Congestion and Admission Control in WDM Optical Networks

by

Mohamed A. Ghaly©

Under the Supervision of Dr. Abdelhamid Tayebi
and Co-Supervision of Dr. A. Shami

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

Master of Science
in Control Engineering

Lakehead University, Thunder Bay, Ontario, Canada

January 2006

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-24055-7
Our file Notre référence
ISBN: 978-0-494-24055-7

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

The demand for more communication bandwidth and network resources, has pushed researchers to find faster and more reliable data communication networks. Wavelength division multiplexing (WDM) is a promising technology to meet such increasing demands. To make use of the WDM networks, some issues need to be dealt with. This thesis discusses three problems, constraint-based path selection, Congestion Control and Admission Control.

When selecting a path between the source and destination, in which some constraints are present, the choice of the path can have dramatic effects on the Quality of Service (QoS). Three path selection algorithms are compared in order to achieve optimum path selection. These algorithms are presented and analyzed in this thesis. The algorithms do not just deal with one path selection constraint but k-constraints.

Two controllers are presented: A proposed congestion controller and the second is a call admission controller in circuit switched networks. The proposed congestion control algorithm is based on the fuzzy logic technique and aims to control the congestion in a WDM network through an adequate adjustment of the delay on the calls that are in the queue of the server. The adaptive admission controller for circuit switched networks is based on the optimization of resources in the network. Numerous Simulation results are presented which show the performance of each controller.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgments

I gratefully acknowledge Dr. Abdelhamid Tayebi for his help in preparing and writing this thesis. I would like to thank him for the ideas that led to this work, for his timely comments, guidance and patience throughout the course of my masters program.

I would like to take this opportunity to thank my co-supervisor Dr. Abdallah Shami for his help in building the background needed for this research and putting my foot on the right step from the very beginning.

I would like to thank Miss D. Ibrahim for taking time out from her busy schedule, to check and review the thesis report several times for grammar accuracy and spelling mistakes.

Last, but definitely not least, my deepest gratitude to my family and friends, who have given me the love and emotional support throughout my stay in Thunder Bay.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

The demand for more communication bandwidth and network resources has pushed researchers to find faster and more reliable data communication networks. Wave division multiplexing is the promising technology to meet such increasing demands. An optical fiber has high bandwidth, but not all of its bandwidth can be used because of the limitations of the end users when accessing the network. Therfore, it is very difficult to use all of the single fiber bandwidth in a single high capacity wavelength channel because of the optical to electronic bandwidth mismatch. Two factors contribute to the development of the WDM optical fiber which are; the method of sending many light beams of different wavelengths simultaneously down the core of an optical fiber, and the erbium doped fiber amplifier which amplifies the signals with different wavelengths simultaneously, regardless of their modulation scheme or speed. Although WDM optical networks solve a lot of bandwidth problems, but when it comes to implementation other problems are discovered. This thesis looks at the most popular issues that face WDM technology, such as path selection subject to multiple constraints, and admission and congestion control schemes. Section 1.2 discusses some of the solutions presented in previous studies regarding these matters, although there is a lot of literature in these areas I chose only recent and most relative to my research.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.1 Thesis Objectives

WDM has been designed to support various classes of multimedia traffic with different bit rates and Quality of Service requirements. Due to the unpredictable fluctuations and burstiness of traffic flows within multimedia networks, congestion can occur frequently. Therefore, it is necessary to design appropriate congestion control mechanisms to ensure the promised quality of service is met. Congestion Control is a process which is used in networks to avoid congestion, which is desired since network resources are limited. If uncontrolled, users can easily overload certain networking resources thus making the network unusable. The complexity is increased since network traffic is a complex nonlinear and non stationary process which is significantly affected by immeasurable parameters and variables. Hence, a precise model of this process becomes increasingly difficult as the complexity of the process increases. This area has been the interest of many researchers. In the networking literature several congestion control mechanisms have been proposed.

WDM networks entails the reservation of limited resources (i.e., bandwidth) at each node along the path. If, upon arrival of a call, the desired resources are unavailable at any of the intermediate nodes, the call is said to be blocked. Blocked calls are assumed to be lost form the system, a mode of operation known as "blocked calls cleared" [1]. Common performance measures for this mode of operation include blocking probability and throughput. A fundamental issue arrises in networks supporting quality of service requirements is that of call admission. Call admission is the decision to accept or reject a new call. The need for admission control, even when network resources are available, is due to the fact that the acceptance of certain calls can have detrimental effects on the performance of currently active calls. The call admission problem has attracted considerable interest in recent years and is most often placed in the context of high-speed integrated services networks.

