorithm obtains the solution in relatively short time in this scenario – it takes less than 1/8th the time when the problem size is 50 nodes – there are some advantages to the use of this algorithm in this case. However, when the cost functions are linear functions of the capacity, more efficient techniques exist to solve the problem such as that proposed by McGibney. Hence, the usefulness of this approach in the linear case is questionable. It is included here for comparison with the other cases.

Number of Nodes	Relative Greedy Cost	Greedy Solution Time	Relative SA Cost	SA Solution Time
10	0.47863	0	0.821101	1.0
20	0.442224	27.4	1.026898	12.2
30	0.368313	250.8	1.153846	49.8
40	0.31966	1195.4	1.184053	137.2
50	0.283738	3949.2	1.191688	302.8

Table 4-5: Cost vs. quality comparison for problems with the stepwise incremental cost function. The SA algorithm results are for the SA1 parameters above.

The results obtained in the case in which the link cost functions are piecewise linear are somewhat different from those obtained when the link cost functions are linear. In this case, there is a very substantial discrepancy between the results obtained using the simulated annealing algorithm and the greedy algorithm. For the above 50 node problems the simulated annealing algorithm results in a cost almost 4 times higher than that obtained using the greedy algorithm. This solution is effectively useless. Consequently, the time spent obtaining this solution can be considered wasted time. Hence, for these problems, the results obtained by the simulated annealing algorithm are not very useful. The same is true of the results obtained for the stepwise incremental cost function.

4.5.5 Objective Analysis of the Results

In general, it is difficult to perform any kind of objective analysis of the solution quality because the problem can have arbitrary cost functions. For this reason, no objective analysis of the quality of the cost function is included here.

However it was realised towards the end of this work that one useful approach may be to generate linear functions which upper bound and lower bound the cost function over the appropriate interval. The approach used by McGibney in [McG95] could then be used to obtain and upper and lower bound on the solutions. These could be used to determine the quality of the solution obtained using this approach. This only makes sense in an environment in which the link cost function is an increasing function of the used link capacity. Since this is usually the case, this should not be a problem.

4.6 Conclusion

This chapter serves to illustrate how the generic framework can be applied to solve a specific network design problem. The specific network design problem was described first. A model for the problem was then constructed and the mapping from this specific problem to the generic problem was discussed. A number of random example problems were then used to illustrate the use of the approach. These were then mapped to generic problems and the generic problem solvers were then used to obtain solutions to the generic problems.

The approach was analysed both in terms of the time taken to obtain solutions and the quality of the solutions obtained. It was found that for many of the cases the version of the simulated annealing algorithm was not particularly useful; the greedy algorithm always obtained better results and often very considerably better results. Interestingly, the simulated annealing algorithm that was used here sometimes obtained worse solutions when the parameters of the algorithm were chosen to increase the amount of processing time. Hence, increased processing time sometimes resulted in poorer solutions. When comparing solution quality against the time required to obtain the solution, the simulated annealing algorithm performs very poorly in the case of the piecewise linear and stepwise incremental cost functions.

The poor performance obtained in some cases above is not a specific problem with the framework – the problem is with the approach used to solve the generic problem. Specifically, the simulated annealing algorithm was found to obtain poor results above. This can most probably be attributed to the choice of state space and neighbourhood used in this work: an alternate choice of state space and neighbourhood could have been used to obtain a much better solution. An alternate choice of state space and neighbourhood could form the basis of an alternative solution approach within the framework.

CHAPTER 5 DIFFSERV/MPLS NETWORK CONFIGURATION PROBLEM

5.1 Introduction

Here, another specific problem is described that can be solved using the network optimisation framework discussed in Chapter 3. The specific problem under study here is that of configuring a core network to carry a set of customer demands with some QoS requirements. In this problem the customer demands are a set of *Data-based Virtual Private Network* (D-VPN) demands with associated qualities. The core network is implemented using *Differentiated Services* or *diffserv* and MPLS. The problem is to determine how best to configure the core network such that the demands can be carried with the requisite QoS while balancing the load on the network.

The chapter is structured as follows. First, an overview of both diffserv and MPLS is given. Next, the particular core network configuration problem considered here is described; the scenario is described and any assumptions made relating to the implementation are discussed. The problem is then formulated into a specific problem model. Issues associated with the mapping from the specific problem to the generic problem are considered next. Once an appropriate mapping function is defined, it is possible to apply the generic network design approach to solve the specific problem. To illustrate this, some diffserv/MPLS core network configuration problems are constructed; solutions to these problems are obtained using the generic approach described here. The solutions are validated by simulating some small networks, while larger problems are solved to illustrate the benefits of optimising the network in this fashion.

5.2 Diffserv and MPLS

Diffserv and MPLS are two relatively new technologies that have been developed within the *Internet Engineering Task Force* (IETF). Diffserv was developed to facilitate differentiation between packets as they are processed in network nodes with a view to providing some support for QoS. MPLS was developed to facilitate flexible routing in networks and hence provide much greater control over network routing. These two technologies can work well together to provide QoS across a single administrative domain.

Each of these technologies is considered in more detail below. Then, the current solutions for interoperation of these technologies are discussed.

5.2.1 Diffserv

Diffserv [RFC2475,RFC2474] was developed as a means to offer some support for **scalable** QoS over IP through service differentiation. The idea is simple: service differentiation is facilitated by giving each packet one of a standardised set of markings. These markings indicate how the packet should be treated by the network elements: different markings result in different treatments of packets, resulting in different QoS. The packets are processed at each node in the network according to their marking. Thus the QoS required by the packet is obtained.

Here, the initial efforts to develop an IP QoS solution which resulted in the Intserv architecture are first discussed. The limitations of the Intserv approach are identified. Then the diffserv architecture is described, including a description of standardised packet markings and some suggestions on how they should be treated at nodes. Devices which operate at the network edge to limit the amount of traffic entering the network are also detailed. Some implementation details follow: a typical queue and scheduler implementation is described and some implementation details for the edge traffic conditioners are also given. Finally, some examples of diffserv-based services are described.

Early Work on an IP-QoS Solution

Earlier work on IP QoS resulted in the so-called *Integrated Services* or *Intserv* architecture [RFC1633]. The Intserv approach borrows much from the more conventional telecommunications mindset in which QoS is offered on a **per-connection** basis: it focuses on individual IP flows with QoS requirements. Before such a flow starts transmission, it is necessary to determine whether or not the network can meet the requirements of the new flow without adversely affecting commitments made to other traffic. If there are not sufficient resources to carry the flow with the required QoS, the network identifies what QoS it can support for the flow and informs the source. The source then decides whether to accept this lower QoS or to abandon or defer the communications.

Intserv requires a signalling protocol to facilitate communications between the sources and the network. The *Reservation Protocol* (RSVP) [RFC2205, Whi97] was specifically developed to satisfy this need. Other signalling protocols could be developed that meet the requirements of the Intserv architecture but RSVP is currently the only standardised solution that meets these requirements. The use of RSVP signalling in the Intserv architecture in this manner results in the so-called Intserv/RSVP solution.

The Intserv architecture is a departure from the more usual thinking within the Internet community. Before Intserv was developed, the emphasis was very much on minimising the amount of critical and complex functionality in the network core and ensuring that most of it remained at the edge of the network [SRC84]. This was due to scalability and robustness concerns. This resulted in end-to-end protocols. For example, the *Transmission Control Protocol* (TCP) [RFC793] was designed to enable congestion control from the network edge. Security mechanisms such as *Secure Socket Layer* (SSL) are also implemented on an end-to-end basis. This philosophy typically results in robust, scalable solutions.

The Intserv architecture is not very suitable for wide-scale deployment in large core networks. The primary reason for this is that it suffers from scalability problems. Nodes in large core networks may have to simultaneously process many thousands of flows. It is necessary to keep state information for each flow traversing each node and to process and respond to signalling messages for all of these flows. Large core nodes would require very substantial resources to process all this signalling traffic. This heavy load is further compounded by the fact that RSVP connections are 'soft-state' connections – the flows periodically send keep-alive signalling messages to ensure that the resources remain reserved. For a large number of flows, the load induced by a large number of keep-alive messages could be significant. This large processing load and retention of substantial state information causes significant scaling problems: as the network size increases and the amount of flows with QoS requirements increases, the required node size increases dramatically. Note that it is most probably feasible to implement a large core Intserv/RSVP-based network if enough memory and processing power is used. However, such a router would necessarily be very expensive.

Aside from the scalability problems, an end-to-end QoS solution based on Intserv suffers from another critical problem: evolution to such a scenario is difficult. In order to support end-to-end QoS using the Intserv/RSVP architecture most of the routers between source and destination must support RSVP. Otherwise, the communications between the source and destination will not receive the required QoS. If parts of the connection do not contain RSVP aware routers, then the flow will only receive best effort service on this part of the connection. Deploying RSVP on a large scale can require costly upgrades to a very large amount of routers. Unless the benefits of this are very clear, operators will be reluctant to spend the large amounts of money required to implement an end-to-end Intserv/RSVP-based network. Also, a partial deployment is not so useful, since the key benefit of this architecture is predictable QoS; something that is not delivered if signalling is only supported by a minority of routers.

A Simpler Approach – diffserv

Diffserv is a fundamentally different and much simpler approach to offer some level of QoS in the network. Diffserv focuses on aggregates of flows rather than single flows as in the Intserv case. Diffserv does not require signalling and hence requires little intelligence in the network core. It operates simply by giving each packet an appropriate marking at the network edge and treating the packet in a way that is dependent on its marking at each node in the network core. Consequently, it is a much simpler and much more scalable approach than that of Intserv.

Diffserv focuses on facilitating **services**. These services are offered to the customer by the network operator. They are typically bulk transport services: the operator agrees to carry some quantified amount of the customer traffic while offering some level of QoS. The specifics of the service offering are detailed in a *Service Level Agreement* (SLA). This provides the basis for the common understanding of the service as well as defining the quantifiable commitments made by both parties, e.g. the QoS to be offered by the operator, and the amount of traffic that the customer can inject into the network.

Diffserv and Intserv are not mutually exclusive technologies: both can exist within a network. Indeed, the most likely development of today's networks is that both diffserv and Intserv will be used to facilitate end-to-end QoS. Diffserv will most likely be deployed in the network core because it is simple and scalable, and RSVP will most probably be used in the access part of the network to control access to the diffserv resources. Such an architecture is illustrated in Figure 5-1 and is discussed further in [XN99].

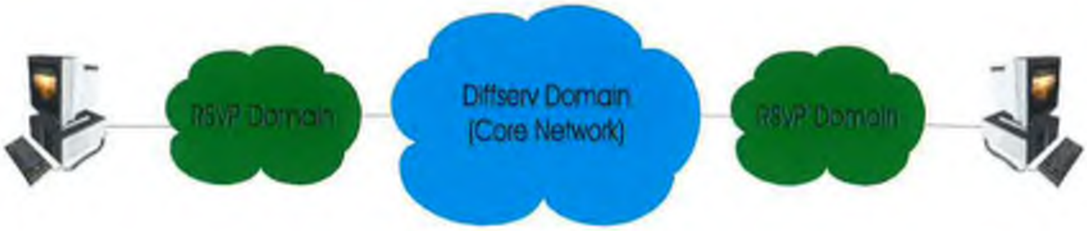


Figure 5-1: End-to-end QoS implemented with Diffserv and RSVP.

Diffserv is interesting from a research perspective because it is a radical departure from the more traditional circuit-oriented view of offering QoS (e.g. Intserv/RSVP, ATM). Diffserv has two key benefits over the more traditional view – lower cost and higher scalability – and hence if it can be used to offer a reasonable level of QoS to customers, then it is a more favourable approach than the more conventional alternative. However, it is not clear what level of QoS can be obtained using the diffserv approach and it is not clear whether reasonable quantitative guarantees/assurances can be offered with diffserv services. Here, some efforts are made to address these concerns.

The diffserv Architecture

A complex diffserv network capable of offering diffserv-based services over a large geographical area will consist of multiple diffserv *domains*. Each domain is characterised by having its own set of operating policies. These policies could, for example, define the specific packet markings and their corresponding treatments in the network, or the set of operating characteristics for the different services supported by the network. A single diffserv domain would probably be managed by a single operator.

Each diffserv domain can consist of a number of *Administrative Domains* (ADs) as shown in Figure 5-2. A diffserv domain may be very large and it is typically easier to manage a set of smaller domains than a single very large domain. Hence, the diffserv domain is decomposed into a number of ADs. The operator's policies could be applied consistently and uniformly across all of the ADs. A single AD is controlled by a single management system. The AD management systems interoperate to provide consistent end-to-end QoS over the diffserv domain. Each AD is then well defined by the system that manages it.

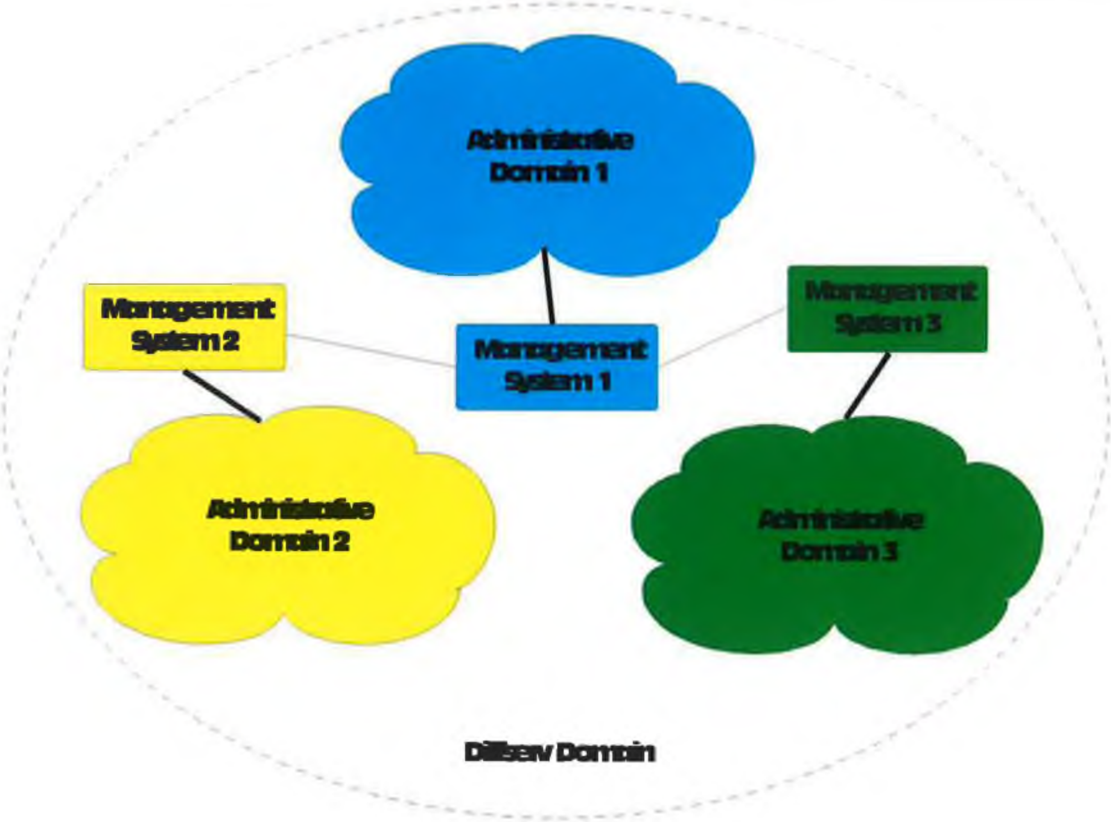


Figure 5-2: Administrative Domains within a diffserv domains.

Each AD has a well-defined boundary. Nodes at the boundary – so-called *boundary nodes* or *edge routers* – of the AD connect to other ADs or other networks. These may be ADs operated in the same diffserv domain, ADs operated by a different operator, or customer networks. Nodes that do not interface with other ADs or networks are called *interior nodes* or *core routers*. Every node in the AD must be either a boundary node or an interior node. A single diffserv domain connecting two customer premises is illustrated in Figure 5-3.

Boundary nodes and interior nodes have slightly different functionality. Boundary nodes perform *traffic conditioning functions,* which condition the traffic, on entry to the network. This conditioning is performed based on an SLA negotiated between the customer and the operator and involves ensuring that the traffic that enters the network is consistent with the SLA agreed with the customer. If the customer tries to inject very large amounts of traffic into the network, the conditioners will act to ensure that the customer's traffic does not have a detrimental effect on the network performance perceived by other users of the network. Interior nodes can simply route the traffic through the core treating each packet according to its packet marking. The architecture also has support for interior nodes performing traffic conditioning but this is not considered any further here: here, it is assumed that the interior nodes simply perform high-speed routing functions and the boundary nodes perform traffic conditioning and routing.

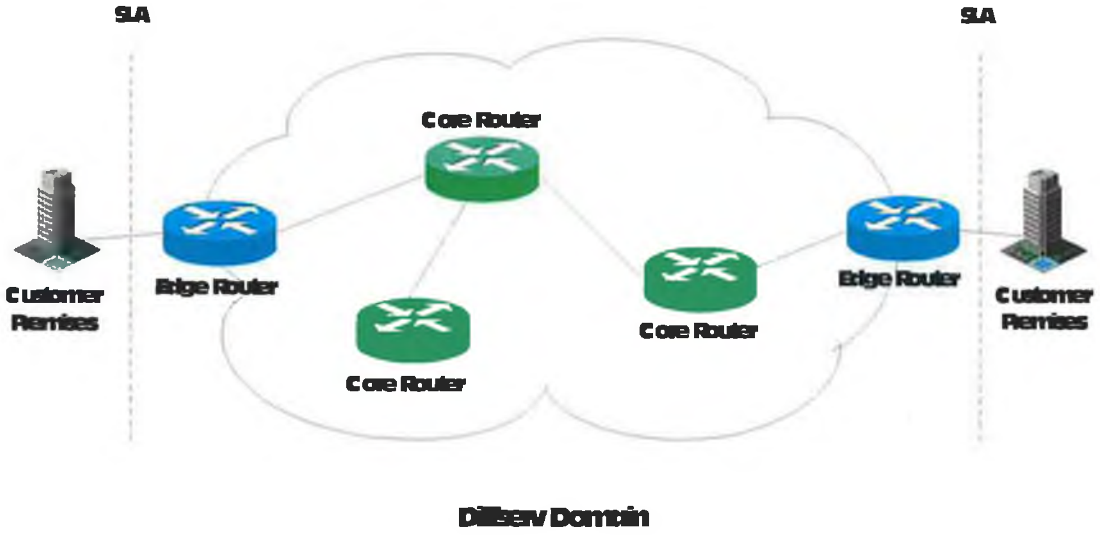


Figure 5-3: Illustration of diffserv network.

While the purpose of diffserv is to enable services to be deployed, the diffserv standardisation effort has not focussed on specification of services. Rather, the diffserv standardisation effort has focussed on standardising an architecture within which services can be implemented. This involved developing a set of building blocks that can be used to implement a flexible array of services. In theory, this means that an operator is able to offer a wide variety of services; in practice, the service offerings will probably be small in number. However, this approach is certainly more flexible than one in which the services themselves are specified. This means that if new 'killer' applications are developed that require specific diffserv service offerings to be deployed, the operator can implement new services to cater for these new applications without difficulty.

A more detailed explanation of the building blocks that can be used to offer diffserv services follows. First, the so-called *Per-Hop Behaviour* (PHB) – the way that a diffserv node processes and queues a packet – is discussed, followed by a discussion of the traffic conditioners at the edge of the network. The former facilitate service differentiation by treating packets differently, depending on their marking, while the latter limit the amount of traffic entering the network to ensure that the network resources are not overutilised, compromising the QoS.

Per Hop Behaviours and Diffserv Codepoints

The PHBs are the building blocks that have been standardised within the architecture to facilitate the development of diffserv-based services. These PHBs specify how a node processes a packet with a given packet marking. They can be used in conjunction with some specific traffic conditioners at the edge of the network to implement some specific services.

The PHBs specify how a node should treat a packet with a specific *Diffserv Codepoint* (DSCP) or packet marking. They are specified in terms of implementation requirements. The PHBs do not specify actual implementations; rather, they specify how to determine whether or not a particular implementation is conformant with the PHB specification.

A number of DSCPs and their associated PHBs are standardised. Three sets of PHBs are defined. These are:

- the *Expedited Forwarding* (EF) PHB;
- the *Assured Forwarding* (AF) PHB group;
- the *Class Selector* (CS) PHBs (which includes a default PHB).

The EF and AF PHBs are new and are intended to facilitate implementation of new services. The CS PHB is intended for legacy use: some existing networks use an approach similar to diffserv to prioritise some traffic (such as network control traffic). The purpose of the CS PHB is to ensure that such applications will work without modification in a diffserv environment.

The EF PHB [RFC2598] was designed to support services with stringent QoS requirements. It is intended for low loss, low delay services. The PHB specification for the EF service class specifies that sufficient resources must exist at the node egress to carry all the EF traffic. Specifically, the output resources reserved for EF traffic must be no less than the input EF rate. Hence EF traffic should experience minimal loss and delay. The EF service class is implemented using a single DSCP: all EF traffic in the network is marked with this codepoint.

The AF PHB group [RFC2597] was designed to be sufficiently flexible to facilitate deployment of service offerings with considerably different QoS characteristics. The AF PHB group consists of four classes, each of which contains three separate *drop precedences*. The drop precedences enable traffic to be prioritised within a single class. Higher priority traffic is less likely to be dropped or may experience a shorter delay at a node. Each of these drop precedences requires a distinct DSCP. Hence, twelve DSCPs are reserved for the AF PHB group. [RFC2597] specifies that traffic of a particular class must leave a node in the same order in which it entered, i.e. no packet reordering can take place within an AF class at a node. It also specifies that the higher priority traffic within a class must not experience a higher long term drop probability than lower priority traffic within the same class.

The CS PHBs are intended to support and be consistent with legacy applications that support some QoS. The particular IP packet header byte that contains the DSCP was also used in some IP legacy networks to support some service differentiation. In non-diffserv IP networks this byte is called the TOS byte. Some particular values of the TOS byte are typically used in legacy networks to differentiate between priorities of traffic; for example, network control traffic often has a particular TOS byte setting which differs from that of standard data traffic. The particular codepoints reserved for the CS PHBs are exactly consistent with the use of the TOS byte for legacy applications: a diffserv network will interpret TOS byte settings as indicating that the packet is marked with the CS PHB. Thus legacy applications using the TOS byte can work in a diffserv network using the CS PHBs. The CS service class simply specifies some set of priorities. The PHB then specifies that nodes implementing the CS service class must treat the different CS DSCPs with different priorities. The higher value CS codepoints should receive higher priority than lower valued CS codepoints.

Note that a node does not have to support all of these PHBs to be diffserv compliant. Similarly, an operator does not have to implement services using all of these PHBs. More DSCPs are reserved for local use within a diffserv domain, so an operator can choose to define a new PHB (assuming the equipment supports it) and associate it with some of the DSCPs that are reserved for local use. In this way an operator can implement and deploy a new service. In practice, it is more likely that an operator would use the DSCPs that are reserved for local use to implement another instance of the EF, AF or CS PHBs that has different operating characteristics if the standardised set of PHBs is not sufficient.

Traffic Conditioners

Traffic conditioners operate at the edge of the network to condition the traffic entering the network. This conditioning limits the amount of traffic of each type that enters the network. The traffic conditioners determine whether or not each packet conforms to an

SLA. If so, then the traffic conditioner performs no action; if not, then the traffic conditioner can perform one of a number of actions. The traffic conditioner can do one of the following:

- Drop the packet – the packet does not enter the network;
- Remark the packet – the packet is allowed to enter the network but with a codepoint different from that which it originally had;
- Delay the packet – the packet is allowed to enter the network, but must wait until such a time as its entry into the network is conformant with the SLA.

Which action is performed is dependent on the service that the operator implements.

Not all actions are appropriate for all services. For example, services implemented using the EF service class typically have low loss and delay requirements. A traffic conditioner at the node ingress that shapes traffic and causes delays to be introduced is not suitable. Also, remarking is not very suitable for EF services, since the packets would be marked down to lower quality services and they would arrive at the destination too late to be useful. AF-based services are more flexible and any of the above operations can be naturally applied to these services.

Implementation Details

The objective here is to determine how to configure the network to accommodate the customer demands in an efficient and robust manner. To do this, it is necessary to make some assumptions on the way the network is implemented. Here, some example implementations of components of the diffserv network are assumed and part of the overall problem is to choose appropriate parameters for these devices to deliver the required QoS.

The standards do not define how diffserv should be implemented. Those implementing diffserv must decide how to construct an implementation that is compliant with the standards. However, the standards do describe example implementations and it is very likely that many implementations will be strongly influenced by the example implementations. Such implementations are assumed here.

Two key components in the diffserv architecture are considered here. First an implementation of the combined queue and scheduler is described followed by a discussion on the implementation of the traffic conditioners.

Queue and Scheduler Implementations

Queues and schedulers exist in the network nodes to control access to resources. In general, large core network nodes can consist of multiple switching stages and some form of queuing and/or scheduling can occur at each stage. Here, the nodes are modelled simply as an output stage; any effects that can be attributed to characteristics of the internals of the node are ignored.

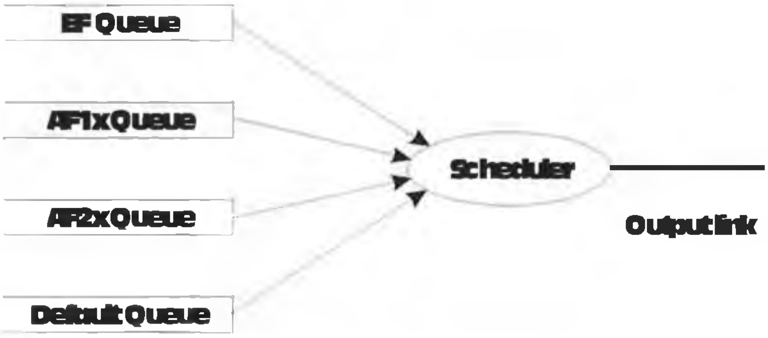


Figure 5-4: Queue/scheduler configuration assumed at the output stage of the diffserv code node. The scheduler determines which queue to take a packet from when the link becomes idle.

If the nodes are modelled simply as output buffering stages, then the nodes can be modelled simply as a configuration of queues and schedulers as shown in Figure 5-4. The constraints imposed by the standards on the different codepoints coupled with example uses of the codepoints give rise to some rather natural queue/scheduler configurations. For example, EF-based services are intended to be low delay services. Hence, it does not make sense to have large EF buffers, otherwise large delays could be introduced. AF classes have a reordering constraint – traffic from an AF flow cannot be reordered within a node. This implies that all the AF traffic for a particular flow should be queued in the same queue.

Using the above ideas, the queue scheduler system shown in Figure 5-4 was chosen. This consists of four queues: one for EF, AF1x, AF2x and a default or *Best Effort* (BE) queue. This is not standards compliant in two senses: first, the CS codepoints are not implemented and secondly, not all four AF PHB classes are implemented. However, it is easier to work with a smaller number of traffic classes and is sufficient to validate the approach used here to configure the network.