A network model has been designed using C++ to test the proposed congestion

[1] In some models, calls that are denied immediate access can be queued until network resources are available.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

control algorithm and the previously proposed call admission algorithm that was presented in (Gokbayrak and Cassandras 2002). The network model uses discrete event simulation, and the network is designed to follow the WDM network architecture. The network model consists of a traffic generator, a path selection, a wavelength selection algorithm and a controller. Therefore, it is worth mentioning how the wavelength selection and the path selection schemes are chosen and how performance is degraded if they are not chosen carefully. Three constraint-based path selection algorithms are examined. The algorithms examined are the Delay Cost Constrained Routing (DCCR), Least Delay Path (LDP) and the HZ_1 algorithm.

A control algorithm is proposed to control congestion in a WDM network which is based on the Fuzzy Logic approach. The fuzzy model used is a standard additive model. An adaptive online admission control is presented. The admission controller uses an online surrogate problem methodology to formulate the call admission control problem. Each controller is implemented in the network model and simulated to see how the blocking probability can be controlled using either the proposed congestion control algorithm or the adaptive online admission control algorithm(Gokbayrak and Cassandras 2002). The performance of each controller is presented through a series of simulation results.

1.2 Previous Related Work

1.2.1 Path Selection Subject to Multiple Constraints

In (Chen and Nahrstedt 1998) an algorithm is proposed to solve the Multi-Constrained path problem which uses a polynomial time complexity. The authors first reduce the NP-complete problem to a less complicated one, which can then be solved in polynomial time. The authors prove that the solution of the simplified problem is the same as the solution of the original problem. This can then be solved using an extended

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

version of Dijkstra's algorithm or by Bellman-Ford algorithm. The total time complexity of the algorithm, if the extended Dijkstra is used is $O(X^2V^2)$. However, if Bellman-Ford algorithm is used, then it is $O(XVE)$, where X is an integer defined inside the algorithm, V is the set of nodes, and E is the set of links. The value of X is chosen by the user such that, the higher the value of X, the bigger the chance of finding a satisfactory path and a higher overhead. The algorithm proposed in (Chen and Nahrstedt 1998), can give more accurate results, but at the expense of higher overhead.

The authors in (Korkmaz and Krunz 1999) try to solve the problem of finding a path between the desired source and destination which satisfies one or more constraints. They also consider selecting a path which has multiple additive constraints. The problem is then defined as a Multiple Constrained Path Selection. The algorithm presented consists of two main parts; it first filters out the links that are not on any feasible path, secondly, it then uses a randomized search to find a path that meets the requirements (if such a path does exist). The algorithm has the worst case computational complexity of $O(n^2)$, and a storage complexity of $O(n)$, where n is the number of nodes in the network. The algorithm presented has a slight chance of not finding a path that meets the required QoS, even though one might exist. The results of the randomized algorithm are close to the results to the optimal results. This algorithm has the potential of achieving better results at the expense of more computational costs. One of the advantages of this randomized algorithm is that it does not need to know the true state of the network at each node, but it achieves high performance when this information is available.

The solutions that have been proposed for the problem of finding a path under multiple constraints suffer either from excessive computational complexities and/or unacceptable performance. The authors in (Korkmaz and Krunz 2001) introduce an algorithm which solves the downfall of previous algorithms. In (Korkmaz and Krunz 2001), a nonlinear cost function $g\lambda$, is introduced which can be used as the basis for efficient heuristic solutions to the Multi-Constraint Optimization Problem (MCOP).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The proposed algorithm is called H-MCOP, this algorithm tries to approximate the minimization of $g\lambda$. The algorithm searches for a feasible path and also optimizes the use of available resources. The authors show through a series of simulations that H-MCOP provides the same performance as the predecessors and sometimes exceeds them. The main advantage of this algorithm, is that it has the same complexity as Dijkstra's shortest path algorithm. The authors also mention that the performance of the H-MCOP can also be increased if it is used with the k-shortest path algorithm. Another advantage that this algorithm has is that even when the constraints are negatively correlated, or even at no correlation, the algorithm still provides significant performance improvement over other presented algorithms. How the algorithm performs in the presence of an inaccurate state information and how it could be used in a distributed manner still need to be investigated.

In (Liu and Ramakrishnan 2001), an algorithm that solves the K-Multiple constraint shortest path (KMCSP) problem was presented. The algorithm can be applied to multiple constrained shortest path and k-shortest path problems. The algorithm's running time can be exponential in worst case scenarios, but still is comparable to the algorithms used in existing practical networks. The algorithm first makes a candidate list containing paths that go from the source to destination. The paths are then sorted by shortest paths. On the other hand, paths that definitely violate the constraints are removed from the list. Dijkstra algorithm can be used in performing both of these functions. The simulation results presented in (Liu and Ramakrishnan 2001), show that the results are comparable to the current best known polynomial time ϵ-approximation algorithms.

In (Yuan 2002), a multi-constrained routing is defined as finding a route between the desired source and destination that would satisfy multiple independent QoS constraints. This paper presents two heuristics; the first is called the limited granularity heuristic, the second is the limited path heuristic. These algorithms can be applied to the extended Bellman-Ford algorithm to obtain a solution to the k-constrained QoS routing problem. The limited granularity heuristic presented in (Yuan 2002)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

uses bounded ranges of integers in order to obtain an approximate solution in the polynomial time. In other words, the main idea of this algorithm is to simplify the original NP-hard problem. This algorithm is then used to solve the k-constrained problem by approximating $k-1$ QoS matrix with $k-1$ bounded ranges. The author shows that this algorithm maintains a table size of $O(|N|^{k-1})$ in each node to achieve high performance, where N is the number of nodes. Thus, the cost of this high performance is the time complexity of $O(|N|^2|lg(|N|))$ entries at each node. The author shows through a series of simulations that when $k > 3$, the limited path heuristic is more efficient than the limited granularity heuristic in solving general k-constrained QoS problems. Both algorithms can solve the multi-constrained problem when $k = 2$, however the advantage of the limited granularity heuristic is that it maintains a table size of $n^{k-1}N^{k-1}$, where n is a reasonably high constant. It also guarantees finding $(1-(1/n))$ approximate solutions, while the limited path heuristic can not provide such a guarantee.

A problem that is presented in (Guo and Matta 2003), is finding the least cost path, which is subject to some delay constraint in a network. This problem is defined as a Delay-Constrained Least Cost (DCLC) routing problem. The algorithm presented by Guo and Matta uses a nonlinear weight function and then applies a k-shortest algorithm, which makes the search for a path faster and more accurate. The authors want to increase the accuracy of the algorithm by using another DCLC heuristic. It can narrow down the search, consequently trading some extra execution time for a more accurate search. The authors show, by simulation, that even when the cost and delay are negatively correlated, the improved SSR (Search Space Reduction)+DCCR algorithm proposed always returns a feasible path, whose cost is very close to the optimum one. The optimal one is produced by using an extensive computational effort, which makes it undesirable in the real world.

It is worth mentioning that the algorithms mentioned in this section are used in the centralized approach.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.2.2 Admission and Congestion Control in Optical Networks

In (Haas 1991) a congestion control scheme is proposed. The algorithm is based on periodic exchange of sampling packets and on adaptive admission control. The scheme is designed for an end to end distributed operation. One of the advantages of this design is that no changes or additions are required within the subnet. An important feature of the proposed scheme is that it can cope with traffic surges that are short compared to the round trip delays.

The congestion control problem in high speed wormhole routing networks is tackled in (Leonardi *et al.* 1996), by comparing and evaluating two alternate approaches. First, the back pressure flow control, which is an explicit mechanism, and secondly, the deflection routing with host input rate control, which is an implicit mechanism. The problem with the back pressure flow control is that it has the potential for dead-locks, while the problem with the deflection routing is that it has the potential for live locks. It is shown in the paper that deflection routing with worm alignment provides throughputs which are relatively close to those obtained from back pressure with virtual channels. In terms of complexity of the node, they both are very close (in the case of the back pressure with virtual channels versus the case of deflection routing approach). The authors sacrifice low complexity in order to solve the dead lock problems related to back pressure by using restricted routing or using virtual channel technique. The simulations and results presented show that the restricted routing technique gives bad performance due to congestion on the most used network links near the root of the spanning tree. The memory requirement depends on the network size. On the other hand, the problem with deflection routing is the live lock problem and also poor performance, if the deflection process is completely asynchronous. The authors avoid live locks by routing several worms together and random choices avoid deterministic circular paths. In (Leonardi *et al.* 1996), the authors show that the performance does greatly improve due to the reduced deflection probability. The memory requirements of deflection routing are related to the size of the data units, but does not depend on the size of the network. The advantages of the de-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

flection routing are: simplicity of implementation, robustness against failures, can be implemented on any topology with few restrictions, and has the ability to implement more complex input control schemes based on the observed link load. However, when the traffic is unbalanced, as in the case of a single server and several circuits, the back pressure flow control mechanism using a virtual channel was shown to perform better than the deflection routing.