In this system, some of the queues have so-called *Active Queue Management* (AQM). This is a mechanism that operates on a queue that tries to prevent congestion by dropping small numbers of packets before the queue fills up. Its purpose is to increase the stability of the system, which has desirable effects such as increasing the overall throughput. AQM is particularly suited to adaptive traffic – such as TCP – which reduces its transmission rate when it detects a packet loss. It causes the adaptive sources to reduce their transmission rate in small steps. If no such mechanism was used, then the queue would fill and multiple packet losses could occur; for TCP sources, this would result in a very substantial reduction in the transmission rate. AQM attempts to increase stability in the system by causing single packet drops rather than multiple packet losses, which is more common when AQM is not used.

Floyd and Jacobson proposed the *Random Early Detection* (RED) AQM scheme in [FJ93]. RED is a particular AQM scheme in which the packet drop probability increases linearly with the mean queue occupancy. Clark and Fang proposed a variant of this that is applicable to a scenario in which traffic with different priorities or drop precedences use the same queue in [CF98]. The latter was termed *RED for In and Out traffic* (RIO). It operates by applying two separate RED mechanisms – one for high priority traffic and one for low priority traffic. It is assumed that traffic is marked as either high priority or low priority somewhere in the network (most probably the edge of the network) and that the low priority traffic should experience a higher loss probability than the high priority traffic. More AQM mechanisms have been proposed, which are variants of the above two – self-configuring RED [FKSS99], Fair RED [LM97], and others – but they are not considered here.

Here, AQM mechanisms are assumed for the AF and BE queues, since these service classes would most likely contain substantial amounts of TCP traffic. Such mechanisms are not used for the EF queue because the EF queue would typically not contain TCP flows, and these mechanisms are most suited to queues with substantial amounts of adaptive traffic. The AF queues contain traffic with different drop precedences and hence an RIO AQM mechanism is used; the BE queue contains homogeneous traffic (in the sense that no BE traffic has priority over other BE traffic) and hence an RED mechanism is used.

The scheduler assumed here is a *Weighted Round Robin* (WRR) scheduler. In this scheduler, different weights are given to each queue: the queue weights are proportional to the proportion of the link resources that each queue should receive when the system is congested. When the system is uncongested, there can be more flexibility in the system, i.e. some queues can obtain more than their allocated capacity because others need less.

The WRR scheduling mechanism used here is based on the notion of 'rounds'. In each round, some 'credit' is allocated to each queue to be used within the round. If a packet arrives at the queue, and the queue is next to be served, the scheduling mechanism first checks to see if the queue has accumulated sufficient credit to allow transmission of the packet. If so, then the packet is transmitted on the link and the amount of credit associated with the queue reduced; otherwise, other queues are examined. If every queue is always ready to transmit data, then the queues will be given access to the link in proportion to their weights. A more detailed description of the operation of the WRR scheduler assumed here is in Appendix B.

Note that this is one possible implementation – other implementations could be developed based on Priority Queuing, Class-based Queuing or some other queuing and scheduling mechanisms. These different approaches may result in better performance in some respect. The scheduler described here, however, meets the design requirements of low EF delays, controllable service differentiation for the other classes and a minimum resource allocation for BE traffic. Hence it can be used to deliver a reasonable level of QoS.

Traffic Conditioner Implementations

Traffic conditioners identify whether traffic conforms to some prespecified SLA and if not, the traffic is 'conditioned' according to the SLA so as not to have a detrimental impact on the network performance perceived by other network users. Since the traffic conditioning is only applicable in a QoS context, it is not applied to BE traffic.

Two basic methods of operation of conditioners have been proposed. The first is based on estimation of a transmission rate over a time interval and action will be taken on traffic that exceeds this rate; the second is based on a token bucket mechanism.

In the first approach – the so-called *time-sliding window* approach – described in [RFC2859], the current transmission rate of the source is estimated. If the current rate estimate exceeds the target rate, then packets are dropped probabilistically, with drop probability $(r_{\text{est}} - r_{\text{tar}})/r_{\text{est}}$ where r_{est} is the estimated rate and r_{tar} is the target rate.

The alternative approach is based on token buckets [RFC2698]. These buckets obtain tokens at the target rate. If a packet arrives and all the tokens have been exhausted, then the packet is non-conformant, otherwise the packet is conformant, and tokens corresponding to the packet size are removed from the bucket.

The above two mechanisms to determine which packets in a stream are conformant to some given profile are parameterised by a rate and another parameter. In the case of the time sliding window approach, the parameter is the size of the sliding window and in the case of the token bucket approach, the second parameter is the bucket size. These two parameters perform similar functions, i.e., they specify over what interval the rate is measured.

The above two mechanisms can be used to determine whether or not packets conform to a particular profile: what happens if they don't conform is a separate issue that is closely coupled to the service definition. Conformant packets typically enter the network unchanged. Non-conformant packets, on the other hand, can be treated in three separate ways – dropping, shaping or remarking. The first option would be used in a service in which it is critical to limit the amount of traffic entering the network. The second option would be used in a case in which it was critical to limit the amount of traffic entering the network, but the particular user of the service was willing to put up with some delays rather than incur extra retransmission. The final case is applicable in a situation in which it is not so critical to limit the amount of traffic entering the network and utilisation of the resources allocated to the service is as important as assuring QoS targets are met.

In some cases, the traffic arriving at the conditioner is not marked at the source. In this case, the conditioner must also mark the traffic. This is done in accordance with the agreed SLA. Marking at the source is more desirable since the source can identify high and low priority traffic more easily. If the traffic arriving at the conditioner is unmarked and the conditioner is marking it, it may assume that all traffic is equal and mark accordingly. This can cause some applications to receive poorer QoS than they would if source marking was performed.

Service Examples

Here, some services that can be implemented using diffserv are described. First, the so-called Virtual Wire service is described. Then the so-called Olympic service model is described. How the Olympic service model can be used to implement concrete services is also discussed.

Virtual Wire Service

The Virtual Wire[12] service was designed to enable dedicated or leased line circuits to be replaced with diffserv-based IP transport. The service is implemented using a suitably configured EF PHB and a suitable choice of traffic conditioner.

The customer sees this service as a direct replacement for services based on TDM-switched dedicated lines. As such, the service is parameterised by a single rate. In principle, implementing the service over a diffserv core network permits dedicated line services with almost arbitrary capacity to be emulated. This service would typically be a costly service that would be used to carry traffic that is very sensitive to loss and delays. Typical applications would include interactive voice or video applications, time-critical transactions, stock prices, etc.

In order to implement the service, it is necessary to specify how the service should be implemented in the operator's core network such that the user does not perceive any difference between the service implemented using diffserv and a dedicated circuit. The following issues arise when considering this problem:

- Configuration of edge conditioners;
- Ensuring timely delivery of packets;
- Resource reservation in the network;
- Routing of flows in the core network.

Each of these is discussed separately.

To offer this service, in which there are strict bounds on the QoS, network resources must be strongly protected. If there are fluctuations in the traffic, then the network will not be able to offer the required QoS. Hence, no fluctuations in the traffic are permitted. Only traffic conforming to the negotiated rate is permitted into the network – any traffic exceeding this rate is dropped at the network ingress.

Ensuring timely delivery of packets is not trivial. The diffserv core network is assumed to contain links of much higher capacity than that required by the customer: diffserv

[12] The Virtual Wire terminology was first introduced in an IETF contribution. There were problems with this document and at the time of writing no definitive version of this document existed. The essence of the service is described here.

was designed for large traffic aggregates after all. This, coupled with the fact that the core network will be switching other traffic means that the inter-packet spacing for packets arriving at the egress router will differ from the inter-packet spacing generated at the source as illustrated in Figure 5-5.

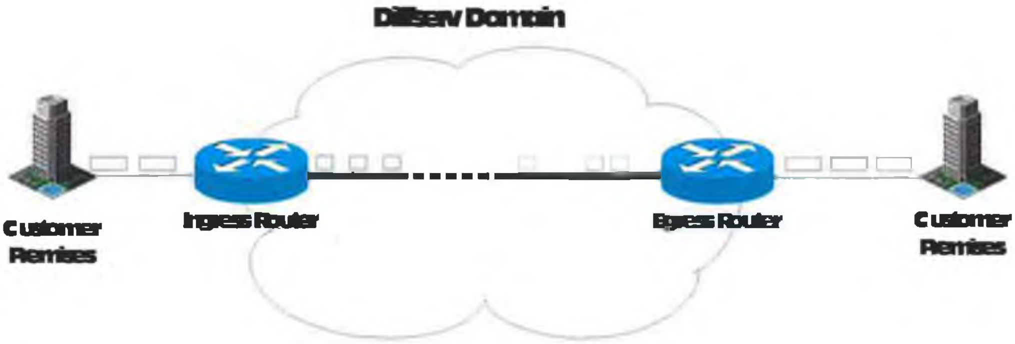


Figure 5-5: Inter-packet spacing is changed in a diffserv domain. The higher transmission rates in the core network result in smaller packet transmission times – hence the packets appear to be compressed in the core network.

To ensure that the service delivered to the customer is equivalent to a service based on dedicated connections, the packets must arrive at the egress router in time. If they arrive too late, then there will be an observable difference between the delivered service and the service that would have been obtained had a dedicated circuit been used. In order to assure this, some bounds are imposed on the *jitter* – the variation in the delay experienced by each packet as it traverses the network. A so-called *jitter window* is defined: this defines the time interval in which the packet must arrive at the egress router. If the packet arrives at the egress router outside this jitter window, then the service delivered to the customer is not equivalent to a dedicated circuit based service.

There are problems associated with delivery of this service. These problems are, in part, related to problems with the definition of the EF PHB [RFC2598] and are discussed in detail in [BBC01]. The problems arise when trying to assure end-to-end delay and jitter for EF traffic. The authors highlight extreme cases in which the flows are synchronised and large delays can be incurred for some flows at many nodes. This results in a large end-to-end delay.

It is worth noting here that these problems arise only when there is sychronisation between sources, and even then it is rare that such a situation would arise. However, if the objective is to make strong assurances to customers then these extreme case must be considered. One approach to avoiding this situation is to ensure that there is some random component to the source traffic. Indeed, it is likely that there would naturally be such a random element to the traffic, since the traffic would likely come from different applications and consist of different size packets. In this case, the extreme problems highlighted in [BBC01] would not arise.

Olympic Service Model

In the Olympic service model, there exist three distinct classes of service – Gold, Silver and Bronze. Gold service naturally offers the highest quality of service, followed by silver and then bronze. These services are not intended for applications that are very intolerant of loss and delay. Rather, they are intended for services that can suffer some loss and delay without making the application unusable.

The Olympic service model is used to illustrate how the AF PHB group can be used. The Gold service can be implemented using one AF class, the silver using another and the bronze service using yet another AF class. The network can be designed and managed such that different AF classes have different operating characteristics. For example, the AF class used to implement the gold service can be configured such that the mean nodal packet delay is, say, 5ms, and loss ratio of say 0.5%, or that the ratio of load to available resources is fixed at some parameter.

Particular services can be implemented by adding traffic conditioners to the scenario. With the AF service classes, there is some flexibility when choosing the traffic conditioners. The traffic conditioners may drop, shape or remark packets. In the case of services implemented using an AF class, it is most likely that the conditioners at the edge of the network will remark traffic exceeding the pre-negotiated profile. So, for example, if a customer attempts to load the network with more traffic than is agreed in the SLA, then the excess traffic can be marked down to a lower priority.

To implement a service, it is necessary to define how the customer traffic entering the network is measured and what actions are taken if the customer exceeds the agreed traffic. One way that this could be done could be to set a limit on the amount of traffic of each priority that enters the network. If the traffic of any of these priorities exceeds the agreed limit, then the packets are marked down to a lower priority. It may also be necessary to impose a limit on the amount of lowest priority traffic entering the network. If this is exceeded, then the traffic could be dropped. It may also be necessary to impose a limit on the overall traffic using a particular service.

5.2.2 MPLS

MPLS [RFC3031,Arm00] is a technology that was developed within the IETF to facilitate more control over routing in networks. It allows paths to be configured arbitrarily in the network and arbitrary traffic to be carried on each path. Once a packet enters one of these paths, it will then follow that path through the MPLS domain and reach the other end of the path (assuming it doesn't get dropped somewhere on the path).

This is in contrast to more traditional forms of routing in data networks in which each router typically made an isolated routing decision for each packet: each router determined how to forward each packet more or less independently of the other routers in the network – arbitrary paths are not possible. RIP [RFC2453], OSPF [RFC2328] and ISIS [RFC1142] all operate by distributing network information amongst the nodes such that each node can make intelligent routing decisions. This has worked very well throughout the network, but lacks the flexibility afforded by MPLS.

This flexibility can be used manage the load on the network in a more controlled fashion, to perform traffic engineering functions and also to isolate different customer traffics of different priorities and to treat them differently.

The basic mechanism used by MPLS to construct routes is *label swapping*. Each packet on an MPLS path has a label associated with it. This label can (and probably will) change as the packet traverses the network. At each node, the label of each incoming packet is identified. A table look-up is then performed to find the outgoing label and interface for the given incoming label and interface. Then the packet is switched to the appropriate outgoing interface and assigned the appropriate output label. Label swapping is not such a new idea: it is also used in ATM. The ATM *Virtual Channel Identifier/Virtual Path Identifier* (VCI/VPI) can be considered to be equivalent to labels. Moreover, MPLS can be implemented over ATM such that the labels are exactly the VPI/VCI headers.

The label information is distributed through the network using a label distribution protocol which was specifically developed for MPLS [RFC3036].

The LSPs can be set up using information in the routing tables or directly from the management system. If the layer 3 routing tables form the basis of the LSP network, then packets would follow the same routes if the MPLS functionality was not used.

Alternatively, other means – signalling or management actions – can be used to construct arbitrary paths through the network.

MPLS does not have any inherent support for QoS. It was designed to facilitate flexible routing; QoS concerns were not initially taken into account when designing MPLS. However, it has since been recognised that MPLS, in conjunction with a signalling protocol – be it constraint routed LDP [Jam01] or *RSVP with Traffic Engineering extensions* (RSVP-TE) [ABG01] – can be used to support QoS in some way. Both CRLDP and RSVP-TE provide a means to configure paths in the network that have some QoS parameters associated with them. Examples of such parameters could include a peak and a committed transmission rate.

MPLS is quite a complex technology and much more could be written about it here. In particular, the MPLS LDP could be described at length. However, the emphasis here is on illustrating how the generic network design approach can be employed in some specific problem. For the problem of interest here, a comprehensive description of MPLS is not necessary. Rather, it suffices to note that MPLS can be used to implement arbitrary routing in a network, which is most suited to the generic problem solver.

5.2.3 Diffserv over MPLS

Using both diffserv and MPLS in a network enables an operator to implement a network in which QoS can be offered with some level of assurance. Both diffserv and MPLS can offer some level of QoS to customers without requiring the other. However, diffserv does not have any routing support; hence, it is difficult to control exactly how traffic is routed in the network and hence make any quantitative guarantees to the customers. Current MPLS implementations do offer some QoS support – they enable paths with an associated bitrate to be configured in the network. This approach can suffer from the scalability problem of the Intserv/RSVP solution. Consequently, a more scalable approach which combines the benefits of MPLS and diffserv is desirable. Combining diffserv and MPLS enables the operator to have sufficient control over the network to offer diffserv-based services to customers with some level of QoS assurances.

Diffserv and MPLS operate at different layers in the protocol stack. Some effort is therefore required to make these two technologies work together. Diffserv operates at the network layer – the IP header contains the DSCP. MPLS is said to operate between the link layer and the network layer. MPLS packets are therefore switched without looking at the IP header, and the DSCP cannot be used to identify the QoS required by the packet. The problem then is how to determine what PHB should be used to treat each MPLS packet as it is switched through the node.

This problem has been identified within the IETF and two approaches have been proposed to solve the problem [LWD01]. The first approach uses the MPLS label to determine how the packet should be treated at the node; the second approach uses a field in the MPLS header coupled with the label to determine the priority of the MPLS packet.

In the first approach, the traffic on each LSP is mostly homogeneous, i.e. all the traffic is of the same diffserv class. The problem is then to determine how the traffic on the LSP obtains the correct treatment at each node in the network. The solution proposed in this approach is to infer the required treatment from the MPLS label: the node will contain a table containing the set of LSPs and their associated treatments. This is the so-called L-LSP solution.

In this solution, not all L-LSPs contain exactly homogeneous traffic in the sense that some L-LSPs may, of necessity, carry multiple DSCPs. This is the case for AF traffic. The AF class imposes a reordering constraint: packets within the same AF class on a particular flow must not be reordered at a node. If packets from the same connection but with different drop precedences used different LSPs, then such a reordering of packets could occur. Consequently, all the AF packets associated with a single connection must use the same LSP. Hence, multiple codepoints are carried in a single LSP. In this case, it must be possible to differentiate between packets of different drop precedences within an LSP. This is done using two of the three bits in the experimental EXP field in the MPLS header. Some values have been standardised for these bits.

In the second approach, both the MPLS label and the EXP field are used to infer the priority of a packet. Traffic consisting of multiple codepoints are typically carried on a single LSP. This is the so-called E-LSP solution. The EXP field is 3 bits long, so up to 8 different codepoints can be carried on a single LSP.

While the L-LSP solution does use the EXP field in the MPLS header, it is much less flexible than the E-LSP solution. In the L-LSP solution, specific fixed values of the EXP field are used to indicate the drop precedence of an AF packet. In other cases, the L-LSP carries purely homogeneous traffic and the EXP field is unused. For example, if the LSP carried EF traffic, or one of the CS codepoints, this would be the case. In the E-

LSP solution, much more arbitrary mappings from the label and the EXP field to the DSCP can be used. Consequently, the traffic in an E-LSP can be much more heterogeneous. These mappings can be chosen by the network operator.

5.3 Problem Description

Here the specific problem that will be used to illustrate the use of the generic network design approach is described. In this problem, the technologies described above are used to implement a core network that can offer diffserv-based D-VPN[13] services to customers. First, the customer service offerings are described. Next, a specific way of implementing the network to enable the services to be implemented is discussed. Then the problem of configuring the network to effect load balancing is discussed. A similar description of the problem studied here was discussed in [MBC00].

5.3.1 Diffserv Service Offerings

The network operator offers D-VPN services to the customers. The D-VPN service offered consists of a set of different interconnects with QoS support between customer locations. The customer may have any number of interconnects between its premises. Here, it is assumed that the customer desires a fully connected network. It is assumed that these interconnects are static: their characteristics do not change with time, although some real service offerings may permit interconnects with some dynamic characteristics.

The customer D-VPNs can, in general, carry quite different types of traffic – traffic with different characteristics and different QoS requirements. For example, the D-VPN could carry delay sensitive interactive voice and video traffic as well as, say, delay-insensitive mail traffic; WWW traffic may also be carried with some delay or throughput requirements: the network may need to support some delay sensitive secure transaction processing. Many more examples of different traffic types with different requirements could be considered.

From the customer perspective, the network is implemented with diffserv. The customer sees only a diffserv interface into the network. It is not apparent to the customer that the

[13] VPN is used here in the context of data networking. A different notion of a VPN exists in conventional telephony: there, a VPN is a way of using the resources of the PSTN to implement something that looks like a private network. The VPN can be used to implement a private numbering plan, for example.

network is implemented using MPLS. Indeed, the customer probably does not care very much how the core network is implemented as long as it is reliable and can offer the required QoS.

The diffserv standards stipulate that an operator can offer diffserv-based services with either quantitative or qualitative assurances. Services with quantitative assurances are ones in which the operator makes some measurable assurances to the customer; qualitative assurances are ones in which the operator makes some more vague assurances to the customer. Qualitative assurances are not strictly measurable. Quantitative assurances are more difficult to deliver.

For the purposes of this problem it is assumed that the customers desire quantitative assurances. Such assurances are desirable for the customer, so the customer can predict whether the service is suitable for the customer applications. Also, services with quantitative assurances are more difficult to deliver, which makes the research problem more interesting.

In reality, it is likely that customers will be able to obtain services with either qualitative quantitative assurances. The latter will be more expensive, but may be most suitable for some customers, while other customers who may not require such strict assurances will be able to opt for the lower cost services with only qualitative assurances.

In this problem, it is assumed that the operator offers premium services – the interconnects with associated QoS – based on the EF and AF PHBs and a default BE service which has no associated QoS. The EF-based services can be used to carry traffic with stringent delay and loss requirements and the AF-based services can support traffic with less stringent requirements but requiring service better than BE. Limits are imposed on the amount of high-priority traffic the customer can inject into the network but the customer can generate as much BE traffic as desired.

This is a somewhat realistic scenario, but it does have one important drawback: the problem does not cater for the Class Selector codepoints. These are required in any diffserv implementation, but are ignored here, although the problem considered here could probably be extended to include such traffic if the services using the traffic were defined. The purpose is to illustrate the use of this approach in a somewhat realistic problem rather than to obtain concrete solutions to very concrete problems.

The QoS parameters associated with each service are specified in terms of a delay and a loss. In the case of the EF-based services the QoS parameters associated with the service are the peak end-to-end delay and the peak loss ratio. In the case of the AF-based services, delay and loss parameters are specified for the situation in which the customer generates traffic conformant to the agreed profile. The customer can exceed this and the network will still attempt to deliver the excess, but it may be subjected to high loss and will also incur extra delays on both high and low priority traffic in buffers.

The different services operate in a slightly different manner. Specifically, they treat traffic in excess of the agreed rate in different ways. For the EF traffic, a limit is imposed on the peak traffic that a customer may inject into the network. If the customer exceeds this limit, then traffic is dropped at the ingress to the network. All of the EF traffic that enters the core network will have delay and loss assurances.

The EF-based service offerings described here are Virtual Wire service offerings as described above. As mentioned above, such services can suffer extreme synchronised behaviour which makes it difficult to offer any reasonable delay assurances. Here, it is assumed that some randomisation is either present or is introduced into the EF traffic to ensure that such synchronisation is very unlikely.

The AF traffic is a little more complex. AF supports the notion of multiple drop precedences. A source can generate traffic and if it generates traffic in excess of its agreed profile the traffic can be reduced in priority by giving it a lower drop precedence. Then, if there are sufficient resources available in the network, the network will deliver the lower priority traffic. If there are insufficient resources, the lower priority traffic will be dropped first.

Each AF class supports up to three drop precedences. However, AF-based services can be implemented using just two service classes. This operation is assumed here. Packets marked with AFx1 are high priority AF packets, while packets marked with AFx2 and AFx3 are low priority packets.

The AF traffic is also permitted to be more variable than the EF traffic. Since the assurances required by the AF traffic do not need to be as stringent as those required by the EF traffic, it is not necessary to control the traffic entering the network in such a conservative fashion. The traffic conditioner for the AF traffic conditions traffic on both a mean and a peak rate: if the peak rate is exceeded, traffic is marked down to a lower precedence and if the mean rate is exceeded over some time period, traffic is also marked down. This variability is desirable in AF-based service offerings, since customers will be able to negotiate SLAs that match their variable requirements better.

The conditioners can either be window based or bucket based as in [RFC2859] and [RFC2698] respectively. These are discussed above. In this work, the conditioners were assumed to be bucket based, since it is easier to offer assurances in a system in which the conditioners are bucket based.

The customer has some set of applications with some associated QoS requirements. Since the operator defines the level of QoS for each of the service offerings, the customer can determine which service offerings are most suited to which applications. The customer may decide to choose a number of different service offerings for the different applications, or, alternatively, the customer may choose to use a single service for all the traffic to minimise management overheads. The latter option would be effectively a trade-off of service costs against management costs: the customer chooses to pay more to have a single service which is easier to manage rather than to pay less to the operator and have higher administration costs.

Here, the demands are assumed to be point-to-point demands. Diffserv permits services of broader scope. Indeed, as mentioned above, these are more attractive to customers, and they can result in efficiency gains for operators if managed properly (see [DGG99] for a more detailed discussion of this). Such traffic can be decomposed into sets of 1:1 demands as part of the mapping function if some knowledge of the demands is assumed, as described above. This is a non-trivial problem and is certainly an interesting area for further research, but it is not considered here.

5.3.2 Network Implementation

In the core network implementation considered here, it is assumed that the SLAs are carried over MPLS LSPs, thus permitting flexible routing. In this problem L-LSPs are used because they are the simplest to use and manage. Extending the problem to the use of E-LSPs is not principally more difficult. However, it does require that some demands of be routed together, which imposes an extra constraint on the problem. This extra complication is not considered here.

Here, it is assumed that a core network already exists. This core network consists of a set of nodes interconnected by a set of links. In general, the network can consist of both core nodes and boundary nodes: the boundary nodes have slightly different functionality. Such an architecture is discussed in [XN99]. Note that in many cases, boundary nodes can be co-located with core nodes, providing both high capacity core network switching and access to/from customers. This work can be applied to the general case, but in all of the examples below it is assumed that every core router is co-located with a boundary router. As such, every core router has access to/from customers.

The output interface to each node has a queue/scheduler system that controls access to the link. This is assumed to be the same as that described in section 5.2.1 above. This is one possible implementation, which is not necessarily the best, but serves to illustrate the use of the design approach in this context.

The interconnects between the nodes are bidirectional links with some associated capacity. These may not necessarily be physical links – for example, they may be SDH containers or ATM Virtual Pipes – but this is not important. Thus, the network is implemented in order to offer the services described above.

5.3.3 Optimisation Problem

The problem then is to determine how to configure the network efficiently to accommodate the demands – the set of SLAs – while ensuring that the required level of service is delivered. The objective is to ensure that the QoS targets for each service are attained while balancing the load on the network. Load-balancing in this manner is desirable, since it makes the network more robust in the face of unexpected congestion/traffic patterns.

Here, only those demands that have some associated QoS are parameterised and explicitly routed in this problem: the BE demands are not routed explicitly through the network. The capacity remaining after routing the demands with QoS can be used by the BE traffic.

This approach can cause all of the link capacity to be consumed by the priority demands leaving no resources available to the BE traffic. Clearly, this is highly undesirable. This can be avoided by the reserving capacity for the BE traffic in advance – this capacity will not be available to the high-priority traffic in the optimisation problem. This reserved capacity can be removed before the optimisation problem is solved and replaced when the routing for the high-priority demands is determined.

Load balancing is a desirable objective since this ensures that the network is somewhat resilient to spikes in load or changes in traffic demands. Also, if the demands are forecast demands and the network designer is determining how to plan the network, a solution with a balanced load will be more resilient to errors in the forecast demand.

Here, there are two key components to configuring the network: the routes have to be determined and the node configuration parameters must be chosen. Specifically, the parameters associated with the queue/scheduler system as described above need to be determined.

It is possible to devise a problem formulation that includes both the route configuration element and the node configuration element. If this was solved without decoupling the route configuration and node configuration element, then the load balancing would have to be implemented using some packet level measures – queue delays, for example. This, in turn, would require that the traffic be defined using detailed packet level models. This formulation would result in very complex problem.

The packet-based formulation also suffers from another problem: great sensitivity to the traffic model used. This makes the design procedure sensitive to the traffic modelling assumptions. Of course, if the traffic models are accurate, then the optimal solution may result in a much more efficient network configuration. However, if the traffic models are inaccurate (as internet traffic models usually are – see [PF97] for more discussion on this), then the network performance may be worse than predicted.

Here, no assumptions are made on the nature of the source traffic, although the traffic is conditioned at the ingress to the network, so the characteristics of traffic entering the network are known. Using this approach, the network, if configured appropriately, can deliver the required QoS with a high probability. This approach does have the disadvantage that it may result in inefficient use of network resources, but the emphasis here is on QoS delivery rather than operational efficiency. This approach can result in inefficiency due to over-conservative resource allocation to ensure QoS delivery. However, the schedulers should offer sufficient flexibility to enable underutilised reserved premium resources to be used by lower priority traffic.