Congestion control in packet switching networks is tackled by Mascolo in (Mascolo 1999). Mascolo uses transfer functions to represent the system that is being controlled. The dynamic behavior of each network queue, in response to data input, is presented in a cascade of an integrator with a time delay. Propagation delays are emphasized, since they are very important in high speed communication networks. The Smith principle is chosen in order to design the congestion controller. The controller can be used over any path with any bandwidth delay product. The control law is applied to control the Available Bit Rate in Asynchronous Transfer Mode (ATM) networks and it compares its performance to other control laws that can be used. The mathematical analysis presented is, in a realistic network, which has different round trip times and shares available bandwidth with high priority traffic. The main advantages are the simplicity of the algorithm applied in the network, and the ability to examine the transient and steady state behaviors using mathematical analysis. Other advantages are the absence of overshoots and/or oscillations when converging from input rate to stationary rates and the fact that it is adaptive as it adapts to the changing traffic conditions.

In (Zhao and Jia 1999) a new admission control algorithm has been proposed. The main idea presented is the Adaptive Real Time Connection (ARTC), which is a generalized traditional real time connection. The QoS specifications are not given by exact values, but instead by regions in the QoS parameter space. Therefore, when a call is accepted the best possible QoS from the specified region can be offered. The algorithm presented has three main objectives. First, to increase the acceptance probability, secondly, to provide admitted calls with the best possible QoS in terms

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of available network resources, and thirdly, to maintain the run time of the algorithm at an acceptable level. The main advantage of this algorithm is that it can be applied in real world systems.

A robust adaptive congestion control algorithm is presented in (Imer *et al.* 2000), which requires only the knowledge of the maximum round trip network delay and the maximum number of simultaneous connections switched through the same output port. Imer, Basar and Srikant assume that the available service rate and the incoming call rate are controlled by one of the output lines of the switch. The authors show that if the value for the gain is chosen properly, the algorithm achieves max-min fairness along with queue length stability under Minimum Cell Rate (MCR) and Peak Cell Rate (PCR) constraints, and different number of sources and their round trip delays. The algorithm is also able to achieve high utilization of available bandwidth. The computational complexity of the algorithm is low, since there is only one single design parameter, β, to be tuned and the switch has to perform only two divisions; one multiplication and two additions per output line in order to determine the Explicit rate (ER). The main disadvantage of this algorithm is that if β is small, it results in a smaller overshoot, but a larger settling time.

In (Ma and Hamdi, 2000), a complete mechanism which include an admission control policy, a traffic regulator, and a scheduling algorithm has been presented. The scheduling algorithm is for the reservation medium access control protocol, in the single hop passive star coupled WDM optical network. This provides guaranteed deterministic performance service to the application streams, composed for real time variable length calls. A new admission control policy is presented, which is used to make decisions on which application streams can be admitted, when more than one new application stream requests to enter into the network. The scheduling algorithm is dedicated to scheduling the variable length messages in the specified WDM optical networks. A mathematical model has been formulated in order to evaluate the delay bound. The network service scheme has been validated through a series of simulations presented by the authors.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A two level bandwidth allocation and admission control strategy is presented by (Davoli and Maryni 2000), in the perspective of an access multiplexer to a multi-user, multi-service broad-band telecommunication network. The main idea is the addition of the neural network controller in a two level hierarchical scheme over an infinite time horizon. Since the controller has to adjust its parameters to varying traffic changes, it makes use of the neural approximations whose dynamics change.

In (Liu *et al.* 2000) a scheduling application for a WDM optical network is presented. In the design presented, agents are used to provide the enabling mechanism for monitoring and controlling the network equipment. (Liu *et al.* 2000) uses a resource broker to take care of the communication and interoperablilty issues between the agents and the application. In order to decouple the communication between agents and the scheduling application and to enable communication among the agents themselves, the authors designed an event service to handle this. The application presented is equipped with a wavelength scheduling algorithm to provide traffic control and efficient resource allocation.

In (Tan and Yang 2002), traffic controllers are presented and designed using the classical control theory and Schur-Cohn stability criterion. This provides the algorithm with the required stability condition under which the controlled switching network is stable, in terms of buffer occupancy. A class of end to end rate based congestion controllers are proposed, which meet the necessary requirements for the relevant stability condition. Furthermore, ideas are presented on how to choose the most desirable controller from this class to meet more specific performance requirements. The mathematical analysis and simulations presented show the validity of this approach. The drawback of the approach is that it does not guarantee the required QoS in real time communication networks.

The authors in (Zhou and Mouftah 2002) propose the Least Load Routing (LLR-k) algorithm and the Adaptive Least Load Routing (ALLR-K) algorithm. These algorithms try to improve the performance of the Least Load Routing (LLR) and the fixed path least-congestion (FPLC) algorithms. In the LLR, instead of blocking a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

call because there are no resources on the least loaded route, k number of routes are searched for an available wavelength. On the other hand, in FPLC, instead of fixing the alternate route, let some of the routes be fixed and others are dynamically generated according to the history of the traffic distribution. Simulations and numerical analysis presented show that LLR-k is much less complex than FPLC; LLR-k has higher performance than FPLC and is more robust in the sense that the value of k is independent of the network topologies.

The Weighted Sequential Greedy Scheduling (WSGS) protocol is proposed in (Smiljanic 2002). This approach is similar to the Weighted Probabilistic Iterative Matching (WPIM) protocol but instead of using the probabilistic iterative match, the protocol presented uses the sequential greedy scheduling instead. Thus, the WSGS implementation is further simplified in a comparison to the WPIM. Using this protocol makes it very simple to flexibly share bandwidth in switches with input buffering. The author showed that the proposed WSGS can reserve 50 % of the total switch capacity.

In (Boudriga and Obaidat, 2003) an initial look at how Advanced Reservation (AR) affects admission control, path selection, and resource reservation. The authors proposed an admission controller and resource management scheme for optical networks that deals with service uncertainty, helps optimize traffic engineering and reduces reservation costs. This method allows users to specify advanced reservation requests based on what they know about their quality of service. The authors also provided an efficient framework for optimized service utilization and continuity. They show through a set of simulations that their Multi-step Reservation Admission (MRA) scheme, outperforms the traditional schemes.

In (Kim *et al.* 2004), a Delay Buffer Control for Mixed Traffic (DBCM) is used for QoS of mixed traffic, from real-time (RTT) and non-real time traffic (NRT) sources in Networks. The authors measure the transfer delay at the node, which they have defined as the time in which the user transmits a packet, and the node takes this packet and forwards it to the output link. The transfer delay is maintained around a desired value, by dynamically adjusting the output bandwidth and also adjusting its

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

overshoot, accordingly. Fairness is considered in (Kim *et al.* 2004) when adjusting the bandwidth. The authors present packet loss in delivering NRT by measuring and controlling the buffer level in each node. The proposed algorithm is different than its predecessors in three main areas. First, this algorithm considers mixed traffic sources for both RTT and NRT. Secondly, the delay is measured and used as a control target, thus delay and buffer occupancy are controlled for RTT and NRT which corresponds to their different QoS requirements. Thirdly, the output rate of the node is adjusted as a control input, which improves the feedback delay. Buffer utilization is then improved by dynamic control and buffer control based on proper available bandwidth. Simulations show that DBCM performs well, in terms of punctuality rate and packet loss ratio, when compared to other algorithms. The downfall of the presented algorithm is that it does not consider diverse priorities of RTT and NRT sources.

The structure that is proposed by Kim, Park, and Ko in (Kim *et al.* 2004) is a system over communication networks, and a system over asymmetric path delay configurations (SOAP) on the high speed networks. The SOAP uses two different path sharing reliable transport protocol. One path has a constant delayed data to the destination by using its buffers, and the other path delivers the data with their delay information to the destination. In (Kim *et al.* 2004), two kinds of dynamic output-feedbacks are proposed: The Previous Mode (PM)-dependent and PM independent controllers. The mode represents one of the status that a switching controller can belong to. The PM dependent controller obtains its mode based on its previous mode, while the PM independent controller does not. The H_∞ norm minimization problem of the SOAP is based on two methods using a piecewise Lypanov function for the PM-dependent controller. For the PM-independent controller a common Lypanov function is used for the H_∞ norm minimization. The authors show, by the means of simulations, that the PM-dependent controller performs better than that of the PM-independent switching controller, for all the admissible mode transitions in the most uncertain communication network conditions.