5.3.4 Application of the Design Problem

As can be seen from the results presented in the previous chapter, it can take a considerable amount of time to obtain a solution to the generic problem using the problem solvers developed throughout the course of this work. Moreover, as is mentioned above, the generality of the generic problem makes it difficult to find an algorithm which can solve the problem quickly for all but the very smallest problems. Consequently, the approach used to solve this specific problem is not applicable for making real-time decisions.

The work described here could be usefully employed to solve the optimisation problem in the context of a tool which works offline to ensure that the load on the network remains balanced. This could operate in a network in which new demands are routed using some routing protocol such as, say, OSPF. This can result in a network in which the load is not balanced. The tool could determine a routing for the demands and compare it to the existing route configuration. If the new route configuration is substantially better than that which is configured on the network, then the network could be reconfigured according to the new route configuration. The problem of determining the optimal approach to reconfiguring the network is not considered here.

One possible problem with this approach is that the number of demands could conceivably be very large. The time required to obtain a solution to the problem increases with the amount of demands. Consequently, the time taken to obtain a solution could be large. The problem can be solved by aggregating the demands.

If the demands are aggregated, then the number of demands to be routed can remain constant or almost constant. The solution time should then be quite constant. The disadvantage of aggregating the demands in this manner is that the load may not be balanced as well as it could be: if the demands are not considered in an aggregate fashion, the load balancing can be performed with finer granularity. There are issues with this: the approach used to aggregate demands in particular, but these are not considered here. For the purposes of this discussion, it is sufficient to note that the size of the problem can be limited by aggregation and that the network optimisation framework can then be used in an offline tool which ensures that the load remains balanced on the network.

5.4 Specific Problem Inputs and Outputs

Here, the specific problem is defined. The specific problem defined here is not defined in sufficient detail to be formalised and solved. Rather the specific problem is defined in terms of the input parameters and the required output parameters. The problem cannot be formalised because the objectives are not stated in sufficient detail.

The inputs to the problem are as follows[14]:

- N : the set of nodes in the network;
- L : the set of candidate bidirectional links in the network;
- $G(N, L)$: an undirected graph that defines how the nodes and links are connected;
- c_l : the capacity of link $l \in L$;
- Z : the set of customers;
- E_z : the set of directed EF SLA demands[15] for customer $z \in Z$;
- $E = \bigcup_{z \in Z} E_z$: the set of directed EF SLA demands;
- $(o_e, p_e, r_e^{\text{peak}})$: the characteristics of EF customer SLA demand $e \in E$ – o_e is the source node, p_e is the destination node and r_e^{peak} is the peak rate of the demand;
- A_z: the set of directed AF customer SLA demands for customer $z \in Z$;
- $A = \bigcup_{z \in Z} A_z$: the set of directed AF SLA demands;
- $(o_a, p_a, r_a^{\text{peak}}, r_a^{\text{mean}}, t_a^{\text{on}}, t_a^{\text{off}}, s_a)$: characteristics of the AF SLA demand – o_a is the source node, p_a is the destination node, r_a^{peak} is the peak rate, r_a^{mean} is the mean rate, t_a^{on} is the mean on time, t_a^{off} is the mean off time and s_a is the class for demand $a \in A_c$;
- $(d_{\text{EF}}, l_{\text{EF}})$: delay and loss bounds for the EF traffic;
- $(d_{\text{AF}i}, l_{\text{AF}i})$: delay and loss targets for the AF traffic of class $i \in \{1,2\}$ – these targets apply to the AF traffic of the class with drop precedence AFi1;
- y_l: fraction of the link capacity that can be used by the priority traffic – $(1 - y_l)$ is the fraction of the link capacity reserved for the BE traffic.

[14] A large number of variables are used to define this problem. Hence some of the variable names that were used in the enterprise network design problem are used here in this chapter for a different purpose.

[15] The demands in this problem are the SLAs: the term 'demands' is used throughout the following to refer to the set of SLAs.

Most of the inputs require little explanation but a some of the parameters warrant some comment. The EF traffic is characterised by a peak rate. This could be for leased-line type applications.

The AF traffic is characterised by more parameters: the peak and mean rates and the mean on and off time, as well as the service class[16]. These parameters can be used to condition traffic at the network ingress. It is assumed here that these descriptors can be used to characterise the AFi1, where i is the service class of the traffic. The AF service class permits more traffic than this to be input to the network: any traffic which is non-conformant with the above descriptor is given a different drop-precedence, and is more likely to be dropped in congested conditions.

The delay and loss bounds for the AF traffic then only apply to the AFi1 traffic. The network can deliver service in which the AFi1 traffic loss and delay are below the loss bound in all but the most difficult situations. No delay and/or loss assurances are offered for drop precedences other than AFi1, since it is not known how much of this traffic will enter the network.

The outputs of this network configuration problem are

- R : a route configuration – the set of routes for all of the demands;
- the queue/scheduler configuration parameters comprising of:
  - → $(w^{l_{\text{up}}}_{\text{EF}}, w^{l_{\text{up}}}_{\text{AF1}}, w^{l_{\text{up}}}_{\text{AF2}}, w^{l_{\text{up}}}_{\text{BE}})$: the scheduler weights for the scheduler controlling access to link $l \in L$ in the upstream direction;
  - → $(w^{l_{\text{down}}}_{\text{EF}}, w^{l_{\text{down}}}_{\text{AF1}}, w^{l_{\text{down}}}_{\text{AF2}}, w^{l_{\text{down}}}_{\text{BE}})$: the scheduler weights for the scheduler controlling access to link $l \in L$ in the downstream direction;
  - → $(q^{l_{\text{up}}}_{\text{EF}}, q^{l_{\text{up}}}_{\text{AF1}}, q^{l_{\text{up}}}_{\text{AF2}}, q^{l_{\text{up}}}_{\text{BE}})$: queue lengths for the queues served by the scheduler controlling access to link $l \in L$ in the upstream direction;
  - → $(q^{l_{\text{down}}}_{\text{EF}}, q^{l_{\text{down}}}_{\text{AF1}}, q^{l_{\text{down}}}_{\text{AF2}}, q^{l_{\text{down}}}_{\text{BE}})$: queue lengths for the queues served by the scheduler controlling access to link $l \in L$ in the downstream direction;

[16] It is assumed here that an on-off model is used to model the demand. This is a worst case model. The demand will most likely be less bursty than this, but assuming the worst case is necessary to make delay and loss assurances.

→ AQM configuration parameters for the queues served by the scheduler controlling access to link $l \in L$ in the upstream and downstream direction – these are listed separately.

A substantial amount of parameters need to be defined to configure the queue/scheduler interface at each link: the scheduler weights need to be defined, the queue lengths need to be defined and the AQM configuration parameters for some of the queues need to be defined. The scheduler weights determine what proportion of the link is allocated to each queue. The meaning of the queue lengths parameters is quite clear. Here, it is assumed that they are measured in packets and that there is a standard packet size using each queue: this can be an average packet size.

The AQM configuration parameters at each interface consist of three sets of parameters: configuration parameters for the AF1 queue, configuration parameters for the AF2 queue and configuration parameters for the BE queue. The AF1 queue is assumed to operate like an RIO queue as described in [CF98]. The BE queue is assumed to operate as an RED queue as described in [FJ93]. The parameters for configuring the queues are then

- $(\mathrm{th}_1^{\mathrm{min}_{\mathrm{out}}}, \mathrm{th}_1^{\mathrm{min}_{\mathrm{out}}}, \mathrm{linterm}_1^{\mathrm{out}}, \mathrm{th}_1^{\mathrm{min}_{\mathrm{in+out}}}, \mathrm{th}_1^{\mathrm{max}_{\mathrm{in+out}}}, \mathrm{linterm}_1^{\mathrm{in+out}}, p_1^{\mathrm{weight}})$: the AQM configuration parameters for the AF1 queue;
- $(\mathrm{th}_2^{\mathrm{min}_{\mathrm{out}}}, \mathrm{th}_2^{\mathrm{min}_{\mathrm{out}}}, \mathrm{linterm}_2^{\mathrm{out}}, \mathrm{th}_2^{\mathrm{min}_{\mathrm{in+out}}}, \mathrm{th}_2^{\mathrm{max}_{\mathrm{in+out}}}, \mathrm{linterm}_2^{\mathrm{in+out}}, p_2^{\mathrm{weight}})$: the AQM configuration parameters for the AF2 queue;
- $(\mathrm{th}^{\mathrm{min}}, \mathrm{th}^{\mathrm{max}}, \mathrm{linterm}, p^{\mathrm{weight}})$: the AQM configuration parameters for the BE queue.

The AQM parameters require some explanation. The RED parameters used to manage the BE queue are explained first since these are the simplest and illustrate how AQM works. The first two parameters are thresholds: the first parameter determines the lower threshold below which AQM has no effect, and the second parameter specifies the upper threshold above which all packets are dropped. If the (averaged) queue length falls between these two thresholds, when a packet arrives, the packet can be dropped with some probability. This probability is dependent on the queue length: if the queue length is high, then the drop probability is higher. The linterm defines the rate at which the drop probability increases with the queue length. Finally, the last parameter, p^{weight}, determines the time interval over which the queue length is averaged.

The AQM parameters for the AF queues are similar, but a little more complicated. In this case, there is a different drop probability for packets with different drop precedences. Since only two drop precedences are assumed in this network, the packets can be considered to be in-profile and out-of-profile as described in [CF98]. The AF queues essentially operate two different RED mechanisms; one which drops out packets and one which drops in packets. The AQM parameters are then analogous to the RED parameters, except that there are two sets of them for the AF queue. Note that in the RIO case, the same queue length measurement is used as the basis for both (separate) RED mechanisms.

In this problem, the objective is to route the demands on the network such that the load on the network is balanced. The problem cannot be formulated here in any more specific terms. The way that load balancing is effected and the cost function that must be optimised are introduced in the mapping function below.

The above specific formulation is perhaps not the most natural formulation for this problem, since the objective of the optimisation is specified in a vague manner. This is because it is constructed with the generic problem in mind. Hence, the specific problem models formulated in this framework will have a tendency to be somewhat biased towards the generic problem model. This is not surprising.

5.5 Mapping to the Generic Problem Model

Here, the problem of mapping from the specific problem discussed above to the generic problem described in chapter 3 is discussed. Two aspects of the mapping function are discussed here: mapping from the specific problem to the generic problem and mapping the solution of the generic problem to a solution to the specific problem.

The specific problem described above is not described in sufficient detail to be formulated. However, it must be formulated at the generic problem layer. Consequently, extra information is introduced in the mapping layer that enables the problem to be properly formulated. This extra information and the assumptions made in the mapping process are described here.

The mapping layer is a two-way process and the output of the generic problem must be used to generate results which are appropriate for the specific layer. The output of the generic problem is a routing for the demands. This is required in the solution to the specific problem. However, it is not sufficient: the queue/scheduler configuration parameters must also be obtained to solve the specific problem.

The two elements of the mapping function are described in more detail in the following sections.

5.5.1 Mapping from the Specific Problem to the Generic Problem

This component of the mapping function consists of three important elements:

- Mapping of the demands from the specific problem to the generic problem;
- Generating the cost function;
- Treatment of the BE traffic in the model;

The problem is solved by decoupling the routing problem from the queue/scheduler configuration problem. This is done by determining an effective bandwidth for each demand. The purpose of the effective bandwidth is to attempt to quantify the amount of resources required by a variable-rate demand on each link. This is particularly applicable to the AF demands. The effective bandwidth captures the demand characteristics and its QoS requirements: if each demand is allocated its effective bandwidth on each link in its path, the demand should obtain the desired QoS. As such, it enables the packet level issues to be decoupled from resource allocation and routing issues. Once this effective bandwidth is determined, efforts can be focussed on solving the routing problem.

The effective bandwidth notion can then be used to balance the load on the network. Instead of using some kind of packet based metric for load balancing, effective bandwidth based metrics can be used. For example, the objective can be to attempt to balance the aggregate effective bandwidths on the network. This can be achieved by choosing an appropriate cost function for the generic problem.

Finally, some preprocessing of the specific problem may be required to ensure that there are sufficient resources for the BE traffic in the generic problem. This is discussed further below.

Mapping the Demands from the Specific Problem to the Generic Problem

In the generic problem, all demands are characterised by a single parameter. In this specific problem, some of the demands are characterised by more than one parameter.

For these demands, it is necessary to map from the parameter set of the specific problem to a single parameter for the generic problem.

There are three different types of demands in the specific problem: EF demands, AF demands and BE demands. The EF demands are characterised by a single rate. This rate is used to characterise the equivalent demand in the generic problem – no transformation of this rate is necessary. If this rate is allocated to the demand, the traffic will obtain the desired QoS. The BE demands are not quantified in this problem; the customers use the capacity available to BE traffic at will and the operator does not make any assurances to the customer pertaining to the level of service offered by BE traffic.

The AF demands are a little different from the EF and BE demands and they require more sophisticated treatment. The AF demands are quantified using a peak rate, a mean rate, a mean on-time and a mean off-time. The problem of mapping these parameters to a single parameter, such that the desired QoS is delivered, is considered in detail in Appendix A, but some comments are given here on the overall approach that is used.

The simplest and most conservative approach to obtain a single parameter for the AF traffic for the generic problem is to use the peak rate of the demand. This will ensure that the users perceive a very high QoS. However, this approach has two serious drawbacks. Firstly, if this approach is used, there will be little difference between EF and AF service. Consequently, customers will opt for the (presumably) cheaper AF service. Secondly, it is grossly inefficient: resource allocation based on such a conservative approach results in severe resource underutilisation. Hence, a less conservative approach is desirable.

Another approach, which cannot be used to obtain an effective bandwidth for an individual SLA, but rather is a means to perform admission control is *Measurement Based Admission Control* (MBAC) (see [BJS00] for a review of MBAC). This is mentioned here because it is perceived by many to be a good approach to performing admission control for AF services. As such, it can be used in situations in which the effective bandwidth ideas are used and it is useful to identify why such an approach is not suitable here.

The principle on which MBAC is based is that the resource utilisation is measured and the measured utilisation is used in conjunction with the declared parameters of the traffic to make an admission control decision. It is not applicable in this situation for two reasons: firstly, it can only work in a dynamic situation, since it measures actual traffic usage and secondly it cannot make very good assurances since it cannot predict future traffic patterns. For these reasons, it is not considered further here.

Here, a parameter based approach is used to determine the effective bandwidth required for an AF demand, i.e. declared parameters of the demand are used to calculate the effective bandwidth. This is more sophisticated than the simplistic peak rate based approach.

The effective bandwidth notions were developed to solve the admission control problem for ATM [Hui88]. The admission control problem is to determine whether a new connection can have access to a resource such that the appropriate QoS is delivered to the new connection without compromising the QoS delivered to the connections already using the resource. This problem can be solved using effective bandwidths by associating an effective bandwidth with each connection. The effective bandwidth reflects how much of the resource is needed by the source to obtain the required QoS. The admission control decision is then simple: if the sum of the effective bandwidths – including that of the new source – exceeds the capacity of the resource, then the new source is blocked.

A similar idea can be applied in this diffserv context: the effective bandwidth required for each SLA can be calculated and can be used to reflect the amount of resources required for the SLA to ensure that the required QoS is delivered. In this case, it is not used to perform a dynamic admission control decision; rather it is used to determine how much capacity on a link is required by the AF traffic to ensure that the loss and delay targets are met. The effective bandwidth can then be used as the single parameter which characterises demands in the generic problem.

The specific approach to determining an effective bandwidth that is used here is described in Appendix A. A brief description of the approach is included here for completeness.

The approach is based on the work of Guerin [GAN91]. There, Guerin et al use an approach consisting of two components to obtain an effective bandwidth for a flow: the first approach is based on a buffered model and the second approach is based on a bufferless model. In both cases, the objective is to determine the effective bandwidth required to ensure loss is no more than some specified bound. In the first approach, a variation of the work of Anick, Mitra and Sondhi [AMS82] is used to determine the effective bandwidth. There, it is assumed that a number of homogeneous on-off sources are input to a buffer. The problem is to determine the required rate at which to serve the buffer such that the buffer loss is no more than some specified value. The effective bandwidth of each source is then this aggregate effective bandwidth divided by the number of flows. The second approach is based on a bufferless model. The idea here is to determine the stationary probability that a number of flows will be in the on state simultaneously. Thus the aggregate bandwidth generated by the sources can be determined. The effective bandwidth is chosen by choosing that aggregate bandwidth for which the probability of exceeding it is less than the loss probability.

Using the above two approaches, then some effective bandwidth for the set of flows can be determined. The effective bandwidth that is chosen then, is the minimum of the two effective bandwidths as calculated above.

The effective bandwidth as determined above can be used to ensure that the loss experienced at the buffer does not exceed some pre-specified limit. Here, the objective is also to ensure that there is a bound on the delays experienced at each buffer. This is achieved by permitting the buffer size to be a parameter. A simple relation relating the maximum buffer delay, the effective bandwidth and the buffer size is introduced. The bandwidth and buffer size required to deliver the desired per-queue loss and delay can be obtained iteratively determining the required bandwidth for a given buffer size.

The Cost Function

In this problem, the objective is to determine a route configuration for the demands that results in a balanced network load. No objective function was defined for the specific problem: rather the objective was defined in higher level terms. A concrete cost function is necessary to solve the generic problem. Consequently, a cost function must be introduced in the mapping function so that the generic problem can be solved.

Any cost function can be chosen which, when used in the generic problem, results in the load being balanced on the network. The cost function does not necessarily have to reflect real costs associated with service delivery in the specific problem. Consequently, a quite artificial cost function may be chosen.

The generic problem does not permit complete flexibility when choosing a cost function. In the generic problem, the overall cost is the sum of a set of link costs and node costs. The link costs and node costs are dependent on the amount of demand carried on the link and the amount of demand switched through the node, respectively.

This obviates a large set of cost functions, but the set of cost functions that can be characterised in this manner is sufficiently broad and flexible to be useful.

In the specific problem, it is not immediately obvious how the load on the network can be balanced. This is especially true, given that it may be difficult to determine whether or not the load on the network is balanced without the use of measurement. Hence, choosing a cost function that balances the load on the network is a little difficult.

The effective bandwidth concept provides a solution to this. The link and node costs can then be written as functions of the aggregate effective bandwidth carried on the link and switched through the node respectively.

The load is then balanced on the network if the effective bandwidths on the links are balanced, i.e. if the difference between the aggregate effective bandwidth of the traffic carried on each link is minimised. Using this approach, it is possible to show that the optimisation can be useful in balancing the load on the network.

In the generic problem, the cost function can have a link and a nodal cost component. Here, the nodal cost component is not used: the cost function is comprised solely of link cost functions. However, the specific problem could easily be extended to include a term which can be used to balance the load through switches. Having said this, balancing the load on the links should balance the load on the nodes to some extent.

The purpose of load balancing is to ensure that some links are not heavily loaded while others are lightly loaded. The cost function chosen for this problem should have a high cost associated with high link utilisation. Conversely, if the link is lightly utilised, then the link cost should be low. Hence, an increasing function is required.

A linear function could be used, but the rate at which it increases is not fast enough as the link nears full utilisation. For example, if a link is highly utilised, then some of the traffic should be rerouted on an alternative path. In a system in which the link cost is linear in demand, rerouting the demand will probably incur a greater cost (according to a linear cost function), especially if the demand is routed on a longer path. Consequently, the demand would not be rerouted using such a cost function.

Fortz and Thorup [FT00] consider quite a similar problem and they use a piecewise linear approximation to an exponential as their link cost function. This has the effect of balancing the load in the problems they consider.

Here, an exponential link cost function is used. The cost function is a function of the difference between the amount of capacity carried on the link and the link capacity. This is necessary to incorporate the capacity of the link into the expression. Hence, the link cost function can be written as:

$$f_l(s_l^{\text{up}}, s_l^{\text{down}}) = e^{j(s_l^{\text{up}} - c_l)} + e^{j(s_l^{\text{down}} - c_l)}$$

where s_l^{uo} and s_l^{down} are the upstream and downstream capacity respectively and j is a parameter that can be used to control the resulting costs. s_l^{up} and s_l^{down} are sums of the effective bandwidths of the demands using the link. j must be greater than 0.

This cost function results in a value less than 1 if the used capacity is less than the link capacity and results in a value of more than 1 if the used capacity exceeds the link capacity.

If all of the links in the problem have the same capacity, then the link capacity term can be removed since it simply acts as constant multiplier over all of the terms in the cost function. However, this is not the case if the link capacities differ: the link capacity term is essential in this case to effect load balancing.

The scaling parameter in the exponential was introduced to ensure that there are not enormous differences between the orders of magnitude of the terms being added to obtain the overall network cost function. For example, if the capacities are measured in Mb/s and the link capacity is 155Mb/s and the used capacity is 100Mb/s, without the scaling parameter, the cost of this link is $e^{-55} = 1.30 \times 10^{-24}$. If then, another link has only 50Mb/s utilisation, then the resulting link cost is $e^{-105} = 2.5 \times 10^{-46}$. It is difficult to perform calculations with these two numbers due to their enormous difference. Hence, scaling can be added to reduce the differences. However, if the scaling parameter reduces the difference too much, there may be little difference between link cost functions, even though the difference in spare capacity on the links may be substantial.

Note that this cost function could not have been constructed for the specific problem since the effective bandwidth concept was not well developed there. Hence, it is included in the mapping function.

Catering for BE Traffic

It is quite straightforward to cater for BE traffic in this problem. The amount of BE traffic entering the network is unpredictable. A reasonable approach to cater for the BE traffic which will ensure some level of service for the BE traffic is to reserve some of the resources on each link for it. This can be done on an absolute or relative basis. In the former case some specified amount of capacity must be reserved for the BE traffic at each link, while in the latter, some fraction of each link is reserved for the BE traffic. These parameters could vary across the links.

In either case, the capacity made available to the BE traffic is removed from each link at this stage, the problem solved as before and the BE traffic is reincorporated when mapping from the generic problem back to the specific problem. At this stage the core link capacities are reduced by the amount of capacity to be reserved for the BE traffic on each link. In mapping from the generic problem to the specific problem, this capacity is again introduced through appropriate choice of the scheduler weights.

The core link capacities that can be used in the generic problem are then $y_l c_l$.

Formal Mapping from the Specific Problem to the Generic Problem

The content of the three sections above can now be combined to perform the mapping from the specific problem to the generic problem. Here, this mapping is formalised.

The set of generic demands are constructed using the EF and AF demands in the specific problem. The generic demands are characterised by a source node, destination node and a single parameter characterising the demand. The EF demands in the specific problem are characterised in a similar manner. Consequently, the EF demands can be mapped directly to a set of parameters that are can be used in the generic problem. The AF demands are characterised by a source node, destination node and a more complex set of parameters to describe the traffic. These are mapped to the generic demands using the effective bandwidth approach described above.

Thus, the generic demands can be generated as follows.

First, the set of demands in the generic problem that correspond to the EF demands in the specific problem can be written as:

$$D_{\mathrm{EF}} = E .$$

AF demand a in the specific problem can be characterised using the parameters $\{o_a, p_a, r_a^{\text{peak}}, r_a^{\text{mean}}, t_a^{\text{on}}, t_a^{\text{off}}, s_a\}$. Then the equivalent demand for the generic problem can be written as

$$a' = \{o_a, p_a, h_a\}$$

where $h_a = \text{EB}(r_a^{\text{peak}}, r_a^{\text{mean}}, t_a^{\text{on}}, t_a^{\text{off}}, s_a)$ is the effective bandwidth of the AF demand. These demands can then be used in the generic problem. For convenience, the following sets are defined:

$$D_{\text{AF1}} = \bigcup_{a|s_a=1} a'$$
$$D_{\text{AF2}} = \bigcup_{a|s_a=2} a'$$

These are the two sets of demands in the generic problem that correspond to the AF1 and AF2 demands in the specific problem, respectively. The set of demands then used as input to the generic problem is

$$D = D_{\text{EF}} \bigcup D_{\text{AF1}} \bigcup D_{\text{AF2}}\ .$$

The link cost function can be written as

$$f_l(s_l^{\text{up}}, s_l^{\text{down}}) = e^{j(s_l^{\text{up}} - y_l c_l)} + e^{j(s_l^{\text{down}} - y_l c_l)}$$

The specific problem can then be mapped to the generic problem as follows:

- $N \rightarrow \mathrm{N}$;
- $L \rightarrow \Lambda$;
- $G(N, L) \rightarrow \Gamma(\mathrm{N}, \Lambda)$;
- $D \rightarrow \Delta$;
- $f_l(s,t) \rightarrow \phi_\lambda(s,t)$ and
- $\gamma_v(\cdot) = 0$ for all $v \in \mathrm{N}$

5.5.2 Mapping from Generic Problem Solution to the Specific Problem Solution

The solution to the generic problem consists of a route configuration. The solution to the specific problem consists of a route configuration as well as configuration parameters for the queue/scheduler systems controlling access to the links. The solution to the generic network design problem – the route configuration – is part of the solution to the specific problem. The queue/scheduler configuration parameters must also be chosen to obtain a full solution to the specific problem. Here, the choice of these parameters is discussed.

Choosing the Scheduler Configuration Parameters

As noted above, the schedulers in this problem are WRR schedulers. The specific implementation of the schedulers requires a number of parameters to be specified. Not all of these parameters are discussed here: only the scheduler weights are considered. These determine how much of the capacity of the resource – the link – is allocated to each queue.

The amount of resources required on each link for each traffic class can be determined by adding the effective bandwidths of the demands of each traffic class which are carried on the link. For each link, then the amount of resources required for the EF traffic on the link can be calculated as

$$w_{\mathrm{EF}}^{l_{\mathrm{up}}} = \sum_{e|I_e^l(R)=1} r_e^{\mathrm{peak}}$$

where $I_e^l(R)$ is an indicator function, which returns a value of 1 if demand $e \in E$ is routed on link l in the upstream direction under route configuration R; it returns a value of –1 if the demand is routed on link l in the downstream direction and a value of 0 if the demand is not routed on link l. Similarly, for the downstream traffic,

$$w_{\mathrm{EF}}^{l_{\mathrm{down}}} = \sum_{e|I_e^l(R)=-1} r_e^{\mathrm{peak}}$$

The resources required for the different AF traffic classes can be determined in a similar manner, except that effective bandwidths should be used, rather than the peak bandwidth as is used in the EF case:

$$w_{\mathrm{AF}i}^{l_{\mathrm{up}}} = \sum_{a|s_a=i,\, I_a^l(R)=1} h_a$$

The capacity assigned to the BE traffic is the traffic remaining after the resources have been allocated to the higher priority traffic classes. The scheduler weights can then be written as $w_{\mathrm{EF}}^{l_{\mathrm{up}}} : w_{\mathrm{AF1}}^{l_{\mathrm{up}}} : w_{\mathrm{AF2}}^{l_{\mathrm{up}}} : c_l - \left(w_{\mathrm{EF}}^{l_{\mathrm{up}}} + w_{\mathrm{AF1}}^{l_{\mathrm{up}}} + w_{\mathrm{AF2}}^{l_{\mathrm{up}}}\right)$; the weights are determined here for the traffic in the upstream direction.

The weights can be normalised so that they add to 1, although this is not so important.

Note that the resources reserved for the BE traffic do not explicitly appear here. These resources impose a limit on the sum of the resources available to the higher priority classes and thus ensure that $c_l - \left(w_{\text{EF}}^{l_{\text{up}}} + w_{\text{AF1}}^{l_{\text{up}}} + w_{\text{AF2}}^{l_{\text{up}}}\right)$ is large enough.

Choosing the Queue Configuration Parameters

Some of the queue configuration parameters are a little more complex to obtain. Guidelines suggested in previous research work are used to choose these parameters, rather than rigorous modelling and analysis. The queue parameters that need to be determined are the queue lengths and the AQM configuration parameters. Each of these is discussed separately.

For the AF traffic, the queue length parameters are partially calculated when choosing the effective bandwidths: a component of the effective bandwidth calculation described above is the calculation of a queue size. This is the amount of queue capacity required per demand. The queue sizes can be determined by adding these individual queue contributions. These are added in the same way as the effective bandwidth parameters are added to obtain the required queue length at each interface, i.e.

$$q_{\text{AF}i}^{l_{\text{up}}} = \sum_{a|s_a=i,I_a^l(R)=1} q_a$$

Another approach is required to determine an appropriate queue length for the BE and EF traffic. Since delays are critical for EF traffic, the queue lengths for the EF traffic are bounded by delay concerns. Hence, the queue lengths should be calculated based on the maximum delay that can possibly accumulate in a particular buffer. The relation

$$q_{\text{EF}} = \frac{d_{\text{EF}} * w_{\text{EF}}}{m_{\max}}$$

can be used to determine the queue length in bits. This can then be converted to bytes or packets if some mean packet size is assumed. Since nothing is known about the BE traffic, no knowledge can be assumed when calculating the queue length for the traffic. Here, the same method as that used for the EF traffic is used – a maximum delay parameter for the BE traffic is determined and the queue size can be calculated by multiplying the delay by the bitrate for the BE traffic. Since delays are not a big issue for BE traffic, the delays experienced by the BE traffic at queues can be made large and the resulting queue sizes can also be large. Alternatively, all of the buffer space not allocated to the higher priority traffic at an interface can be allocated to the BE traffic.

The AQM parameters are also chosen using some guidelines. AQM parameters must be chosen for the AF and BE queues; AQM parameters do not need to be chosen for the EF queue. The AQM parameters for the BE traffic are discussed first. For RED, four parameters must be specified:

1. The minimum threshold: if the average queue length is below this, no random dropping will occur;
2. The maximum threshold: if the average queue length exceeds this, then all packets will be dropped until the queue length drops below this threshold;
3. The queue weight parameter: a parameter that determines how the average queue length is measured;
4. A linear term used to control the fraction of packets dropped when the average queue occupancy is between the minimum and maximum thresholds.

In [FJ93], Floyd and Jacobson suggest that the upper threshold should be at least twice the lower threshold. The maximum threshold is assumed to be exactly double the minimum threshold here. To ensure the utilisation of the BE traffic resources remains high, the maximum threshold should be high. Here, the maximum threshold chosen is 90% of the queue length. A queue weight parameter of 0.002 is considered reasonable by Floyd and Jacobson: this can be used here. This choice of parameter results in an average queue length measure which reacts sufficiently fast to offset congestion, but still filters out very transient effects. Floyd and Jacobson also propose a small packet drop probability, claiming that a large packet drop probability will cause oscillatory effects. The linear drop probability term is 1/50, which results in a maximum drop probability of 4%.

The AQM parameters for the AF traffic can be chosen using similar guidelines. The work of Kim et al [KLT98], which is influenced by the earlier work of Floyd and Jacobson discusses how parameters for the RIO AQM scheme can be chosen. There, they suggest that the maximum threshold is double the minimum threshold for the out traffic; they also suggest that the maximum threshold for the in traffic is equal to the minimum threshold for the out traffic. The minimum threshold for the in traffic is half of the maximum threshold for the in traffic. By choosing the maximum threshold for the out traffic as 90% of the queue length and using the relations above, all the threshold values are determined. The queue weight parameter is again set to 0.002 and the linear drop probability terms are set to 1/50.

5.6 Example Problems

Some example problems will illustrate the use of this approach to determine a good network configuration. The purpose of these examples is to highlight two specific points: firstly, the method is validated using some small problems and secondly, the utility of performing the optimisation is discussed, particularly for larger problems.

The method is validated by generating some small problems, solving the problems and simulating the solution obtained. The output of the simulator can be used to determine whether the solution delivers the appropriate QoS; if not, then the solution is not valid. It is necessary to use small problems to validate the approach since larger problems take very long times to simulate. The disadvantage of using small problems is that the benefits of optimising the network configuration are not so great: these benefits become more apparent when the problem is larger. Consequently, some examples of the use of the optimisation in the context of larger problems are given.

5.6.1 Problem Generation

A common problem generator was used to generate the smaller problems to be used for validation and the larger problems that illustrate the benefits of this approach.

Each problem consists of a set of nodes, links, demands and desired network operating conditions. The nodes are identified by a node name: they have no interesting properties. Choosing the characteristics of the remaining elements of the problem was a little more difficult and the method used to choose these is discussed in the following paragraphs.

The core network links are chosen at random subject to connectivity and node degree constraints. The connectivity constraint stipulates that the core network be connected and the node degree constraint stipulates that each node has, on average, a number of connections to the other nodes in the network. The node degree constraint is easiest to accommodate. Since the average node degree is defined as

$$\deg_{\text{avg}} = \frac{2|L|}{|N|}$$

where $|L|$ is the number of links in the network and $|N|$ is the number of nodes, the constraint can be met by choosing $|L| \geq \deg_{\text{avg}}^{\text{req}} |N|/2$.

The connectivity constraint is slightly more difficult to satisfy. To ensure that this is satisfied, connectivity tests must be performed. Such tests determine whether all nodes are connected, i.e. whether it is possible to move from any node to any other node by traversing the links. A depth first search algorithm which is attributed to Trémaux is described in [GM84]; it can be used to determine whether a graph is connected.

This test can be used in an iterative procedure to ensure that the resulting network is fully connected. The approach is straightforward: if the network is not fully connected, then a link is added between two unconnected parts of the network. A link is removed at random. This procedure results in a connected network after a number of iterations.

The links are assumed to be bidirectional links and some capacity is assigned to the links when the problem is being generated. Each link also has an associated propagation delay which is specified by the user of the problem generation tool.

The demands are considered next. The demands are generated by a number of customers. It is assumed that each customer has a premises homed on one of the core network nodes. Each customer premises has communications requirements with the other premises which are part of that customer's internal network; each customer wants to implement a fully connected network between its premises. Furthermore, each customer has demands between the nodes which require the use of different service classes.

In the problem considered here, four traffic types are permitted into the network: EF, AF1x, AF2x and BE. The EF traffic is very high priority traffic that has strict delay and loss requirements of the network. The AF traffic also has delay and loss requirements, although they are not as strict as those required for the EF traffic. The BE uses the resources remaining after the higher priority traffic has been catered for.

Each customer has a requirement for EF, AF1, AF2 and BE service between all of the premises. This is probably not a very likely scenario, but serves to illustrate that the approach works. However, this scenario may model a different and more realistic scenario reasonably well. In this second scenario, there are a larger number of customers, each of which only use a single service class for their internodal demands. The scenario described here could model this more realistic situation.

Only the priority demands are quantified – the customer is permitted to inject as much BE traffic as desired into the network but cannot expect any QoS assurances for this traffic type. In contrast, the other traffic types must be characterised and hence some QoS assurances can be given.

Since the EF traffic has very strict delay and loss requirements associated with it, it is characterised by a single peak rate. This is policed at the access to the network and this is the amount of resources that are reserved for each EF demand on its path through the network. When generating random EF demands, only this parameter needs to be chosen.

The AF traffic does not have such stringent requirements of the network. Also, the AF traffic is more likely to be more variable than the EF traffic. Consequently, a less conservative approach to resource reservation can be used. It is assumed that the AF traffic can be characterised by two rates: a peak rate and a mean rate. There is also a parameter that defines the maximum duration of a burst at the peak rate without being adversely affected. These parameters are then used to determine the effective bandwidth and buffer size required by the source as described above.

Finally, the network-wide loss and delay parameters are chosen for the different service classes which have associated QoS parameters. These only apply to the EF, AF11 and AF12 traffic types: specifically, they do not apply to lower priority AF traffic.

5.6.2 Validation of the Approach

The approach is validated by generating, solving and then simulating some small problems. While the problems are small, the amount of effort required to simulate them is not insignificant: there are very many parameters that need to be specified in order to run the simulation. Also, the amount of time required to simulate even small scenarios is substantial.

Simulation of the system was done using the ns IP-level simulator [FV01]. The simulator did not have diffserv and MPLS support when it was investigated first; diffserv support was added as part of this work and MPLS for ns was developed by Ahn [AC00]. The diffserv functionality then worked in conjunction with the MPLS functionality to enable demands to be routed arbitrarily and to offer some service differentiation to the demands. More details on the simulator are given below.

Small Example Problems – Generation and Solution

Two small example problems were generated to verify the concept using the simulator. The first problem had 7 nodes and 5 customers. It had an average node degree of 2.4, which resulted in 9 links. This was problem 5A. The second problem was slightly larger with 9 nodes and 4 customers. This problem also had an average node degree of 2.4, which results in a core network consisting of 11 links in this case. This was problem 5B. The parameters used to generate the first and second problems are listed in Table 5-1 and Table 5-2 respectively. Using these parameters, two specific problems were generated.

Property	Value	
Number of Nodes	7	
Core Link Capacities	155	
Average node degree	2.4	
Number of Customers	5	
Parameter for EF traffic	min: 0.5	max: 1
Parameter for AF1 Mean traffic	min: 0.5	max: 1
Parameter for AF1 Peak/mean	min: 2	max: 2.5
Parameter for AF2 Mean traffic	min: 0.75	max: 2
Parameter for AF2 peak/mean	min: 2.5	max: 4
Token bucket size for peak	2000	
Source on time	min: 0.1	max: 0.15
EF end2end delay bound	10	
EF end2end loss bound	2	
AF1 end2end delay bound	75	
AF1 end2end loss target	2.5	
AF2 end2end delay bound	100	
AF2 end2end loss target	4	
Random Seed	9485	
Output	Dsproblem5b.test	
Max route length	5	

Table 5-1: Parameters used to generate problem 5A.

The problems were then converted to generic problems. In doing this, it was necessary to choose the parameter j which dampened the exponent in the exponential cost function. For the purposes of these experiments, this was chosen to be 0.05.

The generic problems were then specified. These were solved using the solver based on the greedy algorithm since this obtained the best solutions in chapter 4. Since these problems were very small, the solutions were found in a matter of seconds.

Parameter	Value	
Number of Nodes	9	
Core Link Capacities (Mb/s)	155	
Average node degree	2.4	
Number of Customers	6	
Parameter for EF traffic (Mb/s)	min: 0.5	max: 1
Parameter for AF1 Mean traffic (Mb/s)	min: 0.5	max: 1
Parameter for AF1 Peak/mean	min: 2	max: 3.5
Parameter for AF2 Mean traffic (Mb/s)	min: 0.75	max: 2
Parameter for AF2 peak/mean	min: 2.5	max: 4
Token bucket size for peak (bytes)	2000	
Source on time (s)	min: 0.1	max: 0.15
EF end2end delay bound (ms)	10	
EF end2end loss bound (%)	1	
AF1 end2end delay bound (ms)	75	
AF1 end2end loss target (%)	2.5	
AF2 end2end delay bound (ms)	100	
AF2 end2end loss target (%)	4	
Seed	34867	
Output	Dsproblem8.test	
Maximum route length	5	

Table 5-2: Parameters used to generate problem 5B.

Simulation Issues

The solution to the generic problem coupled with the solution mapping described above consists of enough information to enable the core network to be configured. To simulate a scenario, more assumptions need to be made and more parameters need to be specified: for example, the source traffic characteristics and the access link properties need to be specified. These are defined here.

For the purposes of the simulation, a particular configuration for the customer premises network topology and the access link topology is assumed. This is illustrated in Figure 5-6. In this topology, each customer premises is assumed to consist of 4 traffic-generating nodes. These are connected via an access multiplexer to the core node. A number of customer premises can be connected to the core node, each one connected via a different access multiplexer.

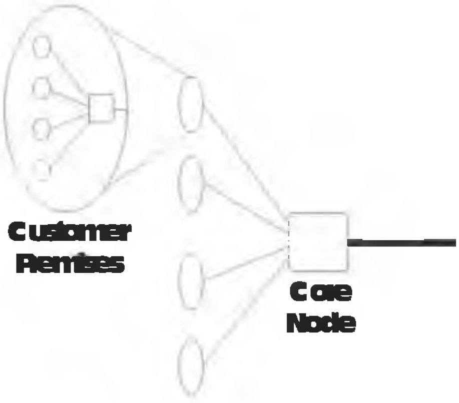


Figure 5-6: Customer premises and access network topology used in simulation.

The 4 traffic generating nodes each generate traffic of a different service class: node 0 generates EF traffic, node 1 generates AF1 traffic, node 2 generates AF2 traffic and node 3 generates BE traffic. The EF source generates traffic at a constant rate with some random variation. The AF sources generate on-off traffic – the source transmits at a peak rate while on and no data is generated when the source is in the off state. This is oft-considered to be a kind of worst case traffic, and if the network can accommodate this, it can accommodate less bursty traffic. The BE sources are used to add load to the network; these simply generate CBR traffic, again with a random perturbation.

The AF service class is most suited to a feedback-based transport protocol such as TCP. In the most likely configuration of AF, most traffic will be permitted into the network, although some traffic will be marked down to a lower priority if it is non-conformant with the agreed traffic profile. AF queues will have AQM such as some variant of RIO [CF98]. If the network is becoming congested, then the AQM mechanisms will cause some packets to be lost. Adaptive transport protocols will react to this packet loss and reduce their transmission rate reducing the likelihood of congestion. Non-adaptive traffic sources can obtain a disproportionate amount of the resources at the expense of the adaptive traffic. For these reasons, AF is more suited to adaptive traffic.

TCP flows, however, are not used in this simulation. The reasons for this are twofold: firstly, TCP sources are much more complex and would greatly increase the simulation time and memory requirements and, secondly, the traffic flow between node pairs is more likely to be an **aggregate** of TCP sources rather than a single flow. TCP sources maintain a number of counters and timers as well as functionality to modify behaviour to network conditions – they are much more complex than the non-adaptive sources described above. Consequently, it is not feasible to run such a large simulation with TCP sources with the available computing power. Furthermore, the scenario described above is one in which the demands are aggregate demands: hence the connection between any two nodes will carry a substantial amount of TCP flows. The behaviour of a TCP aggregate is quite different to that of a single TCP flow. To simulate the system using TCP sources then, a number of sources would be required between each node pair. This is not feasible with the available computing power.

A further reason that TCP can be ignored is that this work takes the view of an operator delivering services with SLAs. The SLAs are measured at the IP layer. If worst case traffic at the IP layer is assumed, then the operator can be confident that the assurances offered can be met if the traffic is not worst case. If the vast majority of the traffic is TCP, which results in a much lighter network usage than the worst case IP traffic, then the network will be overengineered. Overengineering is a secondary issue: the primary objective here is to offer service with assurances. Once this is available, the network efficiency can be improved with operational experience.

The access links must be parameterised. Since the emphasis here is on the behaviour of the core network, it is assumed that the access links are high capacity, low delay access links. As such, they do not have a large impact on the perceived QoS. For these experiments, the access links are were chosen to have a capacity of 75Mb/s and a propagation delay of 0.1ms.

The core network propagation delays are also small: the network is assumed to be a single autonomous system, which typically covers a smaller area, rather than a large nation-wide network. Having said this, it is conceivable that an autonomous system could have a very great geographical spread. For these experiments, the core network link propagation delays are chosen to be 0.1ms.

Packet sizes must also be assumed in the simulation. The EF packet sizes are assumed to be small, since the traffic is assumed to be delay sensitive. The traffic could, for example, be VoIP traffic. The EF packet sizes are assumed to be just 100 bytes[17]. Packet sizes for the other classes are assumed to be 500 bytes. Packets which are approximately 500 bytes in size are very common on the Internet because many transmission facilities have an MSS of approximately 500 bytes [TMW97].

The simulator used to perform the simulations contains MPLS and diffserv implementations. In the MPLS implementation, default paths through the network are generated based on OSPF routing as happens in real MPLS implementations. In this problem, the purpose of MPLS is to facilitate arbitrary routing of demands. MPLS explicit routing is used to enable arbitrary paths to be configured through the network. This is used in conjunction with a filter at the edge of the network to ensure that only some specific set of packets are permitted to use an LSP.

The MPLS implementation does not need to be parameterised, other than to initiate MPLS explicit route messages to enable the MPLS LSPs to be formed.

The diffserv implementation is as described in section 5.2.1 above. It requires a number of parameters to be configured properly – the scheduler weights, queue sizes and AQM parameters. These can all be determined using the approach described above to map from the solution of the generic problem to the solution of the specific problem.

The simulations were run for 100 seconds of simulation time. This was not a very long time, but it was sufficiently long to obtain some reasonable information on the scenario. The core network links were operating at a speed of 155Mb/s and in many cases were very highly loaded. Assuming a 500-byte packet size, 100 seconds of simulation time corresponded to approximately 4,000,000 packets being processed on each of the core network links. It was felt that this was sufficient to enable reasonable data for core network performance to be concluded from the simulation. The 100 second simulation took a number of hours on a single processor Pentium III system operating at 600MHz. The simulation was not made longer because it already consumed a considerable amount of simulation time.

[17] VoIP packets are typically smaller than this – VoIP packets are often 40 bytes long. However, 100 byte packets will experience longer delays than 40 byte packets.

A number of sets of simulation results were obtained: simulation results were obtained when the network was stimulated with the design traffic – these were used as reference experiments – and when various increased traffic loads were used. In all cases, the network was quite heavily loaded: experiments in which the network is less heavily loaded are less interesting since all the users will simply obtain good service.

In each case, the packet loss and delay as perceived by the users were the primary measures of interest. Packet delays were obtained quite easily: each packet was marked with some packet generation time and the time taken for the packet to traverse the network was logged. End-to-end packet loss rates, however, were slightly more complicated to obtain so packet loss at queues in the network were obtained.

Simulating Problems with Design Traffic

The two problems were first simulated with the design traffic as input. The results from problem 5A are discussed first, followed by the results obtained from simulating problem 5B.

Link	Used Capacity (Mb/s)	Utilisation	Link	Used Capacity (Mb/s)	Utilisation
(0→2)	155000.0	1.00	(1→5)	41178.0	0.266
(2→0)	154999.9	1.00	(5→1)	46164.8	0.298
(0→3)	155000.0	1.00	(3→6)	119766.2	0.773
(3→0)	155000.0	1.00	(6→3)	145739.5	0.940
(0→5)	136159.5	0.878	(4→5)	139368.4	0.899
(5→0)	155000.0	1.00	(5→4)	130365.0	0.841
(1→3)	154999.9	1.00	(5→6)	81989.9	0.529
(3→1)	148106.8	0.956	(6→5)	82375.8	0.531
(1→4)	68189.6	0.440			
(4→1)	70337.0	0.454			

Table 5-3: Link utilisation for problem 5A with design traffic.

Problem 5A was simulated. The core network was quite heavily loaded. The core network link utilisations – utilisations in both directions – are given in Table 5-3. The mean link utilisation is 76.69%. Six of the links are fully loaded, and some loss is experienced at each of these links. No packet loss occurs at the queue/scheduler systems controlling access to the other links.

The loss occurring at the congested queue/scheduler systems is shown in Table 5-4. The high priority classes do not experience any loss, while the BE traffic can experience over 46% loss in the case of the link going from node 3 to node 0. Clearly, the BE traffic does not have a very great impact on the other classes. Clearly then, there is service differentiation between the BE traffic class and the higher priority traffic classes.

The end-to-end delays are also of interest. In this experiment, a number of measuring components were created, each at the ingress to each customer location. This could measure the end-to-end delay of all packets arriving at this component. Note that they did not measure the delays experienced by individual flows; rather they measured the delays experienced by all traffic of a particular class arriving at the component. These measurements were then used to obtain some measure of the minimum, maximum and mean packet delays for all end-to-end flows. For each measuring component, the mean delay, and the minimum and maximum delays were recorded. These were then averaged over all of the components to obtain network-wide averages. These results are presented in Table 5-5.

Link	EF loss	AF1x loss	AF2x loss	BE loss
(0→2)	0	0	0	0.172704
(0→3)	0	0	0	0.341545
(1→3)	0	0	0	0.173474
(2→0)	0	0	0	0.409773
(3→0)	0	0	0	0.464854
(5→0)	0	0	0	0.062792

Table 5-4: Loss at queue/scheduler systems for specific links in problem 5A with design traffic.

The results show that the loss and delay requirements are easily met in this small problem. The EF and AF traffic experience small delay. The mean delay for the EF traffic is smaller than that of the AF1 traffic, which, in turn is smaller than that of the AF2 traffic. The mean delays for all three are less than 1ms. This is very substantially lower than the required delay of 10ms for the EF traffic and 75 and 100ms for the AF1 and AF2 traffics respectively. The minimum delay for the EF traffic is smaller than that of the AF traffic because the transmission delay is smaller.

	EF	AF1	AF2	BE
Minimum Delay (s)	0.000434	0.000572	0.000572	0.000576
Maximum Delay (s)	0.001768	0.001327	0.003352	0.107321
Mean Delay (s)	0.000671	0.000741	0.00075	0.029302

Table 5-5: Delays averaged over all the node pairs for different traffic classes for problem 5A with the design traffic.

Interestingly, the averaged maximum delay for the EF traffic is larger than that of the AF traffic in this experiment. This can be explained by noting that the load generated by the EF traffic is approximately equal to the amount of resources allocated to it: the EF component of the system is constantly moderately loaded. In contrast, the AF traffic is generated in a bursty manner. Two factors lead to reduced delays for the AF traffic: the design of the scheduler and the somewhat conservative resource allocation. If an AF queue is underutilised for some short period, then 'credit' for the queue will build up. If a burst of AF traffic arrives in the queue then, the scheduler will have sufficient credit available to the queue to serve the burst quickly. This coupled with the fact that the conservative resource allocation mechanisms reduce the likelihood of data accumulating in AF queues means that the maximum delay experienced by the AF traffic can be low.

The averaged maximum delay for the AF2 traffic is over double that of the AF1 traffic. However, the delays are still very small – of the order of a few milliseconds. The QoS delivered by AF2 then is slightly worse than that delivered by AF1 services. This is because AF1 services have higher effective bandwidths than comparable AF2 services.

The delays experienced by the BE traffic are much larger. The averaged maximum delay in the system is over 100ms. The maximum over all node pairs is over 150ms. The mean delay is just over 29ms. Clearly, the range of the delays for the BE traffic in this scenario is large.

A histogram of end-to-end delays experienced by BE traffic arriving at one measuring component is shown in Figure 5-7. In this figure, there are two large peaks – the first occurring at 23ms and the second occurring at 60ms. These correspond to traffic carried on paths with different numbers of nodes. Traffic queued only at a single node contributes to the first large peak, while traffic which is carried over more than one node contributes to the second. There is a third peak which occurs due to traffic which traverses only lightly loaded links and experiences essentially no queuing. However, the number of such packets in Figure 5-7 is very small. From the graph it is clear that the number of congested links on the path has a strong bearing on the delay.

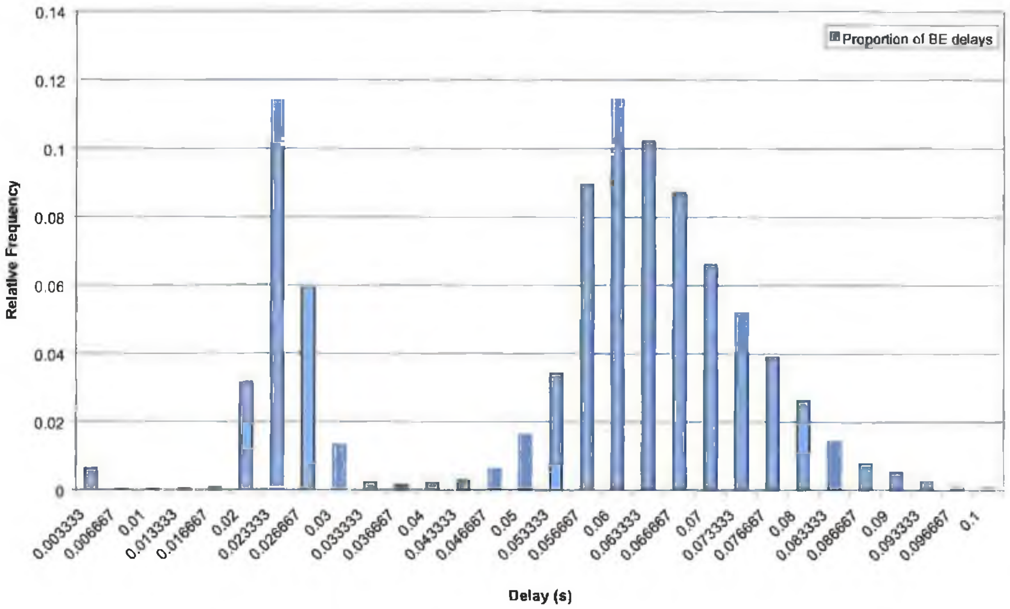


Figure 5-7: Distribution of BE delays for traffic arriving at a single customer premises in problem 5A with design traffic. Note that a significant proportion of the delays exceed 100ms.

In this experiment the EF and AF conditioners have a very small impact. The EF conditioners cause almost 1% of the traffic to be dropped. This is because the EF source is configured to generate packets in a CBR fashion with some random component added. The random aspect to the traffic can cause the source to generate some packets that are close together, although this is quite rare. When this happens, the second of the two packets may be dropped. The AF conditioner causes a very tiny amount of packets to be dropped: the proportion of dropped packets is of the order of 0.002%. This occurs because many AF sources run on a single host. It can be explained as follows. If many of the sources are in the 'on' state, then the aggregate rate can exceed the link rate. Buffering occurs. Then some of the sources generate traffic that gets buffered. This traffic may actually be transmitted on the link at a rate higher than that at which it was generated. In this case, packets could arrive at the conditioner with rate higher than the peak rate, resulting in packet drops for the AF traffic.

Link	Used Capacity (Mb/s)	Utilisation	Link	Used Capacity (Mb/s)	Utilisation
(0→3)	155528.3	0.888733	(3→7)	175000	1
(3→0)	106292.9	0.607388	(7→3)	175000	1
(1→2)	148496.1	0.848549	(4→5)	43670.01	0.249543
(2→1)	131326.8	0.750439	(5→4)	45303.98	0.25888
(1→5)	44078.15	0.251875	(5→6)	114150	0.652286
(5→1)	44046.14	0.251692	(6→5)	102966.1	0.588378
(1→6)	40791.31	0.233093	(6→7)	154309.4	0.881768
(6→1)	39472.68	0.225558	(7→6)	128167.3	0.732385
(2→4)	91653.93	0.523737	(7→8)	127283.2	0.727333
(4→2)	110332	0.630469	(8→7)	156736.8	0.895639
(2→7)	175000	1			
(7→2)	175000	1			

Table 5-6: Link utilisation for problem 5B with design traffic.