In (Mosharaf *et al.* 2004) a call admission control policy is presented to provide

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

fairness control and service differentiation in WDM grooming networks. The problem is first formulated as a Markov Decision Process (MDP) and the optimal policy is obtained by the policy Iteration method. Based on the properties of the optimal policy, a decomposition algorithm is proposed for multi-link networks. The algorithm has lower computational complexity with very good performance. It can achieve substational improvement in terms of fairness ratio and utilization.

A class of congestion control algorithms are presented in (Deb and Srikant 2004) for networks with unicast and Multicast sessions. The congestion control mechanism can be implemented in a decentralized manner and with a simple one-bit marking scheme. The authors proved the stability of one particular congestion-control algorithm among the class of schemes that have been proposed. Fluid level simulation results were provided to show how maximum rates can be obtained using the algorithm. The scheme can be used efficiently in the presence of TCP sessions, and with a judicious choice of parameters, it can ensure fair long-run throughput among different TCP sessions and multicast receivers.

The authors in (Bruni *et al.* 2005) focused on the scheduling and congestion control problems for a band limited network. The authors took into account the QoS requirements of the various connections. Optimal control methodologies are used to tackle and solve the problem. The main features of this solution are the unitary context and implementation of the congestion and scheduling controls, and the feedback structure of the optimal solution, which secures a high level of robustness. In (Bruni *et al.* 2005), the authors assumed discretization of the time interval, which allowed the problem to become an instantaneous optimal control problem.

1.3 Thesis Organization

In order to control a system all system characteristics must be studied thoroughly. Chapter 2 discusses the optical fiber in detail, describes the makeup of a WDM optical network and how the wavelengths are assigned. Three constraint-based path selection

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

algorithms are examined in chapter 3. In chapter 4, the proposed congestion control algorithm is presented. The admission control algorithm (Gokbayrak and Cassandras 2002) is presented in chapter 5. Simulation results are presented separately in each chapter. Chapter 6 discusses the network model that was used for the simulations. Finally chapter 7 concludes the thesis and gives some directions for future work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

WDM Optical Communication Networks

The optical fiber has emerged for its ability to meet the continuous growing demand of data and voice traffic. The reason for that is because the optical fiber has potentially limitless capabilities: huge bandwidth, low signal attenuation and distortion, low power requirement, small space requirement and low cost (Murthy and Gurusamy 2001) and (Tanenbaum 1996).

In order to get the maximum bandwidth so that the users' needs can be met, which include: voice, video conferencing, etc., simultaneous multiple user transmissions are introduced into the network. This can be achieved in an optical network by either wavelength or frequency division multiplexing (WDM, FDM), by time division multiplexing (TDM), or by code division multiplexing (CDM).

Optical TDM and CDM are not appropriate for the network because the TDM bit rate and CDM chip rate would have to operate on a rate higher than the electronic rate. Since WDM has no such requirement, it is the most appealing to be used. The bit rate of a WDM channel can be chosen subjectively, so it is usually chosen at maximum electronic speed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.1 Wavelength-Division Multiplexing

Wavelength division multiplexing solves the problem of huge opto-electronic bandwidth mismatch[1], by requiring that each user equipment operates at the electronic rate. The optical transmission spectrum consists of non-overlapping frequency bands, so that each one is assigned to a single communication channel. The huge fiber bandwidth can be greatly utilized if multiple channels are allowed to co-exist on a single fiber. Another feature of using WDM, is that all WDM devices are easier to implement when working at electronic speeds. As a result, there is a great number of WDM devices available in the market today and they are on the rise.

Research and development on optical WDM networks grew rapidly in the past few years. The main area in which this has been tested and prototypes have been made, is in the telecommunication area.

2.2 WDM Networking Evolution

WDM networking evolution is divided into three main areas:- point to point WDM systems, wavelength add/drop multiplexers, and finally fiber and wavelength cross connects (Mukherjee 2000).

2.2.1 Point to Point WDM Systems

Telecommunication companies are now implementing the WDM technology. The reason behind this is, as more and more users and their demands increase, WDM is becoming the best solution in terms of cost-effectiveness compared to increasing the number of fibers (Murthy and Gurusamy 2001).