Problem 5B was then simulated. In essence, the same observations were made.

Link	EF loss	AF1x loss	AF2x loss	BE loss
(2→7)	0	0	0	0.330095
(3→7)	0	0	0	0.491702
(7→2)	0	0	0	0.131651
(7→3)	0	0	0	0.403198

Table 5-7: Loss at selected queue/schedulers for problem 5B with design traffic.

The capacity utilisations are listed in Table 5-6. Many of the links are quite heavily loaded, although some of them are lightly loaded. The mean link utilisation is 64.54%. Four of the links are used to capacity. As in the previous case, no drops are experienced at the links which do not have 100% utilisation. At the remaining links, the BE traffic suffers, while the other traffic types experience no loss. The loss experienced at each of these queue/scheduler systems is given in Table 5-7.

As with the previous experiment, the BE traffic can experience considerable loss – 49% loss in the case of the link from node 3 to node 7 – while the higher priority traffic experiences no loss.

	EF	AF1	AF2	BE
Minimum Delay (s)	0.000434	0.00057	0.00057	0.00057
Maximum Delay (s)	0.001421	0.001378	0.001576	0.079058
Mean Delay (s)	0.000652	0.000773	0.00078	0.022968

Table 5-8: Delays averaged over all the node pairs for different traffic classes for problem 5B with the design traffic.

The delay information was obtained as in the previous experiment. It is given in Table 5-8. The results obtained for this problem are similar to those obtained for the previous one. The mean delays for the EF traffic are lower than those for the AF1 traffic, which, in turn, are lower than those for the AF2 traffic. In all cases, the delays experienced by the premium traffic are much lower than the target delays. The delays for the BE traffic are considerably larger.

As with the previous case, the averaged maximum EF delay is larger than the averaged maximum AF1 delay. The reasons used to explain this in the previous case are also valid here.

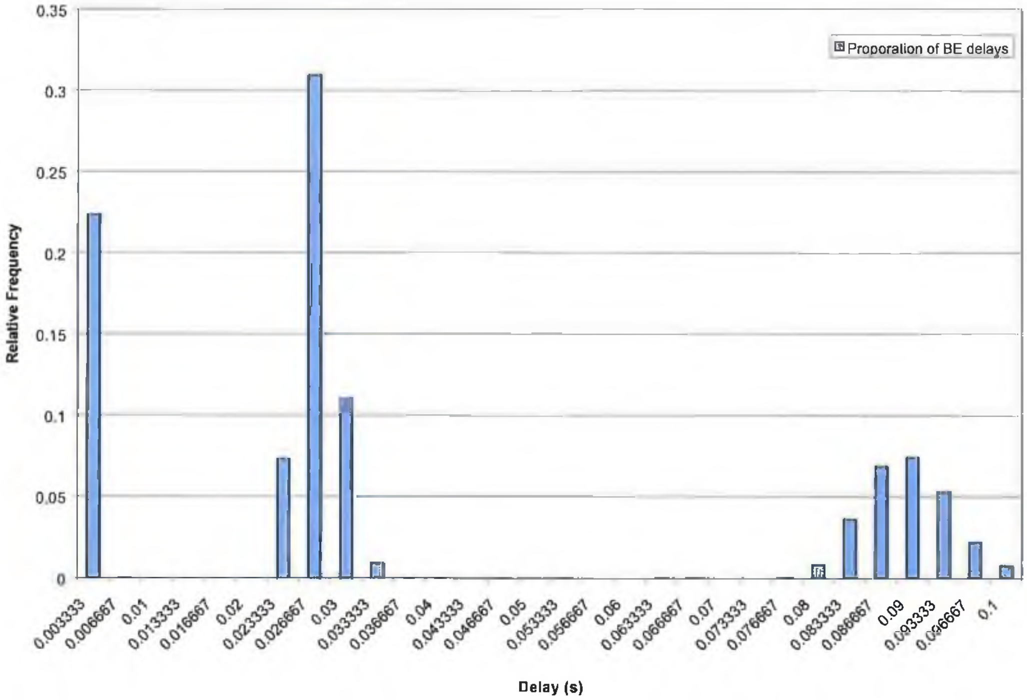


Figure 5-8: Distribution of BE delays for traffic arriving at a customer premises in problem 5B with design traffic. A small proportion of the delays exceed 100ms.

A sample of the delays for the BE traffic is shown in Figure 5-8. This exhibits similarities to Figure 5-7. Again there are three peaks in the graph: one occurs at low delays, one is at about 25ms and another at considerably higher values of delays. Again, the first peak occurs because some of the BE traffic arriving at the customer premises traverses a lightly loaded link experiences little or no queuing. The second peak occurs due to traffic traversing a single congested link, while the third peak occurs due to traffic traversing two congested links. Figure 5-8 differs from Figure 5-7 in that many more packets experience no delays in the former and the number of packets that traverse two congested links is much smaller.

As with the previous case, the conditioners do not have a very great impact on the results obtained. The EF traffic does experience approximately 1% loss, as in the previous experiment. This can be alleviated by increasing the amount of resources and the conditioner policing parameters by 5% over the negotiated EF rate. Similarly, there are a very tiny percentage of AF drops.

Simulating Problems with Increased EF Traffic

The effects of increasing the EF traffic in the network were examined next. Half of the EF sources were chosen at random and the amount of traffic generated by these sources was doubled. As would be expected, the impact of this on the network was minimal: the conditioners removed much of the excess EF traffic and the traffic that was permitted access to the network experienced similar conditions to those of the experiment with the design traffic.

Link	EF loss	AF1x loss	AF2x loss	BE loss
(0→2)	0	0	0	0.173197
(0→3)	0	0	0	0.338660
(1→3)	0	0	0	0.178748
(2→0)	0	0	0	0.404585
(3→0)	0	0	0	0.464489
(5→0)	0	0	0	0.067283

Table 5-9: Loss at queue/scheduler systems for specific links in problem 5A with increased EF traffic.

Problem 5 was simulated with increased EF traffic. The link utilisations are very similar to those in the experiment with the design traffic and the scheduler behaved in a very similar fashion to before: only the BE traffic experienced any loss in the network. The link utilisations were within 1-2% of those obtained in the previous experiment. Similarly, the loss experienced by the BE traffic at each of the different schedulers is within a few percent of the loss experienced in the experiment above. Delays are also similar in this experiment.

Customer ID	Premises number	BE loss	Customer ID	Premises number	BE loss
0	0	0.349034	4	0	0.274487
0	1	0.275961	4	1	0.261317
0	2	0.250303	4	2	0.335936
0	3	0.316149	4	3	0.393947
0	4	0.256898	4	4	0.242054
1	0	0.402977	5	0	0.386674
1	1	0.323102	5	1	0.32796
1	2	0.256824	5	2	0.45099
1	3	0.393514	5	3	0.392304
1	4	0.456485	5	4	0.349312
2	0	0.392956	6	0	0.412665
2	1	0.307318	6	1	0.264706
2	2	0.327093	6	2	0.330626
2	3	0.317043	6	3	0.332762
2	4	0.331132	6	4	0.249807
3	0	0.344955			
3	1	0.251404			
3	2	0.16679			
3	3	0.330109			
3	4	0.353163			

Table 5-10: Fraction of EF traffic lost at conditioners in problem 5A with increased EF traffic.

The real difference in this experiment lies in the effects of the conditioners and the end-to-end perceived behaviour. Firstly, the EF conditioners drop much more traffic in this experiment than in the previous experiment. The losses measured at each of the traffic conditioners are listed in Table 5-10. This is in contrast to a loss of approximately 1% in the reference experiment. Secondly, the perceived throughput for the EF traffic differs from the source transmission rate. This is due to the traffic that is dropped at the conditioners. The conditioner imposes an upper bound on the throughput: if the source exceeds this, then the transmission rate is reduced to that permitted by the conditioner. The delays perceived by all services are very similar to those of the reference experiment, since no more traffic is introduced into the network.

Similar observations were made on problem 5B with the EF traffic increased in the same manner.

Simulating Problems with Increased BE traffic

The BE traffic was increased: 50% of the BE traffic sources were doubled in intensity. Since the BE traffic is not conditioned at the edge of the network, much of this excess traffic entered the core network loading the core network further. However, the high priority traffic experienced little change.

Link	Used Capacity	Utilisation
0→2	155000	1
2→0	155000	1
0→3	155000	1
3→0	155000	1
0→5	155000	1
5→0	155000	1
1→3	155000	1
3→1	155000	1
1→4	82479.2	0.532124
4→1	101857.7	0.657147
1→5	58871.85	0.379818
5→1	50628.67	0.326637
3→6	135078.4	0.871474
6→3	155000	1
4→5	155000	1
5→4	155000	1
5→6	90700.49	0.585164
6→5	109171.1	0.70433

Table 5-11: Link utilisation for problem 5A with increased BE traffic.

The link utilisations for the core network are shown in Table 5-11. The link utilisations are considerably greater than in the reference case due to the increased BE traffic. However, not all links are fully utilised and consequently, some links experience no loss. Of those links that do experience loss, only BE packets are lost – there is no loss for the higher priority packets. Naturally, the amount of BE loss is considerably greater than in the reference experiment.

	EF	AF1	AF2	BE
Minimum Delay (s)	0.000434	0.000572	0.000572	0.000576
Maximum Delay (s)	0.002601	0.002335	0.006957	0.137427
Mean Delay (s)	0.000703	0.000769	0.000784	0.054825

Table 5-12: Delays for experiment 5A with increased BE traffic.

This experiment introduces considerably more traffic into the network. Hence it could potentially affect the delays perceived by the other traffic classes. The delays obtained are shown in Table 5-12. There is a substantial increase in the delays experienced by the BE traffic over the reference case and there are small increases in delay for the other traffic classes. The increase in the delay for the other traffic classes can be explained by the increased likelihood of a scheduler being busy serving a packet in a queue/scheduler system when a packet arrives. These increases are not very substantial: the delays for the higher priority traffic remain very small.

Similar effects were observed with the larger problem, although the increase in the delays experienced by the BE traffic was considerably larger.

Simulating Problems with Increased AF1 and AF2 Traffic

Finally, the amount of AF traffic entering the network was increased. The AF traffic was increased by doubling the peak and mean rate for 50% of the AF sources. All other elements of the simulation remained the same; in particular, the parameters of the conditioners at the edge of the network remained unchanged.

As before, the experiments were performed for both problem 5A and problem 5B, but the results exhibited the same characteristics. Consequently, only the results for problem 5A are presented here.

These results differ a little from the previous ones in that some AF traffic is lost: in the previous cases the only packets that were dropped were BE packets. However, it is worth pointing out that only a tiny fraction of AF packets are dropped and also that only packets that are out of profile are dropped.

In the experiment with increased AF1 traffic, the AF2 traffic suffered a very small amount of loss. Interestingly, no AF1 packets were lost, even though the AF1 resources were more heavily loaded.

	EF	AF1	AF2	BE
Minimum delay (s)	0.000434	0.000572	0.000572	0.000576
Maximum delay (s)	0.001845	0.001366	0.004945	0.117079
Mean delay (s)	0.00068	0.000746	0.000757	0.030063

Table 5-13: Delays for experiment 5A with increased AF1 traffic.

	EF	AF1	AF2	BE
Minimum delay (s)	0.000434	0.000572	0.000572	0.000576
Maximum delay (s)	0.002385	0.002042	0.00936	0.123503
Mean delay (s)	0.000688	0.000748	0.000782	0.030983

Table 5-14: Delays for experiment 5A with increased AF2 traffic.

The AF1 queues suffered no packet loss for the simple reason that the resource allocation for AF1 is quite conservative. Hence, when only high priority AF1 traffic is using the resources, they are somewhat under-utilised. It is possible to carry more traffic on these resources without having much of an impact on the performance perceived by current users of the resources. Hence, the amount of traffic generated by some users can be increased without adversely affecting the QoS perceived by the users. This is what happens in this case.

The packet loss that occurs for the AF2 traffic can be considered to be within normal operating conditions for the AF2 resources. A very small amount of packet loss can occur. Three conditions must be satisfied for this to occur: the packet must be low priority, a queue on the packet's path must be (at least moderately) congested and the AQM mechanism must choose to drop the packet. It is rare that these three conditions will be met if the traffic conditioners are generating traffic which is almost exactly conformant with the conditioner, but it is possible. Hence, 0.02% packet loss occurs for the AF2 traffic at a congested queue/scheduler system.

In the case in which the AF2 traffic is increased, there is increased loss of AF2 out-of-profile traffic. This occurs on two links in this case. The loss is very close to 0.5% on one of the links and is just over 0.06% for another link. Interestingly, this link does not reach full saturation, although it is very highly loaded at just over 96%. In this case, the link is saturated over some periods of time. Also, the arrival rate to the AF2 queue is greater than the rate at which it can be serviced, causing the mean occupancy of the queue to increase. When this is sufficiently high, and there is a significant amount of

AF2 out-of-profile traffic arriving at the queue, some of it gets dropped due to the AQM mechanisms. This occurs here. During other periods, the link is may not be saturated, but some loss may still occur because the mean queue length can remain high for a short time even though the queue is emptying.

While increasing the amount of AF traffic entering the network does have a deleterious effect on the performance, the performance is still well within the design parameters.

	Shortest Path Routing	Load Balanced
Problem 1 (conn=2.5)	Mean: 1030.94 Std. Dev.: 416.3 Minimum: 181.583 Maximum: 2301.98	Mean: 1042.27 Std. Dev.: 393.027 Minimum: 272.184 Maximum: 2301.98
Problem 2 (conn=3.0)	Mean: 616.65 Std. Dev.: 375.049 Minimum: 137.749 Maximum: 2048.19	Mean: 642.799 Std. Dev.: 303.031 Minimum: 199.359 Maximum: 1643.88
Problem 3 (conn=3.5)	Mean: 612.088 Std. Dev.: 346.048 Minimum: 26.0788 Maximum: 1833	Mean: 628.694 Std. Dev.: 173.663 Minimum: 270.895 Maximum: 1043.24
Problem 4 (conn=4.0)	Mean: 414.836 Std. Dev.: 211.891 Minimum: 44.4009 Maximum: 1050.34	Mean: 426.549 Std. Dev.: 123.244 Minimum: 123.352 Maximum: 938.602
Problem 5 (conn=4.5)	Mean: 360.436 Std. Dev.: 153.852 Minimum: 80.5553 Maximum: 726.974	Mean: 368.488 Std. Dev.: 58.9925 Minimum: 251.406 Maximum: 632.197
Problem 6 (conn=5.0)	Mean: 331.667 Std. Dev.: 156.724 Minimum: 41.641 Maximum: 816.796	Mean: 336.299 Std. Dev.: 88.7507 Minimum: 130.581 Maximum: 708.268

Table 5-15: Basic statistical properties of link loads obtained when specific problems solved with this approach. The problems vary in network connectivity.

5.6.3 Solving Larger Problems

This approach to determining a good network configuration was, to some extent, validated above, in the sense that the QoS perceived by the end users was delivered appropriately. However, the load balancing nature of the network optimisation was not visible. No efforts were made to demonstrate the load balancing characteristics of the approach because the networks were so small that the choice of routes was limited and hence there was little chance of performing any load balancing.

Here, some effort is made to redress this: the purpose of this section is to illustrate that the approach described above can be used to balance load on networks.

In order to do this, some sample problems were generated using the problem generator described in section 5.6.1 above. Larger problems were generated and they were solved using the network optimisation framework.

Two sets of experiments were performed here. In the first, test problems with varying levels of connectivity were generated. These were generated in order to determine the effect of the connectivity of the network on the load balancing. In the second set of test problems, the effects of varying the cost function generated within the mapping function were determined.

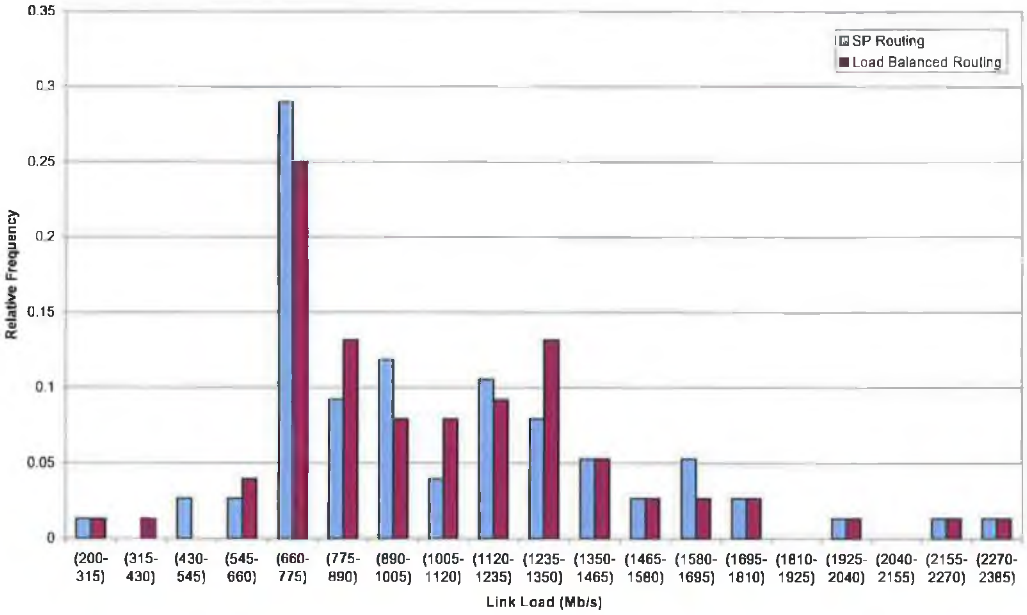


Figure 5-9: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 2.5.

Variation of Load Balancing with Connectivity

In the first set of experiments that were performed, the network connectivity was altered. Six specific problem consisting of 30 nodes and varying amounts of links were generated. The demand between the node pairs was approximately the same in each of these problems. The node connectivity varied from 2.5 to 5.0 in steps of 0.5.

The problem was mapped to the generic problem; the generic problem was then solved using the greedy approach: the results from chapter 4 showed that the greedy algorithm obtained the best results. This resulted in a route configuration.

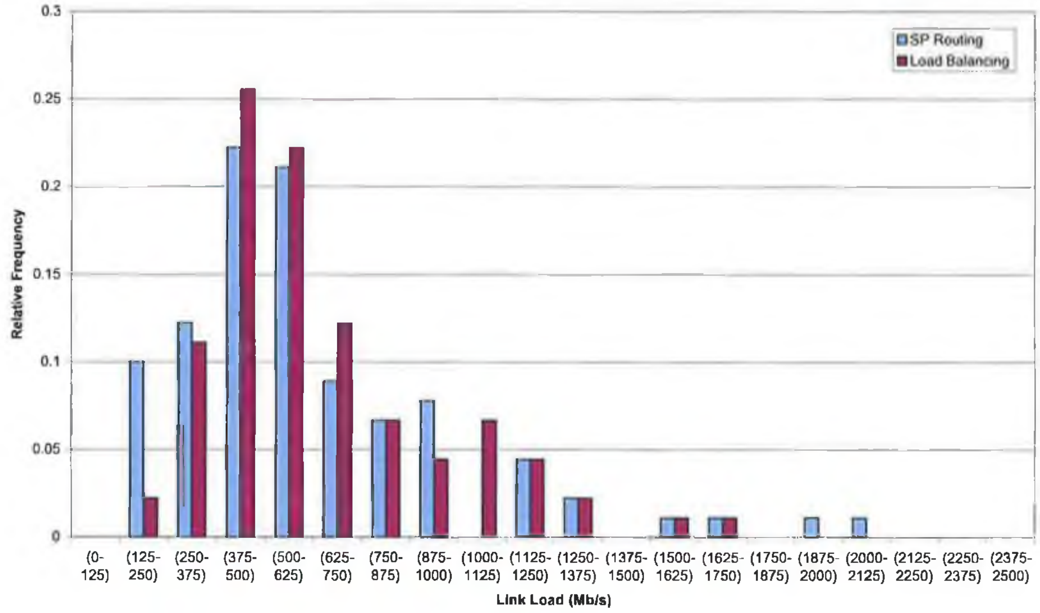


Figure 5-10: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 3.0.

To determine whether the network optimisation resulted in a load balanced solution, the mean and standard deviation of the link utilisations were calculated. These are shown in Table 5-15. There, it can be seen that the load balanced solution results in slightly higher mean link utilisation and lower standard deviation. This indicates that the load on the network is more balanced after the load balancing optimisation has been performed, i.e., the optimisation is useful.

This analysis is not sufficient to illustrate this. To illustrate this more clearly, histograms of link utilisations for each of the different problems were obtained. These are shown in Figure 5-9 to Figure 5-14.

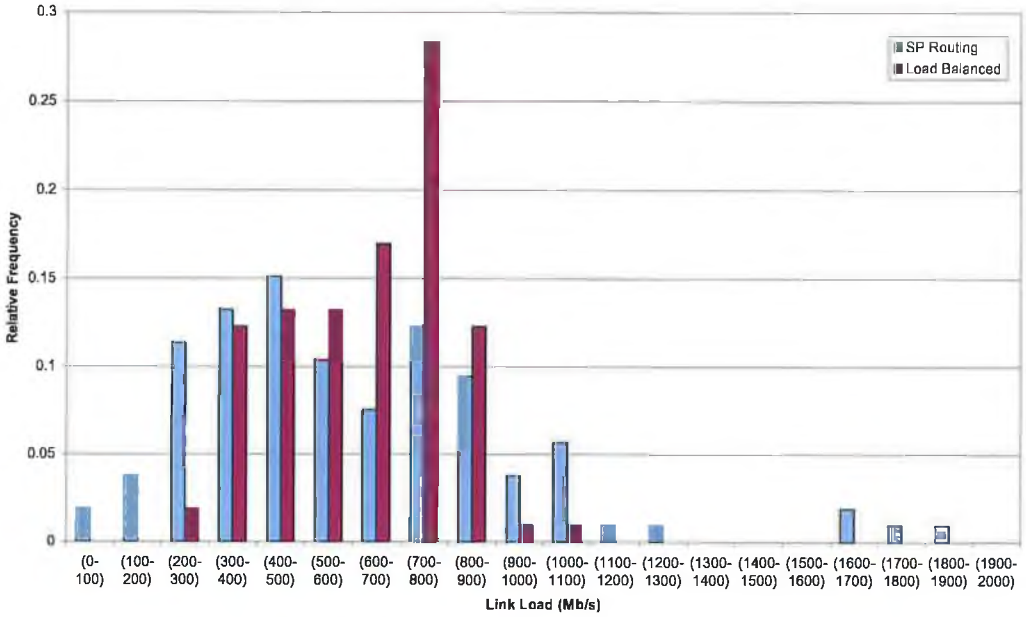


Figure 5-11: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 3.5.

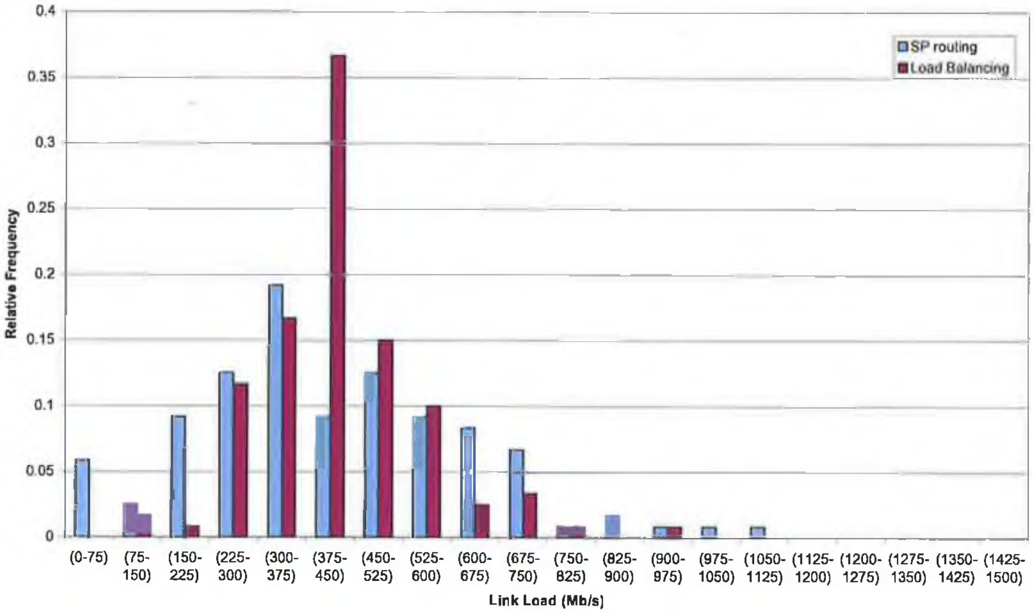


Figure 5-12: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 4.0.

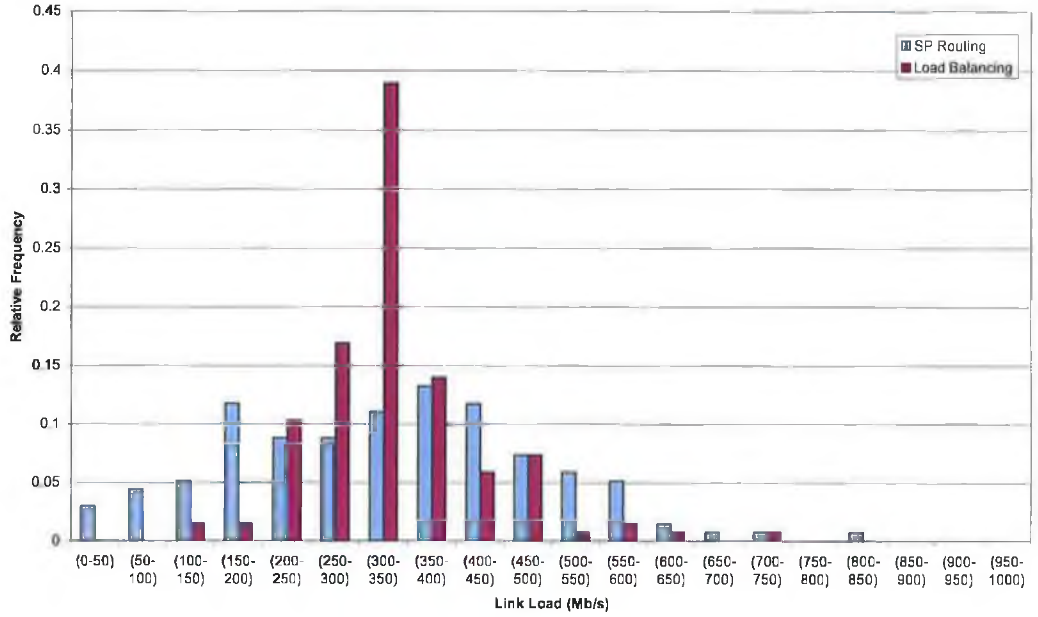


Figure 5-13: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 4.5.

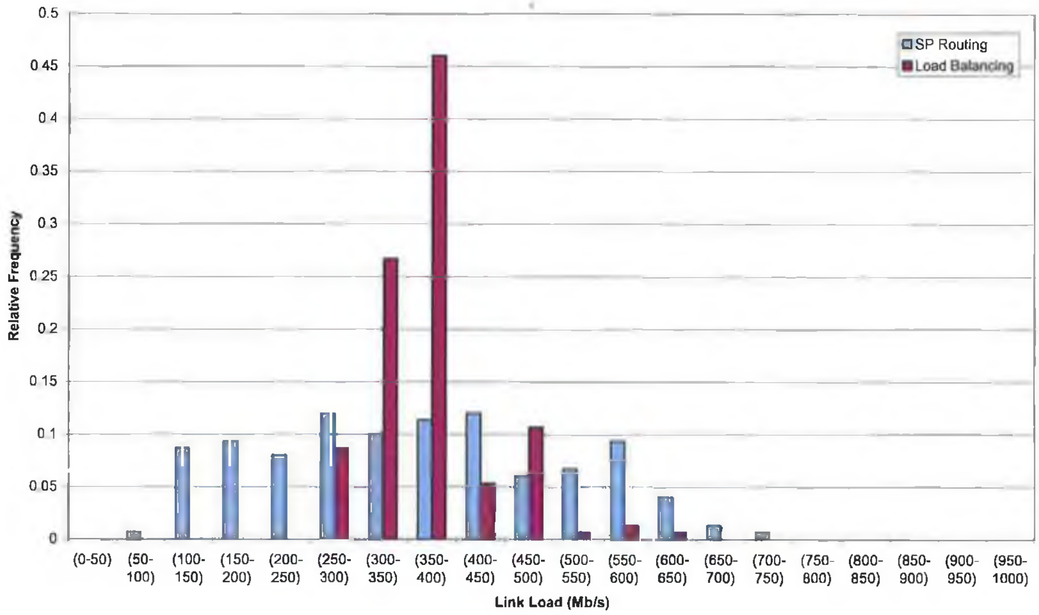


Figure 5-14: Variation of link load for SP routing and load balanced routing for 30 node problem with connectivity of 5.0.

The load balancing effect is particularly clear in Figure 5-14. In this figure, the shortest path routing clearly results in a network in which there is a quite uniform variation in link capacity on each link; i.e. link utilisation could almost be modelled by a uniform random variable. This clearly does not result in a balanced load since some links have high loads while others have minimal loads. The link utilisation is clearly much more like a normal distribution in the case of routing with load balancing. In this case, there is a very pronounced mean value for link utilisation and some links carry traffic which is a small variation off this. Since the link capacities on the network homogeneous, the free capacity on all of the links is then quite uniform and the network load is much more balanced.

Property	Problem 1		Problem 2	
Number of Nodes	30		30	
Core Link Capacities (Mb/s)	3000		3000	
Average node degree	5.0		3.0	
Number of Customers	5		5	
Parameter for EF traffic (Mb/s)	Min: 0.5	Max: 1.0	Min: 0.5	Max: 1.0
Parameter for AF1 Mean traffic (Mb/s)	Min: 0.5	Max: 1.0	Min: 0.5	Max: 1.0
Parameter for AF1 Peak/mean	Min: 2	Max: 4	Min: 2	Max: 4
Parameter for AF2 Mean traffic (Mb/s)	Min: 1	Max: 2	Min: 1	Max: 2
Parameter for AF2 peak/mean	Min: 1.5	Max: 5	Min: 1.5	Max: 5
Token bucket size for peak (bytes)	2000		2000	
Source on time (s)	Min: 0.1	Max: 0.3	Min: 0.1	Max: 0.3
EF end2end delay bound (ms)	15		15	
EF end2end loss bound (%)	1		1	
AF1 end2end delay bound (ms)	50		50	
AF1 end2end loss target (%)	2.5		2.5	
AF2 end2end delay bound (ms)	75		75	
AF2 end2end loss target (%)	4		4	
Random Seed	5097038		7567434	
Output	Bigproblemcost1.txt		bigproblemcost2.txt	
Max route length	6		6	

Table 5-16: Parameters used to generate problems to test the effects of the cost function used.

From the graphs, it is clear that the load balancing has a considerable effect in the problems in which there is more connectivity. This occurs mainly because there are more alternate routes for each demand. More specifically, more alternate routes which differ from the current route by only one node exist. Consequently, it is much easier for the algorithm to move the demand away from congested points than in cases in which the node connectivity is small and the number of options for rerouting a demand is small.

Problem	Shortest Path Routing	Load Balancing
Bigproblemcost1a (j=0.05)	Mean: 270.591 Std. Dev.: 126.31 Minimum: 19.3153 Maximum: 619.684	Mean: 277.199 Std. Dev.: 74.6691 Minimum: 129.449 Maximum: 619.684
Bigproblemcost1b (j=0.1)	Mean: 270.591 Std. Dev.: 126.31 Minimum: 19.3153 Maximum: 619.684	Mean: 277.633 Std. Dev.: 74.9909 Minimum: 135.471 Maximum: 619.684
Bigproblemcost1c (j=0.15)	Mean: 270.591 Std. Dev.: 126.31 Minimum: 19.3153 Maximum: 619.684	Mean: 277.964 Std. Dev.: 74.9769 Minimum: 128.846 Maximum: 619.684
Bigproblemcost1d (j=0.2)	Mean: 270.591 Std. Dev.: 126.31 Minimum: 19.3153 Maximum: 619.684	Mean: 278.175 Std. Dev.: 74.7925 Minimum: 126.081 Maximum: 619.684
Bigproblemcost1e (j=0.25)	Mean: 270.591 Std. Dev.: 126.31 Minimum: 19.3153 Maximum: 619.684	Mean: 278.079 Std. Dev.: 75.0447 Minimum: 125.477 Maximum: 619.684

Table 5-17: Basic statistical properties of link loads obtained when specific problems solved with this approach. The link cost functions vary – specifically, the pre-multiplier in the link cost varies.

For problems with smaller connectivity, the load balancing effects of the algorithm used are not so clear. For example, in Figure 5-9, the difference between the distribution of the link capacities and the case in which SP routing is used and that in which load balancing routing is used is minimal. The effects of load balancing are almost negligible: there are some load balancing effects as evidenced by the mean and standard deviation of the link utilisations, but they are very small.

Varying the Cost Function

The effects of using different cost functions were measured next. The link cost function used here is of the form

$$f\left(s_l^{\text{up}}, s_l^{\text{down}}\right) = e^{j\left(s_l^{\text{up}} - c_l\right)} + e^{j\left(s_l^{\text{down}} - c_l\right)}$$

j is a parameter that can be varied. The effects of choosing different values of j are studied here.

Two problems were generated. The characteristics of the problems are listed in Table 5-16. The two problems differed only in their connectivity: the network in the first problem had an average connectivity of 5.0 and the network in the second problem had an average connectivity of 3.0. The first problem is labelled Problem 1 and the second problem is labelled Problem 2. These specific problems were then mapped to a number of different generic problems. The only difference between these generic problems was the link cost function – it differed in the premultiplier coefficient, j. Values of 0.05 to 0.25, varying in increments of 0.05 were used in generating the generic problems.

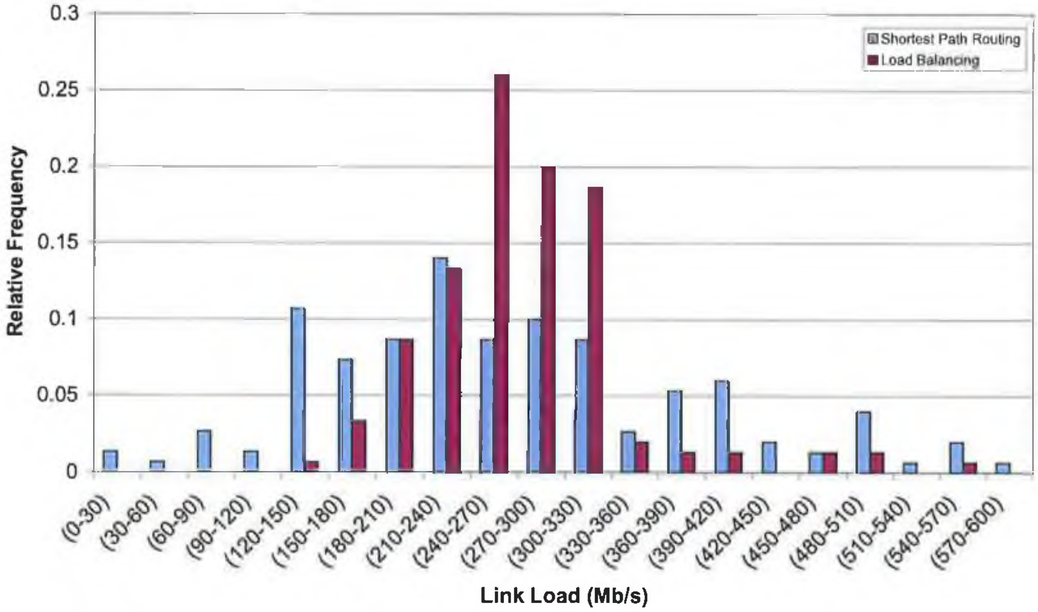


Figure 5-15: Variation of link load for SP routing and load balanced routing for Problem 1 with a premultiplier of 0.05.

These generic problems were then solved using the greedy approach. Each of the problems was solved in approximately an hour. The results obtained for solving Problem 1 are shown in Table 5-17 and the results obtained for solving Problem 2 are shown in Table 5-18. In each of the tables, the shortest path routing is the same: this is independent of the link cost function used. However, differences arise when the load is balanced on the network using the different cost functions. The results obtained from the network with higher connectivity differ from those obtained from the network with lower connectivity. The network with higher connectivity is discussed first.

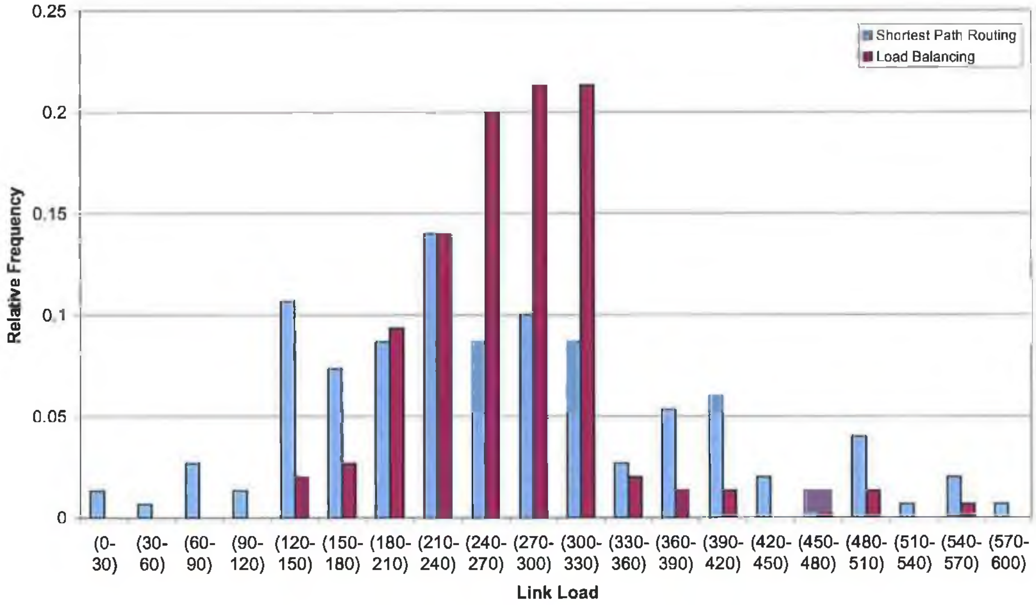


Figure 5-16: Variation of link load for SP routing and load balanced routing for Problem 1 with a premultiplier of 0.25.

The results obtained by solving Problem 1 for a number of different premultipliers are shown in Table 5-17. There, it can be seen that the effects of varying the link cost function are not so great. The lower premultiplier does result in a slightly better result and doesn't take any more time to solve. Hence, it is better to use this. However, the size of the premultiplier chosen shuld be dependent on the size of the link capacities being used in the problem.

The graphs also show the same effect. The link load histograms for Problem 1 when mapped to the generic problem using a premultiplier of 0.05 and 0.25 are shown in Figure 5-15 and Figure 5-16 respectively. When the premultiplier is 0.05, the link load graph has a more pronounced single peak than when the premultiplier is larger.

For the problem with less connectivity, varying the premultiplier over this range has almost no effect. In this problem, the number of ways of rerouting a demand is smaller and hence the effects of the premultiplier are minimal. This can be seen from Table 5-18. There, some load balancing takes place, as can be seen by the small difference between the standard deviations for the solutions with shortest path routing and routing in which the load is balanced on the network. However, the differences in the different load balanced solutions are very minimal and hence it can be concluded that varying the cost function parameter in this case has little effect.

	Shortest Path Routing	Load Balancing
Problem 2a	Mean: 578.116 Std. Dev.: 342.49 Minimum: 182.178 Maximum: 2191.11	Mean: 585.704 Std. Dev.: 320.537 Minimum: 217.822 Maximum: 2191.11
Problem 2b	Mean: 578.116 Std. Dev.: 342.49 Minimum: 182.178 Maximum: 2191.11	Mean: 585.615 Std. Dev.: 320.572 Minimum: 217.822 Maximum: 2191.11
Problem 2c	Mean: 578.116 Std. Dev.: 342.49 Minimum: 182.178 Maximum: 2191.11	Mean: 585.688 Std. Dev.: 320.58 Minimum: 217.822 Maximum: 2191.11
Problem 2d	Mean: 578.116 Std. Dev.: 342.49 Minimum: 182.178 Maximum: 2191.11	Mean: 585.653 Std. Dev.: 320.548 Minimum: 217.822 Maximum: 2191.11
Problem 2e	Mean: 578.116 Std. Dev.: 342.49 Minimum: 182.178 Maximum: 2191.11	Mean: 585.702 Std. Dev.: 320.807 Minimum: 217.822 Maximum: 2191.11

Table 5-18: Basic statistical properties of link loads obtained when Problem 2 is solved with this approach. The link cost functions vary – specifically, the pre-multiplier in the link cost varies.

5.7 Conclusions

In this chapter, the application of the network optimisation framework to the problem of configuring diffserv/MPLS networks was discussed. Diffserv and MPLS technologies were described in detail, followed by a discussion of a particular implementation. This implementation is assumed here and the parameters that need to be determined to realise this solution are highlighted.

The specific problem that is considered here is then defined, and the objective of the problem is discussed. The problem is then mapped to the generic problem. This mapping is complicated by the fact that some of the demands are assumed to be characterised by a number of parameters: the mapping must reduce this to a single parameter. The effective bandwidth ideas developed within the context of ATM are used to perform this mapping.

The use of the approach is then illustrated by solving some example problems. First, some small example problems are solved. These are simulated to demonstrate that the approach works. This is then followed by use of the approach within a larger problem context. There, it is demonstrated that the approach has some use: the objectives are shown to be met to some extent.

CHAPTER 6 CONCLUSION

A flexible, abstract network optimisation framework has been motivated and described. The purpose of this framework is that it may be applied to different network design and configuration problems to enable software to be quickly developed to solve network design and configuration problems. This is a fundamentally different approach from that typically taken today in which each network design or configuration problem is considered in isolation. In this framework-based approach, an abstraction of a network design/configuration problem is devised and solution software is developed to solve this abstract problem. This solution software can be reused in order to solve many different network design and configuration problems. Thus, the time required to develop solution software is reduced.

Another important benefit of this framework based approach – which arises in any framework designed for reuse – is that there is increased confidence in the correct operation of the reusable components. Since these components are used in many different situations, any bugs or problems with these components should become apparent quickly. In this case, the reusable components are the solution algorithms to the generic problem. In the case in which software is written to solve a specific network design/configuration problem the level of confidence in the results is lower because the solution software will not be tested as well.

Many network optimisation problems were reviewed in Chapter 2 in order to identify the most essential characteristics of network design and configuration problems. It was observed that many network optimisation problems focus on determining how a set of demands should be routed on a network.

The framework concept was motivated in Chapter 3. Then the generic model, which lies at the heart of this framework, was described. As observed in Chapter 2, many of these problems focus on routing of demands and this is what the generic problem then focuses on. A set of nodes, links, demands and cost functions are input to the problem. The solution to the problem is a minimum cost routing of the demands on the network. The model permits arbitrary cost functions and consequently it can be used as an abstraction for a large amount of different network design/configuration problems.

A number of different approaches to solving the problem were discussed. These were based on the use of local search heuristics that are applicable to combinatorial optimisation problems. Two particular approaches were chosen for use throughout the remainder of the work – one based on a straightforward greedy algorithm and one based on the simulated annealing algorithm.

The use of the framework to solve two very different network optimisation problems was demonstrated in Chapters 4 and 5. The first problem is an enterprise network design problem and the second is a diffserv/MPLS network configuration problem. In both cases, it was shown that it is possible to use the framework to obtain solutions to the specific problems.

In chapter 4, the use of this approach was illustrated in the context of a particular enterprise network design problem. The objective in this problem was to design a network of leased interconnects which can carry an enterprise's inter-office voice and data traffic at a minimum cost. The interconnects could be realised using either leased lines, FR or ATM and could have arbitrary cost/capacity characteristics. The solution involved determining a network topology as well as a routing for the demands on the network. The network cost and the required link capacities are implied once the topology and routing are specified.

The particular enterprise network design problem was mapped to the generic problem and the generic problem solvers were used to obtain solutions, validating the use of the framework. Using these problem solvers, 50 node problems could be solved in a matter of hours. The time required to obtain a solution increased exponentially; hence, the solution time quickly ran to days for problems larger than 50 nodes. Note that this is a characteristic of the specific solution algorithm used: it is not something which is characteristic of the framework approach.

This problem provided a context for comparing the two generic problems solvers – that based on the greedy algorithm and that based on the simulated annealing algorithm. The solutions obtained using the greedy algorithm were substantially better than those obtained using the simulated annealing algorithm for all choices of the parameters of the simulated annealing algorithm. Moreover, the solution obtained using the greedy algorithm was of much higher quality for larger problems. The choice of state space was not very suited to the use of the simulated annealing algorithm. Better results could be obtained using a different choice of state space. This is discussed further below.

In Chapter 5, a core diffserv/MPLS network configuration problem was described. The objective in this problem was to determine how to configure the network such that the load could be balanced on the network while delivering the desired QoS to the customers. This involved determining how the demands could be routed on the network as well as determining how to configure the routers in the network.

In the specific problem, some of the demands were specified in terms of packet level parameters. In the generic problem demands are characterised by a single parameter. The mapping function mapped the demands characterised by multiple parameters in the specific problem to a demand characterised by a single parameter for the generic problem. This was done using a variant of the effective bandwidth concept that was developed for ATM. Using this approach, packet level issues could be decoupled from the routing problem. With the effective bandwidth concept it was possible to focus on capacities – both used capacity and available capacity – and load balancing could be performed by balancing these capacities on the network.

Load balancing was achieved in all cases, validating the use of the framework. However, the amount of load balancing that was achieved was dependent on some characteristics of the network. Specifically, the amount of links present in the network had an impact on the amount of load balancing could be achieved. If there are more links present in the network, then there are more options for routing a particular demand and resulting load will be more balanced on the network.

The required QoS was delivered to the customers and this was verified through simulation. No reports of doing this have been reported in the literature so far. Although the approach involves allocating resources in a slightly conservative manner, the resources were allocated in a manner which was more liberal than reserving resources according to peak rate.

This framework could form the basis of the design of a network optimisation tool which could be used to solve different network optimisation problems. The reusable solution algorithms could form the core of the network optimisation tool and various mapping functions could be developed to map a number of different specific problems to the generic problem. Development of a module to solve a new problem would just involve developing the mapping function and integrating it into the optimisation tool: something which would certainly involve less effort than developing a customised tool to solve the problem.

6.1 Contributions in this Thesis

The main contributions in this work can be summarised as follows:

- A flexible abstract network optimisation framework was developed;
- A generic problem on which the optimisation framework is based was developed;
- Approaches to solving the generic network optimisation problem based on combinatorial optimisation were devised;
- An approach to designing enterprise networks with voice and data demands was developed using the network optimisation framework;
- An approach to configuring core diffserv/MPLS networks based on this network optimisation framework was developed.

A final contribution, which was used in the work above, but was not discussed in detail is software to simulate a diffserv/MPLS network. This is an interesting and useful contribution which has been used by others to simulate diffserv/MPLS networks.

6.2 Directions for Future Research

The work could be extended in a number of different ways. The directions for extending the work can be divided into two areas: development of the framework and development of the approaches to solve the two specific problems considered in this work. Each of these is considered separately.

6.2.1 Development of the Framework

There are two further divisions that framework developments can take: the core of the framework can be developed or the framework can be extended to accommodate more and more cases. These are discussed separately in turn.

Core Framework Development

It is conceivable that the generic problem proposed above has some deficiencies and that it could be generalised so that it could be applied in more cases. For example, the generic problem may be extended to incorporate traffic which has a broader scope than a single destination – a single demand may originate at one node, but consist of traffic destined for a number of nodes. Alternatively, the generic problem could be extended such that some reliability constraints could be incorporated into the generic problem model. The specific ways in which the generic model could be extended are unclear: this is the research problem. While these ideas appear interesting, the purpose of this framework is to make it general so that it can be applied in some set of cases. Incorporating more and more functionality into the model could make it more flexible, but at a cost of making it impossible to solve: it is necessary to find an appropriate balance between generality and tractability.

The problem solvers at the core of the framework could also be improved. Different algorithms could be used. Algorithms such as those referred to in Chapter 3 could be implemented and used to solve the problem. In particular, an approach based on pre-selection of paths could dramatically reduce the size of the state space and facilitate solution of larger problems. Also, combinatorial heuristics which 'learn' as they traverse the state space and learn characteristics of a good solution could be used. Improved algorithms could improve the efficiency of the approach and/or the quality of the solution obtained.

Extension of the Framework

The framework can be extended: other specific problems can be identified and they can be mapped to the generic problem. In principle, any mesh network design or configuration problem in which the demands can be reduced to a single parameter and the costs are a function of the demand can be solved using this approach. Hence, this approach can be used to design circuit-switched networks, FR, ATM, SDH and IP based networks, if the demands are appropriately specified and the appropriate mapping functions determined. Also, the framework could be used to design networks which incorporate a number of different types of demands. Similarly, this network optimisation framework can be used to solve network configuration problems for networks using the above technologies.

The framework could be developed into a comprehensive network optimisation tool which could be used to solve different network design and configuration problems. The tool could have different modes of operation – one corresponding to each specific problem. The user would choose a mode of operation, enter the appropriate data, perform the mapping, entering any data that was required in mapping the specific problem to the generic problem and choose one of the generic problem solvers to obtain a solution to the problem.

The big advantage of such a tool is that it could be used to solve many network optimisation problems. Also, the tool could be designed to be easily extensible such that new mapping functions could be easily configured to enable the tool to be used to solve new problems.

6.2.2 Development of the Specific Problem Cases

The two specific problems studied here could also be developed further. These developments involve making the specific model more sophisticated and introducing more functionality into the mapping function to reduce this extra complexity to the same generic problem. The development of each of these problems within the context of the network optimisation framework is discussed here.

Developing the Enterprise Network Design Problem

The enterprise network design problem could be extended by constructing more complex traffic models for the traffic on the enterprise network and using these to design the network. These traffic models could be models for more sophisticated applications using the data network: videoconference applications, transaction based processing, etc. Profiles of typical amounts of users could be constructed at each location and these could be mapped to a set of demands which could be used in the generic problem. The problem with this approach would be to determine how the traffic mix can be reduced to the single parameter required in the generic problem. The advantage of such an approach would be that the optimisation would result in more cost-efficient network that accommodated the demands.

Another way that this problem could be developed is in terms of the variety of operators and/or technologies that could be used to meet the needs of the enterprise. The customer may have a choice of a number of different operators for some, say, long-distance and costly links in the network. Alternatively, the enterprise may have the option of a number of different technologies for a particular link. In either case, the result is a choice between different links having different cost/capacity characteristics. The specific enterprise network design problem discussed above does not support this: it would need to be extended to accommodate this case. If it were extended in this way, a way of incorporating the extension into the mapping function is described in Chapter 3.

Developing the diffserv/MPLS Network Configuration Problem

The approach used to solve this problem could be improved by determining some less conservative approach to allocating network resources. The AF traffic in particular suffers from this very conservative approach and while this means that QoS assurances can be made, it is probable that the same QoS assurances can be made if a less conservative approach to choosing the effective bandwidth is used. An interesting research problem would be to investigate different approaches to determining such an effective bandwidth which result in less resources being allocated to the AF traffic, but the required QoS being delivered.

Another direction in which the work could be extended is to permit some variation in the characteristics of the EF traffic: customers could produce traffic which varies in intensity with time but still require strict loss and delay bounds. This is in contrast to the EF traffic considered in this model: there, the peak rate for the EF traffic is allocated in the network. The variable EF traffic could be policed using dual leaky bucket policers and non-conformant traffic could be dropped. As with the AF traffic, an effective bandwidth could be determined for the EF traffic in this case which could be used in the generic problem.

A third direction in which the work could be extended is to the case of more than two drop precedences for the AF traffic. The problem described above assumes only two drop precedences but the standards [RFC2597] specify that up to three drop precedences are possible in each AF traffic class. The specific problem model described above cannot cater for this case. Indeed, it is not clear what kind of assurances can be given when all three drop precedences are used. This is an area which could warrant further study.

APPENDIX A EFFECTIVE BANDWIDTH DETERMINATION FOR AF TRAFFIC

The problem of determining the effective bandwidth for the AF traffic is discussed briefly in chapter 5. A high-level view of the approach is given there. A detailed description of how this effective bandwidth is determined is given here.

A.1 Determining the Appropriate Effective Bandwidth

Many different approaches to effective bandwidth problems have been proposed (see [Kel96] for an overview). Here, the objective was to use a simple approach which performs better than the simplistic peak rate allocation strategy. The approach of Courcoubetis and Weber described in [CFW94] was initially investigated, but proved problematic because the model considered often resulted in large bandwidths – larger than the peak bandwidth[18]. This approach was also somewhat limited because it focussed on a single traffic flow. The approach described in Guerin et al [GAN91] was then tried, since it extended the effective bandwidth concept to multiple traffic flows. This was moderately successful and is described here.

It is important to note that the effective bandwidth work described here was carried out in an ATM context in which the focus is on individual traffic flows – traffic flows generated by a single application. Here, however, the objective is to determine an effective bandwidth for an SLA in which many flows can be aggregated to together. In both cases, the determination of the effective bandwidth is based purely on the declared parameters and hence the same method can be used to determine an effective bandwidth in each case. Here, the term flow is used to identify that for which it is necessary to determine an effective bandwidth. In the ATM case, flow refers to the traffic generated by an individual application, while in this case, flow refers to an aggregate.

[18] This occurred because the expression they developed assumed that the buffer size was large. Here, to ensure small delays, the buffer size is necessarily small in some cases. Hence, the expression that they developed is not valid.

Guerin et al proposed an approach to determine the effective bandwidth of an aggregate of a number of flows, but. it can also be used to determine the effective bandwidth of a single flow. The emphasis in their work was on devising a means to obtain an effective bandwidth quickly. The application that they envisaged was an ATM switch which would be making many admission control decisions in a small amount of time: at that time ATM was viewed as a successor to the telephone network and many believed that ATM switches would have to handle requests for service in similar quantities to today's telephone networks. Consequently, admission control decisions had to be made quickly and hence effective bandwidths had to be determined quickly.