[1]The mismatch occurs because the wavelength capacity is 10Gbps today and increasing, while the sub-rate traffic varies from 51.84 Mbps

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

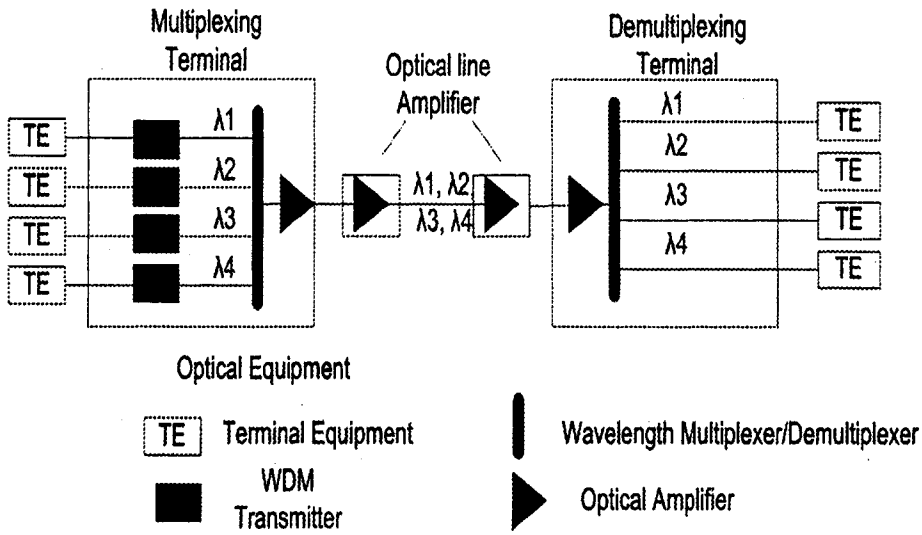

Figure 2.1: A Four channel point to point WDM transmission system with amplifiers

In order to upgrade the capacity of a transmission link there are three possible solutions:

- Add more fiber links
- Use an n-WDM solution (see figure 2.1)
- Use a higher electronic speed solution (*i.e.*, upgrade the hardware)

For short distances (50 Km or less), adding more fiber links seems to be the least cost solution, but for longer distances the WDM solution is the least costly solution, with higher electronic speeds not that far behind. As can be seen in figure 2.1, in a WDM transmission system, there are four wavelengths which are entered into a Multiplexing terminal in order to be put on a single Optical link. In order to separate the four wavelengths at the end of the fiber link, they have to pass through a De-multiplexing terminal.

The WDM Mux and DeMux in point to point links are now available from different manufacturers such as Pirelli, and AT&T. Among these products the maximum number of channels is 160, as of October 2005, using an OC-192 fiber. This fiber gives a bandwidth of 10Gbps (Sycamore Networks).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.2.2 Wavelength Add/Drop Multiplexers

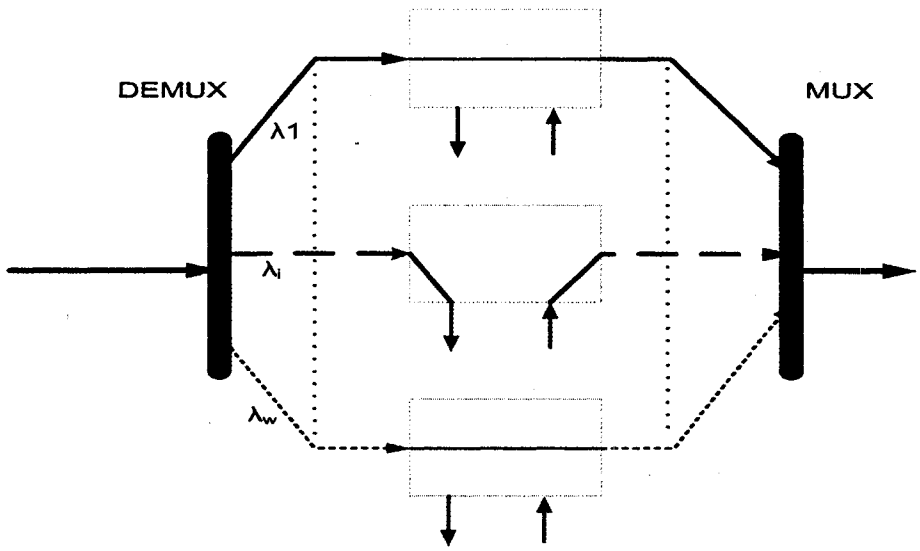

Figure 2.2: A Wavelength Add/Drop Multiplexer (WADM)