They proposed an approach consisting of two methods of determining an effective bandwidth. The resulting effective bandwidth is the lower of the two obtained using the different methods. The first method is one in which buffered multiplexing is assumed and the second is one in which bufferless multiplexing is assumed[19]. Buffered multiplexing refers to a system in which traffic is buffered before it is allowed access to the resource. Bufferless multiplexing refers to a system in which no such buffers exist and if data requires access to a congested resource, it is lost. The latter are not realistic systems, but are easier to model and can be used as some kind of approximation to buffered systems.

In the buffered multiplexing method of determining an effective bandwidth, a Markov model for the system is constructed and analysed. This is used to determine the capacity required for the aggregate of flows. In the bufferless multiplexing method, the effective bandwidth is determined by considering the stationary distribution of the number of active flows in the system and choosing the effective bandwidth such that the probability of the aggregate data rate of the flows exceeds the link rate is less than the loss probability. Both of these methods of choosing an effective bandwidth are conservative. Consequently, the smaller of the two can be chosen and it will still be a conservative estimate of the effective bandwidth. Both of these methods are discussed in more detail.

The first approach is largely based on the earlier work of Anick, Mitra and Sondhi [AMS82]. They derived expressions for the overflow probability of a buffer which is used to multiplex a number of on-off flows. The flows are assumed to be fluid sources

[19] These are sometimes referred to as rate sharing and rate envelope multiplexing, respectively.

with exponentially distributed on and off times. The flow transmits at some fixed rate in the ‘on’ state, and does not transmit any data in the ‘off’ state. The buffer is assumed infinite.

The overflow expression developed by Anick, Mitra and Sondhi can be written as

$$P[x > B] = \sum_{i=1}^{K} u_i e^{-v_i B}$$

where x is the buffer occupancy, B is some fixed position in the buffer, K is the number of multiplexed flows and u_i and v_i are parameters that Anick, Mitra and Sondhi show how to calculate. These parameters are difficult to calculate with the speed required by Guerin et al, so they simplified this expression.

For larger buffers, i.e., B is large, the above expression is dominated by the term containing the smallest v_i, $v_i = v_0$ so Guerin et al focus on this term. They also note that for a significant part of the state space, the pre-multiplier u_0 is approximately 1. Thus,

$$P[x > B] \approx e^{-v_0 B}$$

and (from the workings of Anick, Mitra and Sondhi) v_0 can be written as

$$v_0 = \frac{K(S - Kt' r^{\text{peak}})}{t^{\text{on}}(1 - t')(Kr^{\text{peak}} - S)S}$$

where S is the buffer service rate, t' is the fraction of time the source is in the ‘on’ state, r^{peak} is the peak rate of the source and t^{on} is the mean duration of a burst. t' can be written as

$$t' = \frac{t^{\text{on}}}{t^{on} + t^{\text{off}}}.$$

The loss probability, $P_{\text{loss}}(B)$ – the probability of loss in a finite buffer of size B – can be approximated by the overflow probability, $P[x > B]$. $P_{\text{loss}}(B)$, then, can be written as

$$P_{\text{loss}}(B) \approx e^{-v_0 B}.$$

An approximation to the required buffer service rate, which is the effective bandwidth for the aggregate set of sources, can be determined by choosing some loss probability, P_{loss}^{*}, rewriting the above equation with an equality and solving for S. This is straightforward, since all of the other parameters are known. Thus, the aggregate effective bandwidth of a number of homogeneous sources can be determined using the buffered multiplexing model.

The effective bandwidth for an individual source is simply the aggregate effective bandwidth divided by the number of sources, i.e.

$$S' = \frac{S}{K}$$

where S' is the effective bandwidth of an individual demand. This is then used in the generic problem. However, some care must be taken when using this effective bandwidth figure in the generic problem. As the number of multiplexed sources increase, the size of the effective bandwidth decreases. Hence, if the effective bandwidth is calculated based on the premise that a large number of sources will be multiplexed but the actual number is quite small, then the effective bandwidth could be underestimated. Since diffserv is designed for situations in which a large amount of flows will be aggregated, it is assumed here that the amount of aggregation and the number of demands will be substantial.

The effective bandwidth is obtained using the bufferless model in quite a different way. The stationary behaviour of the flow aggregate is considered. The activity of the flows can range from all flows being in the 'on' state to all flows being in the 'off' state. The effective bandwidth is chosen as kr^{peak}, where k is chosen between 1 and K, such that the loss is certain to be less than P_{loss}. This can be done by choosing k as the smallest integer such that

$$\sum_{i=k}^{K} g_i \le P^{*}_{\text{loss}}$$

where g_i is the probability that i flows are simultaneously in the 'on' state. Noting that the probability that i flows are on can be written as a binomial, the probability g_i can be written as

$$g_i = \binom{K}{i} (t')^i (1-t')^{K-i}$$

In [GAN91], Guerin et al chose to approximate this with a Guassian distribution – an approximation that is valid if the number of flows is large. This was done for speed reasons. Here, this approximation is not necessary, so the binomial expansion of g_i can be used directly. Also, the number of flows in this case may be smaller and the Gaussian distribution may not be a good approximation to the binomial. Thus, k can be determined and the effective bandwidth, kr^{peak}, can easily be calculated.

Both approaches to obtaining an effective bandwidth are somewhat conservative: the method based on the buffered multiplexing is conservative because the pre-multiplier, u_0 can often be considerably less than 1. This is discussed in [GAN91] and in more detail in [CLW94]; there is some discussion of this in [BCDM95]. Consequently, working on the premise that this term is 1 will produce conservative results.

Similarly, the bufferless model is conservative. This is true because no buffering is assumed when in fact there is some buffering. This buffering serves to reduce the amount of loss and hence the amount of loss predicted by the bufferless model is an overestimate. Since both methods result in conservative values for the effective bandwidth, it is reasonable to choose the lower result as the effective bandwidth.

In Guerin's approach, the only QoS measure is the loss probability. Here, the objective is also to ensure that the delay remains below some specified threshold. This can be done by permitting the queue size to be a design parameter; in Guerin's approach, the queue size was a fixed parameter. If the queue size is a design variable, there are two degrees of freedom when choosing the parameters of the system – the queue size and the effective bandwidth. This extra degree of freedom should permit both delay and loss constraints to be satisfied in a substantial amount of cases.

If the queue size and the effective bandwidth are variable, then delay and loss constraints can be satisfied. In general, small queue sizes are necessary to ensure small delays for some fixed service rate. Similarly, small losses are usually only possible in systems with larger queues. Hence, if both small losses and small delays are required, a kind of tension arises and a compromise between these differing objectives may not be possible **for the given service rate**. Increasing the queue service rate – the effective bandwidth – can solve this problem. By doing this, some compromise queue size that meets both the delay and the loss constraints can be used to meet the requirements. Hence permitting the queue size and the effective bandwidth to vary enables both the loss and delay constraints to be met.

Permitting queue sizes to be variable is not unreasonable in practice; existing routers are very flexible and facilitate sophisticated configuration of their buffer memory. In particular, it is not difficult to create a number of queues of specified size at an interface and define some scheduler discipline to enable the queues to be served.

One realistic constraint that could be added here is that the queue size should not exceed some pre-specified amount of buffer memory, i.e. the sum of all the queue sizes at an interface should not exceed the available buffer memory at the interface. However, this constraint is not incorporated into the problem here.

In the buffered multiplexing method used by Guerin some relationship between the queue size and the required capacity is developed. If both of these are free variables, then many queue size/capacity combinations exist that can satisfy the loss constraints. Not all of the solutions satisfy both the loss and delay requirements – this immediately rules out some of them. A trade-off exists in the remaining solution set: the trade-off exists between queue size and effective bandwidth – larger queue sizes result in smaller effective bandwidths and smaller queue sizes result in larger effective bandwidths. The question is then which queue size/effective bandwidth combination should be chosen

Here, a policy of choosing the lowest effective bandwidth is used. Choosing the lowest bandwidth is consistent with a bandwidth-constrained mindset, i.e. bandwidth is somewhat limited in the system. If there is an abundance of bandwidth, then diffserv is arguably not necessary (Kelly discusses this in [Kel00]) – it is implicitly assumed in this discussion that such an abundance does not exist and that congestion will occur at least sometimes.

To determine the appropriate effective bandwidth and buffer size, then, an additional relation is required. This relation relates the delay to the queue size and the effective bandwidth:

$$d = \frac{B}{h}$$

where B is the queue size and h is the effective bandwidth.

This can be used in conjunction with the Guerin's buffered multiplexing model to obtain a queue size and effective bandwidth that meets the loss and delay requirements. Moreover, the resulting solution is the minimum effective bandwidth solution.

The two equations cannot be solved easily using analytical means. An iterative numeric approach is used in which the buffer size is modified and the resulting effective bandwidth determined. This is repeated until the effective bandwidth obtained is as small as possible, but still meets the loss and delay constraints.

There is no queue in the bufferless multiplexing model used by Guerin and the determination of the effective bandwidth in this method is not dependent on any buffer size. In this case an effective bandwidth is determined first using the approach described above and a queue size is then calculated using the equation relating delay, buffer size and effective bandwidth above.

As before, the overall effective bandwidth/buffer size couplet is chosen by choosing the minimum effective bandwidth obtained via the two approaches and the queue size is its associated queue size.

This enables an effective bandwidth and buffer size to be obtained for an aggregation of a set of homogeneous flows. It is assumed that the effective bandwidth of each individual flow can be obtained by dividing the aggregate effective bandwidth by the number of flows. Similarly, the queue size required for each flow can be obtained by dividing the aggregate queue size by the number of flows.

Using this approach, the effective bandwidth for a demand can be determined **if it is multiplexed through a single buffer**. If the flow traverses a number of buffers, then some loss and delay can occur at each buffer and the overall loss and delay can be more unpredictable.

Here, the emphasis is on end-to-end assurances: the objective is to be able to make end-to-end delay and/or loss assurances to the customer. The single buffer results must be applied in a way that enables end-to-end assurances to be made. This is done by assuming that the end-to-end performance parameters can be divided equally between the number of stages traversed by the demand. Since the number of multiplexing stages a flow traverses is not known in advance, it is assumed – for the purposes of calculating the effective bandwidth – that the flow traverses some maximum number of multiplexing points.

If the maximum number of multiplexing points is $m_{\max}$ and the end-to-end loss and delay are $l_{\text{end-to-end}}$ and $d_{\text{end-to-end}}$ respectively, then the permitted loss and delay at each queue can be written as

$$l_{\text{queue}} = \frac{l_{\text{end-to-end}}}{m_{\max}}$$

and

$$d_{\text{queue}} = \frac{d_{\text{end-to-end}}}{m_{\max}}.$$

The end-to-end delays, however, do not consist only of queueing delays – they also consist of transmission and propagation delays. Hence these must be taken into account when calculating the effective bandwidth. A conservative approach to doing this is to assume the maximum route length, and assume some propagation and transmission delays and subtract these from the delay budget to obtain the amount of delays that can be incurred from queueing.

It is straightforward to calculate these values when the end-to-end performance objectives are specified. These are then used as inputs – together with the flow characteristics – to the procedure above to obtain a buffer size and bandwidth that can deliver the required QoS.

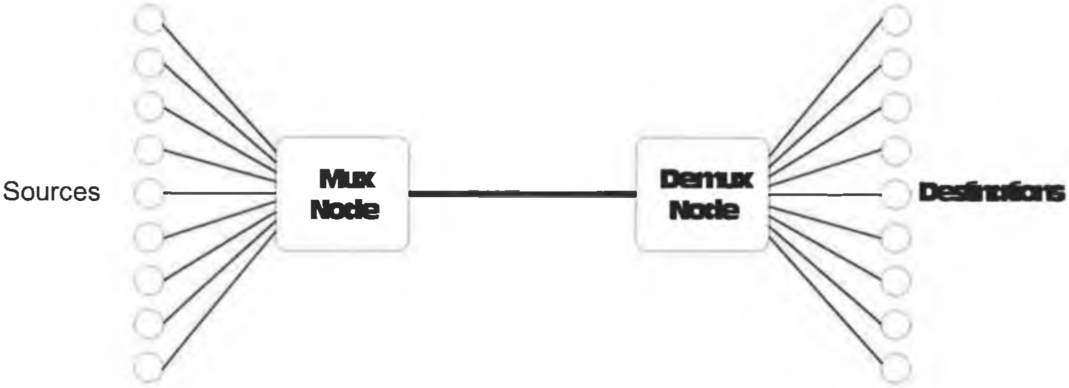


Figure A-1: Topology used in simple scenario used to validate approach to determine effective bandwidth.

A.2 Testing the Effective Bandwidth Approach

A rudimentary test was performed to test the validity of this approach. In this test, a number of traffic sources are multiplexed into a single buffer. The objective is to demonstrate that the approach to determining the effective bandwidth and buffer size described above meets the loss and delay constraints. The effective bandwidth and buffer size for each of the flows is determined using the approach described above. These are then added to obtain the overall effective bandwidth required to serve the queue. Similarly, the buffer sizes associated with each effective bandwidth are added to obtain the total buffer size. The scenario was then simulated and the resulting loss and delay measured.

The experiment contained 9 flows multiplexed in a single buffer. The topology is illustrated in Figure A-1. All of the source were on-off sources with exponentially distributed on and off times. Of the 9 flows, one was monitored and delay statistics for the monitored source were obtained. The amount of loss occurring at the multiplexing stage was also measured.

In this experiment, the access links from the source nodes to the multiplexing node have a capacity of 1Mb/s. The link that all of the sources are being multiplexed onto has a capacity equal to the aggregate effective bandwidth, as does the access link to the destination nodes. The queue size at the multiplexing node is obtained as described above.

The source parameters were chosen at random. These are shown, together with the source effective bandwidth in Table A-1.

Source	On-time (s)	Off-time (s)	Peak Rate (b/s)	Packet Size (bytes)	Effective Bandwidth (b/s)	Buffer Size (bits)
1	0.575758	1.12453	515002	333	507090	5072.02
2	0.482431	1.17507	261687	375	209350	2093.5
3	0.291242	0.622087	768253	217	614603	6146.03
4	1.17983	0.379901	369304	359	366556	3665.77
5	0.824973	0.840969	452478	317	447618	4476.32
6	0.452081	1.01662	473894	222	379115	3791.15
7	0.585979	1.14312	956476	398	942031	9420.17
8	1.11577	0.203031	596784	359	592185	5921.63
9	1.04545	0.492173	987064	292	978739	9786.38

Table A-1: Source parameters used in experiment to validate approach to determine effective bandwidth.

The effective bandwidths were chosen to satisfy a 2% loss requirement and a delay requirement of no more than 50ms. As discussed above, the approach used to determine effective bandwidths is somewhat conservative and the loss and delay obtained when the system was simulated were considerably lower than the target loss and delay parameters. The experiment was run once: 3000 seconds of simulation time were simulated. The overall buffer loss was 0.052% in this experiment. Clearly, this is much lower than the target loss rate of 2%.

The confidence interval on the loss rate is difficult to determine accurately. However, other simulations were performed that produced similar results. This coupled with the fact that the difference between the loss rates – target and simulated – is so substantial, can be used to argue that the loss rate obtained in any real situation is indeed substantially lower than the target loss rate[20].

Observe that in a number of cases, the effective bandwidth is quite close to the peak rate, but in other cases, the effective bandwidth is substantially lower. These lower effective bandwidths, when aggregated, will result in a substantially lower resource allocation than peak rate allocation. Consequently, even though the approach used here is quite conservative, savings can still be made.

The end-to-end delay distribution as perceived by the foreground source is shown in Figure A-2. Most (over 88%) of the packets are not subject to a delay longer than 5.4ms. The remainder are subject to varying delays with a maximum delay of 50ms. The variation in the delays arises from differences in the queuing delay experienced by the packets.

The minimum delay is comprised of transmission delays and propagation delays. The propagation delays on each of the three links is 0.1ms, totalling 0.3ms overall. The transmission delays are substantially greater. In this experiment, the packet size for the foreground traffic is 333 bytes. The first link is a 1Mb/s link, resulting in a transmission delay for a 333-byte packet of 2.664ms. The second link has the aggregate effective

[20] Some initial investigations were performed to obtain confidence intervals on this result. The work of [Raa95] was studied, but it differs slightly from the case here. His emphasis is on obtaining confidence intervals for different parameters – specifically the queue occupancy parameters. His work is not entirely unrelated to the problem of obtaining confidence intervals here. However, the measured loss and delay are sufficiently far from the target loss and delay that it is highly unlikely that the simulation results would not meet the loss and delay targets.

bandwidth capacity – 4.53Mb/s in this case – and incurs a transmission delay of 0.588ms. The third link in the connection also has this capacity and incurs the same delay. The total propagation and transmission delays are then 4.14ms.

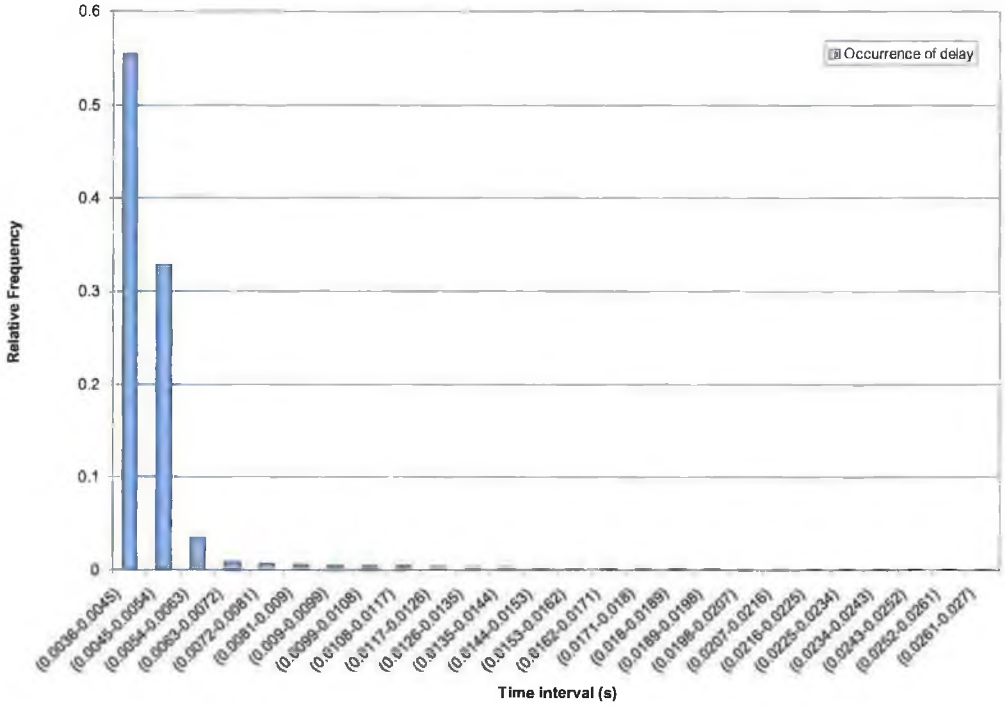


Figure A-2: End-to-end delay distribution as perceived by foreground traffic source in experiment.

In the results obtained many of the packets were delivered with a latency of 4.14ms. These packets experienced no queuing: the link onto which all the traffic is being multiplexed was free when these packets arrived at the multiplexer. Those packets delivered with a latency of greater than 4.14ms experienced some queuing.

Both the loss and the delay are much smaller than the target rates: it is clear that the approach used to determine the effective bandwidth is conservative. However, the objective here is not to devise a highly efficient approach to determine an effective bandwidth; rather it is to use **some** approach to determine an effective bandwidth in the generic network design approach to enable some solution to this problem to be obtained. Clearly, this works for sources multiplexed through a single stage.

Extending this to the case in which there are multiple multiplexing points is also possible. The previous example was extended to five multiplexing points. In this case, the foreground traffic was mixed with interfering traffic at a number of multiplexing points. This interfering traffic was then routed off the path of the foreground traffic and more background traffic was introduced to be multiplexed with the foreground traffic. The scenario is illustrated in Figure A-3.

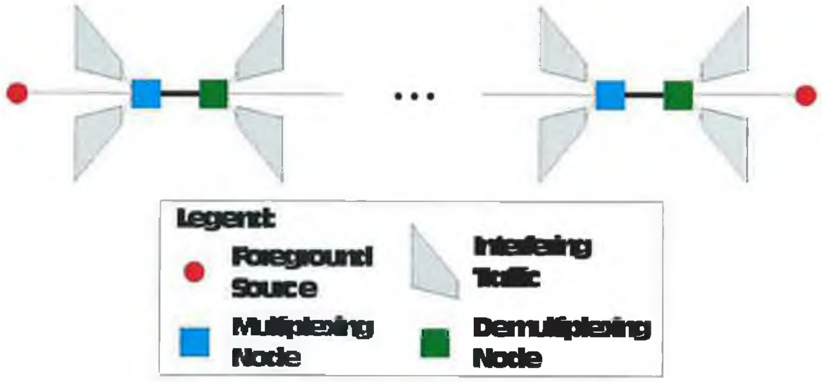


Figure A-3: Abstract topology used in more complex scenario used to validate approach to determine effective bandwidth. In this case there are more multiplexing stages. Five multiplexing stages were used in the experiments.

Source	On-time (s)	Off-time (s)	Peak Rate (b/s)	Packet Size (bytes)	Effective Bandwidth (b/s)	Buffer Size (bits)
0	0.575758	1.12453	515002	333	507090	5072.02
1	0.482431	1.17507	261687	375	209350	2093.5
2	0.291242	0.622087	768253	217	614603	6146.03
3	1.17983	0.379901	369304	359	366556	3665.77
4	0.824973	0.840969	452478	317	447618	4476.32
5	0.452081	1.01662	473894	222	379115	3791.15
6	0.585979	1.14312	956476	398	942031	9420.17
7	1.11577	0.203031	596784	359	592185	5921.63
8	1.04545	0.492173	987064	292	978739	9786.38
9	0.821026	0.869587	409329	465	404908	4049.07
10	0.465589	0.367046	503246	316	493872	4937.74
11	0.872332	1.02891	702712	359	695555	6954.35
12	0.651232	0.917737	495386	483	488659	4886.47

13	0.541648	1.23828	255571	354	204457	2044.57
14	0.427325	0.226729	693966	261	680110	6800.54
15	1.15323	0.265306	445632	290	442275	4422.61
16	1.16979	0.748018	782519	285	776579	7764.89
17	1.13791	0.943095	593867	487	589224	5892.33
18	1.02961	0.614206	421479	306	417859	4178.47
19	0.460382	0.936742	789225	369	631380	6313.8
20	1.08936	0.629673	864075	453	857050	8570.56
21	0.316561	1.23774	568665	356	454932	4549.32
22	0.178773	1.20486	911938	428	547163	5471.63
23	0.578707	0.957083	821161	321	808623	8087.16
24	0.250301	0.733097	416847	461	333478	3334.78
25	0.734586	0.201555	895385	205	884945	8848.88
26	1.07787	0.467172	485929	489	481955	4820.56
27	1.22267	1.09037	619082	266	614568	6146.24
28	0.791624	0.517932	490800	278	485346	4852.29
29	0.308785	0.201999	283588	272	275851	2757.57
30	0.598541	0.971787	416384	351	410233	4102.78
31	0.180436	0.63237	609607	265	487686	4876.86
32	0.363414	1.20782	512666	257	410133	4101.33
33	0.353482	0.89705	290174	435	232139	2321.39
34	0.136809	0.688506	382310	479	229386	2293.86
35	1.19158	0.720129	903772	408	897038	8970.95
36	1.21623	0.7006	548925	467	544921	5448
37	0.932256	1.12261	804774	395	797093	7969.97
38	0.913785	0.834934	454839	480	450423	4503.17

39	0.894327	0.252154	529779	385	524650	5245.36
40	0.278354	0.615036	733444	207	586755	5867.55

Table A-2: Source parameters for multistage experiment.

Multiplexing Stage	Loss Probability
1	0.076%
2	0.052%
3	0.016%
4	0.044%
5	0.060%

Table A-3: Loss probabilities at each multiplexing stage.

The parameters of all the sources and the effective bandwidths and buffer sizes are shown in Table A-2. As before, the simulation was run for 3000 simulation seconds. The amount of packet loss at each buffer was again very small. The packet losses per buffer are shown in Table A-3. The aggregate packet loss for the flow was 0.39% which, again, is very substantially less than the target value of 2%.

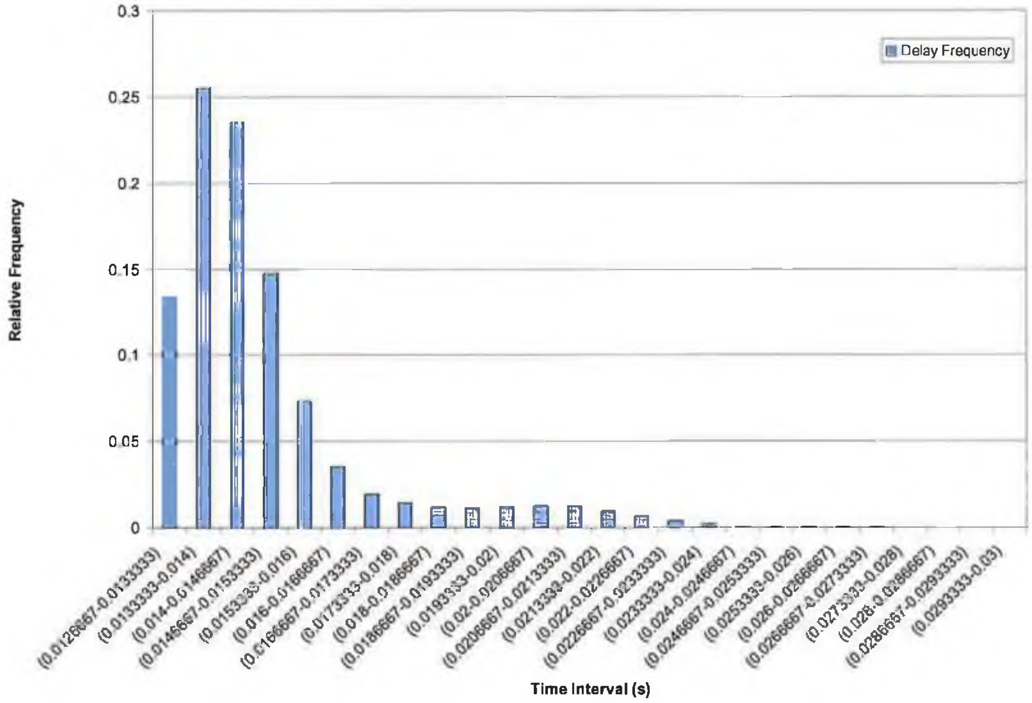


Figure A-4: Frequency of delays in multistage effective bandwidth experiment.

The end-to-end delay perceived by the end station is shown in Figure A-4. Over 87% of the packets are delayed between 12-16ms, which is still substantially lower than the target delay. However, it is worth noting that the delay is increasing and that ultimately, the delays may be comparable to the target delays. The considerable increase in delay over the single stage case can be accounted for by the increased number of transmission delays and propagation delays: the queuing delays are still very small, although they are greater than they were in the single stage case.

The above experiments illustrate that the effective bandwidth approach chosen can deliver the required loss and delay in the case in which the source characteristics are known. The AF traffic class, however, usually permits the customers to exceed their traffic contract and will attempt to carry this excess traffic if the traffic class is not congested[21]. Consequently, the total traffic in the network can, in extreme cases, greatly exceed the contracted traffic. In this case, it is difficult to assure that the target operating conditions are met.

The AF traffic class does permit differentiation of packets into different drop precedences. A well configured diffserv network supporting AF traffic classes should be able to carry traffic conforming to the traffic contract as higher priority than non-conformant traffic. The lower priority traffic should then be dropped in case of congestion. Thus, while the AF class may be congested, it can assure delivery of traffic up to the traffic contract: traffic in excess of the contract receives no such assurances. The delays can also be assured: the buffer sizes are chosen on the premise that there is no differentiation of traffic. As such, the queue sizes are chosen such that even if the queues are almost always full, the packets will be delivered in no more than the required delay. If this traffic differentiation does exist, and the queues are configured such that the low priority traffic is only delivered when queues are uncongested, then it should have little impact on the high priority traffic and hence the delays can also be assured.

[21] This is an AF configuration issue: it can be operated to work in a different fashion, but this is the most likely mode of operation.

APPENDIX B SCHEDULER ALGORITHM USED IN CHAPTER 5

A particular scheduler is used in Chapter 5. There, the details of the scheduler are omitted since they are not central to the discussion. However, the scheduler implementation used is quite interesting and is included here for completeness.

This scheduler determines which of a number of queues should have access to a link. In this case, the queues are the queues for the diffserv traffic: specifically, the scheduler serves 4 queues – an EF queue, an AF1 queue, an AF2 queue and a BE queue. The scheduler controls how much of the link resources are allocated to each of the above queues.

The scheduler used is a variant of a WRR scheduler. Typically, this type of scheduler operates on a round basis – in each round, it serves each queue a certain number of times. The scheduler implemented here is slightly different: in each round, each queue is allocated some amount of 'credit' which is proportional to its weight. As the queue is served, the credit is used up. This is more flexible than the less adaptive system which simply serves each queue a certain number of times.

The scheduler iterates through the queues in the system checking if they have packets ready for transmission and credit available to transmit the packets. If queue i has just been served, this iteration process starts at queue $i+1$. This makes it unlikely that a single queue will be served twice in succession. Using this approach, the intervals between serving a particular queue are not very long.

The fraction of the link resources allocated to each queue is dependent on the weight associated with the queue. The link resources are divided in proportion to the weights. For example, if the weights were 1:2:3:5, then $1/11$ of the link resources would be allocated to the first queue, $2/11$ of the link resources to the second queue, etc.

The scheduler used here operates on a packet-by-packet basis: the scheduler chooses a particular queue to serve, removes a packet from the queue, transmits it on the link and then determines which queue to serve next. This is in contrast to the way most realistic schedulers operate: each time they serve a queue, a number of packets are typically removed. However, for the purposes of this work, the scheduler operating on a packet-by-packet basis is reasonable.

The scheduler used here has some more interesting details. The next round is automatically initiated once the aggregate credit – the sum of the credit available to all the queues – drops below some limit. If only one queue has packets ready for transmission and has no credit available, the next round is started early. This is to ensure that the utilisation of the link remains high.

Starting rounds early in this fashion can cause credit to build up for queues that do not use their available credit over some period of time. Ultimately, this can lead to large accumulations of credit for particular queues. This is undesirable because it means that the previously dormant queue can start generating traffic and dominate access to the link for a prolonged period, during which time the other queues will not get access to the resource.

The above undesirable situation is avoided by imposing a limit on the amount of credit that can accumulate for a particular queue. This is done by testing the amount of credit allocated to each queue when increasing the available credit because the next round is starting. If the increase causes the available credit to exceed the limit, the available credit is reduced to the limit.

Limiting the credit available to any queue in this fashion means that the scheduler may not divide the available link capacity in exactly the proportions of the specified weights over the long term. If a particular queue is not being used, then the credit will accumulate up to the limit. Once the limit is reached, the queue starts to lose capacity that it could have used. Consequently, the proportions specified by the weights no longer reflect the usage of the link.

So, while the scheduler resources may not be divided according to the weights over the long term, in periods of congestion, the scheduler resources are divided in the appropriate proportions. This is a critical distinction. It is only in congested periods that the operation of the scheduler really matters. In uncongested periods, the capacity of the resource will exceed the demand, and the QoS perceived by all the users will be excellent. Outside of these times, the difference in QoS delivered by each service class can be observed.

The particular algorithm used for the scheduler in the simulations described in chapter 5 is listed in Algorithm 3.

DEFINE PROCEDURE:

```
Procedure StartNextRound:
    AggregateBytes=AggregateBytes+BytesPerRound
    For i=1 to NumberOfQueues
        Bytes(i)=Bytes(i)+Weights(i)*AggregateBytes
        If(Bytes(i)>Threshold(i))
            Bytes(i)=Threshold(i)
EndProcedure
```

STEP 1: INITIALISATION

```
    Set BytesPerRound,MinimumAggregateBytes,AggregateBytes
    For i=1 to NumberOfQueues
        Set Threshold(i),Weights(i)
        Set Bytes(i)=0
    Set CurrentQ=0
    Call StartNextRound
```

STEP 2: FIND QUEUE TO SERVICE

```
    Set QueueToServe=-1
    For j=CurrentQ to CurrentQ+NumberOfQueues
        Set PacketSize=SizeOfNextPacket(j)
        If ((PacketSize>0) and (Bytes(i) > PacketSize))
            Set QueueToServe=j
            Break
    If (QueueToServe!=-1)
        Goto 3
    Call StartNextRound
    For j=CurrentQ to CurrentQ+NumberOfQueues
        Set PacketSize=SizeOfNextPacket(j)
        If ((PacketSize>0) and (Bytes(i) > PacketSize))
            Set QueueToServe=j
            Break
    If (QueueToServe=-1)
        Goto 4
```

```
STEP 3: SERVICE QUEUE, REDUCE COUNTERS APPROPRIATELY
    Set AggregateBytes=AggregatesBytes-PacketSize
    Set Bytes(CurrentQ)=Bytes(CurrentQ)-PacketSize
    If (AggregateBytes<MinimumAggregateBytes)
        Call StartNextRound
    Goto 2
STEP 4: WAIT FOR PACKET
    Sleep
```

Algorithm 3: Scheduling algorithm used in the diffserv queue/scheduler systems described in Chapter 5.

References

[ABG01] D. Awduche, L. Berger, D.-H. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," IETF Internet Draft, February 2001, Work in progress.

[AC00] G. Ahn and W. Chun, "Design and Implementation of MPLS Network Simulator Supporting LDP and CR-LDP," in *Proceedings of IEEE International Conference on Networks 2000 (ICON2000)*, IEEE, 2000.

[ACFH91] G. R. Ash, J.-S. Chen, A. E. Frey and B. D. Huang, "Real Time Network Routing in a Dynamic Class-of-Service Network," in *Proceedings of the 13th International Teletraffic Congress*, Elsevier, 1991.

[ACL94] G. R. Ash, K.K. Chan, J.-F. Labourdette, "Analysis and Design of Fully Shared Networks," In *Proceedings of the 14th International Teletraffic Congress*, Elsevier, 1994.

[ACM81] G. R. Ash, R. H. Cardwell and R. P. Murray, "Design and Optimisation of Networks with Dynamic Routing," *Bell System Technical Journal*, vol. 60, pp. 1787-1820, 1981.

[AggRSVP] IETF Internet Draft draft-ietf-issll-rsvp-aggr-02.txt, "Aggregation of RSVP for IPv4 and IPv6 Reservations," March 2000.

[AH93] G. R. Ash and B. D. Huang, "An Analytic Model for Adaptive Routing Networks," *IEEE Transactions on Communications*, November 1993.

[AKK81] G. R. Ash, A. H. Kafker and K. R. Krishnan, "Servicing and Real-Time Control of Networks with Dynamic Routing," *Bell System Technical Journal*, vol. 60, no. 8, October 1981.

[Alm00] K. C. Almeroth, "The Evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment," *IEEE Network*, January/February 2000.

[AMS82] D. Anick, D. Mitra and M. M. Sondhi, "Stochastic Theory of a Data Handling System with Multiple Sources," *Bell System Technical Journal*, vol. 61, no. 8, pp 1871-1894, 1982.

[Arm00] G. Armitage, “MPLS: The Magic Behind the Myths,” *IEEE Communications Magazine*, January 2000.

[Arv94a] A. Arvidsson, “Management of Reconfigurable Virtual Path Networks,” in *Proceedings of the 14th International Teletraffic Congress*, Elsevier, 1994.

[Arv94b] A. Arvidsson, “Real Time Management of Virtual Paths,” in *Proceedings of Globecom '94*, IEEE, 1994.

[AS90] G. Ash and S. D. Schwartz, “Traffic Control Architectures for Integrated Broadband Networks,” *International Journal of Digital and Analog Communications Systems*, vol. 3, pp. 167-176, 1990.

[Ash85] G. R. Ash, “Use of a Trunk Status Map for Real-Time DNHR,” in *Proceedings of the 11th International Teletraffic Congress*, Elsevier, 1985.

[Ash95] G. R. Ash, “Dynamic Network Evolution, with Examples from AT&T's Evolving Dynamic Network,” *IEEE Communications Magazine*, July 1995.

[BBC01] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney and J.-Y. LeBoudec, “Delay Jitter Bounds and Packet Scale Rate Guarantees for Expedited Forwarding,” in *Proceedings of IEEE Infocom 2001*, IEEE, 2001.

[BCDM95] D .D. Botvich, T. Curran, N. G. Duffield, S. Murphy, “Allocating bandwidth from traffic descriptors,” in *Proceedings of the 3rd IFIP workshop on Performance Modelling and Evaluation of ATM Networks*, Bradford, UK, July 1995.

[BCMM94] D. D. Botvich, T. Curran, A. MacFhearraigh and S. Murphy, “Hierarchical Approach to Video Source Modelling,” in *Proceedings of the 15th UK Teletraffic Symposium,* Cambridge, UK, May 1994.

[BFC93] A. Ballardie, P. Francis, J. Crowcroft, “Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing,” in *Proceedings of ACM SIGCOMM 1993*, San Francisco, August 1993.

[BG87] D. Bertsakas and R. Gallagher, *Data Networks*, Prentice-Hall, 1987.

[BGZ94] N. G. Bean, R. J. Gibbens and S. Zachary, “The Performance of Single Resource Loss Systems in Multiservice Networks,” in *Proceedings of the 14th International Teletraffic Congress*, Elsevier,1994.

[BJS00] L. Breslau, S. Jamin, S. Shenker, "Comments on the Performance of Measurement-Based Admission Control Algorithms,'" in *Proceedings of IEEE Infocom 2000*, IEEE, 2000.

[BMM98] A. Balakrishnan, T. L. Magnanti and P. Mirchandani, "Designing Hierarchical Survivable Networks," *Operations Research*, vol. 46, no. 1, January/February 1998.

[CF98] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, August 1998.

[CFW94] C. Courcoubetis, G. Fouskas and R. Weber, "On the Performance of an Effective Bandwidth Formula," in *Proceedings of the 14th International Teletraffic Congress*, Elsevier, 1994.

[CFZ94] I. Chlamtac, A. Farago and T. Zhang, "Optimizing the System of Virtual Paths," *IEEE/ACM Transactions on Networking*, December 1994.

[CGH81] F. R. K. Chung, R. L. Graham and F. K. Hwang, "Efficient Realization Techniques for Network Flow Patterns," *Bell System Technical Journal*, vol. 60, pp. 1771-1786, 1981.

[CL94] K.-T. Cheung and F. Y.-S. Lin, "On the Joint Virtual Path Assignment and Virtual Circuit Routing Problem in ATM Networks," in *Proceedings of IEEE Globecom '94*, 1994.

[CLW94] G. L. Choudhury, D. M. Lucantoni and W. Whitt, "On the Effectiveness of Effective Bandwidths for Admission Control in ATM Networks," in *Proceedings of the 14th International. Teletraffic Congress,* Elsevier, 1994

[CMW89] R. H. Cardwell, C. L. Monma, T.-H. Wu, "Computer-Aided Design Procedures for Survivable Fiber Optic Networks," *IEEE Journal on Selected Areas in Communications*, October 1989.

[CN98] S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network*, Novermber/December 1998.

[Dee91] S. Deering, "Multicast Routing in a Datagram Internetwork," PhD Thesis, Stanford University, 1991.

[DEF94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Lue, L. Wei, “An Architecture for Wide-Area Multicast Routing,” in *Proceedings of ACM SIGCOMM 1994*, 1994.

[DGG99] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan and J. E. van der Merwe, "A Flexible Model for Resource Management in Virtual Private Networks," in *Proceedings of ACM SIGCOMM '99*, 1999.

[DH98] B. Doshi and P. Havardshana, “Broadband Network Infrastructure of the Future: Roles of Network Design Tools in Technology Deployment Strategies,” *IEEE Communications Magazine*, May 1998.

[EM93] A. Elwalid and D. Mitra, “Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks,” *IEEE/ACM Transactions on Networking*, June 1993.

[FJ93] S. Floyd and V. Jacobson, “Random Early Detection gateways for Congestion Avoidance,” *IEEE/ACM Transactions on Networking*, August 1993.

[FKSS99] W.-c. Feng, D. Kandlur, D. Saha and K. Shin, “A Self-Configuring RED Gateway,” in *Proceedings of IEEE Infocom ‘99*, IEEE, 1999.

[FT00] B. Fortz and M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights,” in *Proceedings of IEEE Infocom 2000*, IEEE, 2000.

[FV01] K. Fall and K. Varadhan (eds.), “The ns Manual,” April 2001. Available from <http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf>

[GAN91] R. Guerin, H. Ahmadi, M. Nagshindeh, “Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks,” *IEEE Journal on Selected Areas in Communications*, September 1991.

[GB89] A, Girard and M.-A. Bell, “Blocking Evaluation for Networks with Residual Capacity Adaptive Routing,” *IEEE Transactions on Communications*, December 1989.

[Gir90] A. Girard, *Routing and Dimensioning in Circuit-switched Networks*, Addison-Wesley, 1990.

[Gir93] A. Girard, “Revenue Optimization of Telecommunication Networks,” *IEEE Transactions on Communications*, April 1993.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[GK77] M. Gerla and L. Kleinrock, "On the Topological Design of Distributed Computer Networks," *IEEE Transactions on Communications*, January 1977.

[GKW90] G. Gopal, C.-k. Kim and A. Weinrib, "Dynamic Network Configuration Management," in *Proceedings of IEEE ICC '90*, IEEE, 1990.

[GKW91] G. Gopal, C.-k. Kim and A. Weinrib, "Algorithms for Reconfigurable Networks," in *Proceedings of the 13th International Teletraffic Congress*, Elsevier, 1991.

[GL93] A. Girard and B. Liau, "Dimensioning of Adaptively Routed Networks," *IEEE/ACM Transactions on Networking*, August 1993.

[GM84] M. Gondran and M. Minoux, *Graphs and Algorithms*, Wiley Interscience Series in Discrete Mathematics, Wiley, 1984.

[GMP89] M. Gerla, J. A. S. Monteiro and R. Pazos, "Topology Design and Bandwidth Allocation in ATM Nets," *IEEE Journal on Selected Areas in Communications*, October 1989.

[GN89] B. Gavish and I. Neuman, "A System for Routing and Capacity Assignment in Computer Communications Networks," *IEEE Transactions on Communications*, April 1989.

[GW90] A. Gersht and R. Weihmayer, "Joint Optimization of Data Network Design and Facility Selection," *IEEE Journal on Selected Areas in Communications*, December 1990.

[HB94] M. Herzberg and S. J. Bye, "An Optimal Spare-Capacity Assignment Model for Survivable Networks with Hop Limits," in *Proceedings of IEEE Globecom '94*, IEEE Press, 1994.

[HBU95] M. Herzberg, S. J. Bye and A. Utano, "The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks," *IEEE/ACM Transactions on Networking*, December 1995.

[Hui88] J. Y. Hui, "Resource Allocation for Broadband Networks," *IEEE Journal on Selected Areas in Communications*, December 1988.

[Hui95] C. Huitema, *Routing in the Internet*, Prentice Hall, 1995.

[IMG98] R. R. Iraschko, M. H. MacGregor and W. D. Grover, "Optimal Capacity Placement for Path Restoration in STM of ATM-Mesh Survivable Networks," *IEEE/ACM Transactions on Networking*, June 1998.

[Jam01] B. Jamoussi, (ed.) "Constraint-Based LSP Setup using LDP", IETF Internet Draft, February 2001, Work in Progress.

[Kel00] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society A358*, 2000.

[Kel96] F. P. Kelly, "Notes on effective bandwidths," in: *Stochastic Networks: Theory and Applications*, Eds. F. P. Kelly, S. Zachary and I. Ziedens, Royal Statistical Society Lecture Note Series vol. 4, 1996.

[Ker93] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill Series in Computer Science, McGraw-Hill, 1993.

[KGS96] S. Kheradpir, A Gersht and A. Shulman, "Dynamic Bandwidth Allocation and Path Restoration in SONET Self-Healing Networks," *IEEE Transactions on Reliability*, June 1996.

[KGV83] S. Kirkpatrick, C. D. Gellat and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, 1983.

[KKG91] A. Kershenbaum, P. Kermani and G. A. Grover, "Mentor: An Algorithm for Mesh Network Topological Optimization and Routing", *IEEE Transactions on Communications*, April 1991.

[Kle76] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, Wiley, 1976.

[KLT98] H. Kim, W. E. Leland, S. E. Thomson, "Evaluation of Bandwidth Assuarance Service using RED for Internet Service Differentiation," preprint, 1998.

[KS99] E. W. Knightly and N. B. Shroff, "Admission Control for Statistical QoS: Theory and Practice," *IEEE Network*, March/April 1999.

[Lin94] K. Lindberger, "Dimensioning and Design Methods for Integrated ATM Networks," in *Proceedings of the 14th International Teletraffic Congress*, Elsevier, 1994.

[Llo96] R. Lloyd-Evans, *Wide Area Network Performance and Optimization*, Addison-Wesley, 1996.

[LM97] D. Lin, R. Morris, "Dynamics of random early detection," in *Proceedings of ACM SIGCOMM '97*, 1997.

[LWD01] F. LeFaucheur, L. Wu, B. Davie, S. Dhavari, P. Vaananen, R. Krishnan, P. Cheval, J. Heinanen, "MPLS Support of Differentiated Services" IETF Internet Draft, April 2001, Work in Progress.

[LY91] M.-J. Lee and J. R. Yee, "A Partial Branch and Bound Design Algorithm for Reconfigurable Networks," in *Proceedings of ICC '91*, IEEE, 1991.

[LY95] M.-J. Lee and J. R. Yee, "A Logical Topology and Discrete Capacity Assignment Algorithm for Reconfigurable Networks," *Operations Research*, January/February 1995.

[MBC00] S. Murphy, D. Botvich, T. Curran, "On design of diffserv/MPLS networks to support VPNs," in *Proceedings of the 16th UK Teletraffic Symposium*, May 2000.

[MBC94] J. McGibney, D. D. Botvich and T. Curran, "Modern Global Optimization Heuristics in the Long Term Planning of Networks," in *Proceedings of ATNAC '94*, 1994.

[MBC98] S. Murphy, D. D. Botvich and T. Curran, "Cost comparisons of private voice network solutions," in *Proceedings of the 15th UK Teletraffic Symposium*, Durham, UK, March 1998.

[MBC99] S. Murphy, D. D. Botvich and T. Curran, "Traffic Partitioning using Algorithmic Mechanisms for Cost Analysis," in *Proceedings of the 16th International Teletraffic Congress*, July 1999.

[McG95] J. McGibney, *Modern Global Optimization Heuristics in the Long Term Planning of Telecommunications Networks*, Master's Thesis, School of Electronic Engineering, Dublin City University, 1995.

[Med94] D. Medhi, "A Unified Approach to Network Survivability for Teletraffic Networks: Models, Algorithms and Analysis," *IEEE Transactions on Communications*, February/March/April, 1994.

[Med95] D. Medhi, ``Multi-Hour, Multi-Traffic Class Network Design for Virtual Path-based Dynamically Reconfigurable Wide-Area ATM Networks," *IEEE/ACM Trans. on Networking*, December 1995.

[Med97] D. Medhi, ``Models for Network Design, Servicing and Monitoring of ATM Networks based on the Virtual Path Concept," *Computer Networks and ISDN Systems*, Vol. 29, No. 3, 1997.

[MG97] D. Medhi and S. Guptan, "Network Dimensioning and Performance of Multi-service, Multi-rate Loss Networks with Dynamic Routing," *IEEE/ACM Transactions on Networking*, December 1997.

[Min92] D. Minoli, *Enterprise Networking: Fractional T1 to SONET, Frame Relay to BISDN*, Artech House, 1992.

[Min93] D. Minoli, *Broadband Network Analysis and Design*, Artech House, 1993.

[Obr98] K. Obraczka, "Multicast Transport Protocols: A Survey and Taxonomy," *IEEE Communications Magazine*, January 1998.

[OMPID] IETF Internet Draft draft-ietf-ospf-omp-02.txt, "OSPF Optimized Multipath (OSPF-OMP)", February 1999.

[PF97] V. Paxson and S. Floyd, "Why We Don't Know How To Simulate The Internet," to appear in *IEEE/ACM Transactions on Networking*.

[Raa95] K. Raatikainen, "Simulation-Based Estimation of Proportions," *Management Science*, July 1995.

[Ree95] C. R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Advanced Topics in Computer Science Series, McGraw-Hill, 1995.

[RFC1142] IETF RFC 1142, "OSI IS-IS Intra-domain Routing Protocol," February 1990.

[RFC1584] IETF RFC 1584, "Multicast Extensions to OSPF," March 1994.

[RFC1633] IETF RFC 1633, "Integrated Services in the Internet Architecture: an Overview," June 1994.

[RFC1633] IETF RFC 1633, Integrated Services in the Internet Architecture: an Overview, June 1994.

[RFC1771] IETF RFC 1771, "A Border Gateway Protocol 4 (BGP-4)", March 1995.

[RFC2189] IETF RFC 2189, "Core Based Trees (CBT version 2) Multicast Routing," September 1997.

[RFC2205] IETF RFC 2205 "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," September 1997.

[RFC2236] IETF RFC 2236, "Internet Group Management Protocol, Version 2," November 1997.

[RFC2328] IETF RFC 2328, "OSPF Version 2," April 1998.

[RFC2362] IETF RFC 2362 "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," June 1998.

[RFC2453] IETF RFC 2453, "RIP Version 2," November 1998.

[RFC2474] IETF RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," December 1998.

[RFC2475] IETF RFC 2475, "An Architecture for Differentiated Services," December 1998.

[RFC2475] IETF RFC 2475, "An Architecture for Differentiated Services," December 1998.

[RFC2597] IETF RFC 2597, "Assured Forwarding PHB Group," June 1999.

[RFC2598] IETF RFC 2598, "An Expedited Forwarding PHB," June 1999.

[RFC2638] IETF RFC 2638, "A Two-bit Differentiated Services Architecture for the Internet," July 1999.

[RFC2661] IETF RFC 2661, "Layer Two Tunneling Protocol 'L2TP'," August 1999.

[RFC2698] IETF RFC 2698, "A Two Rate Three Color Marker," September 1999.

[RFC2859] IETF RFC 2859, "A Time Sliding Window Three Colour Marker (TSWTCM)," June 2000.

[RFC3031] IETF RFC 3031, "Multiprotocol Label Switching Architecture," January 2001.

[RFC3036] IETF RFC 3036, "LDP Specification," January 2001.

[RFC3107] IETF RFC 3107, "Carrying Label Information in BGP-4," May 2001.

[RFC793] IETF RFC 793, "Transmission Control Protocol," September 1981.

[RJSV98] M. Rohne, T. Jensen, I. Svinnset and R. Venturin, "Designing VP Networks," in *Proceedings of the 14th Nordic Teletraffic Seminar*, Copenhagen, Denmark, 1998.

[RKMK00] R. Rabbat, K. Laberteaux, N. Modi, J. Kenney, "Traffic Engineering Algorithms Using MPLS for Service Differentiation," in *Proceedings of ICC 2000*, IEEE, 2000.

[RMV96] J. Roberts, U. Mocci and J. Virtamo, (Eds.) *Broadband Network Teletraffic: Final Report of Action COST 242*, Springer, 1996.

[Ros95] K. Ross, *Multiservice Loss Models for Broadband Telecommunications Networks*, Springer, 1995.

[Sie94] R. Siebenhaar, "Optimized ATM Virtual Path Bandwidth Management under Fairness Constraints," in *Proceedings of IEEE Globecom '94*, IEEE, 1994.

[SRC84] J. H. Saltzer, D. P. Reed and D. D. Clark, "End-To-End Arguments in System Design," *ACM Transactions in Computer Systems*, November, 1984.

[Sys86] R. Syski, *Introduction to congestion theory in telephone systems*, North-Holland, 1986.

[TL00] A. Tomaszewski and J. Lubacz, "A Network Planning Model," First Polish-German Teletraffic Symposium, Dresden, Sept. 2000.

[TMW97] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, November/December 1997.

[VHS96] P. A. Veitch, I. Hawker and D. G. Smith, "Administration of Restorable Virtual Path Mesh Networks," *IEEE Communications Magazine*, December 1996.

[Wu92] T.-H. Wu, *Fiber Network Service Survivability*, Artech House, 1992.

[Whi97] Paul P. White , "RSVP and Integrated Services in the Internet: A Tutorial," *IEEE Communications Magazine*, May 1997.

[WY90] E. W. M. Wong and T.-S. Yum, "Maximum Free Circuit Routing in Circuit-Switched Networks," in *Proceedings of IEEE Infocom '90*, IEEE, 1990.

[XN99] X. Xiao and L. M. Ni, "Internet QoS: the Big Picture," *IEEE Network*, March/April 1999.

[Yag71] B. Yaged, Jr., "Minimum Cost Routing for Static Network Models," Networks, vol. 1, pp. 139-172, 1971.

[YST99] H. Yoshimura, K.-I. Sato, and N. Takachio, "Future Photonic Transport Networks Based on WDM Technologies," *IEEE Communications Magazine*, February 1999.

PUBLICATIONS ARISING FROM THIS WORK

[MBC00] S. Murphy, D. Botvich, T. Curran, "On design of diffserv/MPLS networks to support VPNs," in *Proceedings of the 16th UK Teletraffic Symposium*, May 2000.

[MBC99] S. Murphy, D. D. Botvich and T. Curran, "Traffic Partitioning using Algorithmic Mechanisms for Cost Analysis," in *Proceedings of the 16th International Teletraffic Congress*, July 1999.

[MBC98] S. Murphy, D. D. Botvich and T. Curran, "Cost comparisons of private voice network solutions," in *Proceedings of the 15th UK Teletraffic Symposium*, Durham, UK, March 1998.

[BCDM95] D .D. Botvich, T. Curran, N. G. Duffield, S. Murphy, "Allocating bandwidth from traffic descriptors," in *Proceedings of the 3rd IFIP workshop on Performance Modelling and Evaluation of ATM Networks*, Bradford, UK, July 1995.

[BCMM94] D. D. Botvich, T. Curran, A. MacFhearraigh and S. Murphy, "Hierarchical Approach to Video Source Modelling," in *Proceedings of the 15th UK Teletraffic Symposium,* Cambridge, UK, May 1994.