An Add/Drop optical multiplexer whose architecture is shown in figure 2.2, consists of switches. Each switch corresponds to a specific wavelength. Based upon the direction of the switch it can either disturb or un-disturb the signal corresponding to that wavelength. In other words, it can either continue on the next optical link or it can be dropped locally and that wavelength would be free so that a new data stream can be added. Note that more than one wavelength can be dropped or added at a time.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.2.3 Fiber and Wavelength Cross Connects

- **Passive Star**

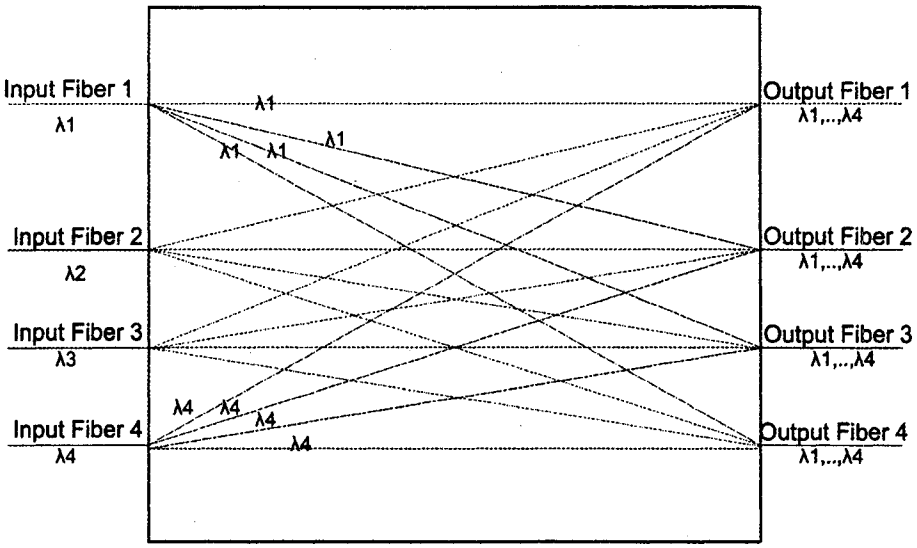

Figure 2.3: A Passive Star

A passive star having a number of input and output ports, is used in network applications. An optical signal introduced into any input port is distributed to all other ports. Because of the nature of the construction of a passive star coupler, the number of ports is usually of a power of 2 as seen in figure 2.3. The passive star is used to build local WDM networks.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- **Passive Router**

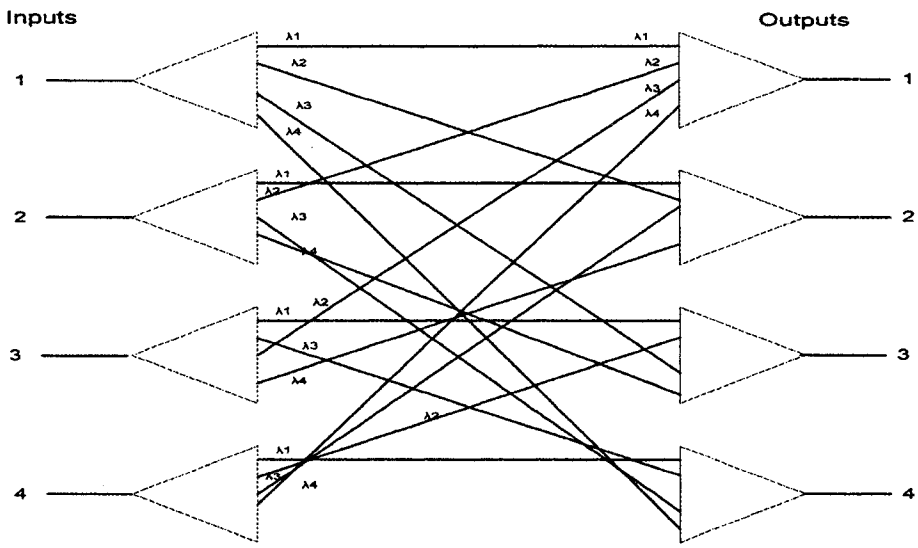

Figure 2.4: A Passive Router

A passive router routes each connection depending on a routing matrix. The difference between the passive star and the passive router, is that the passive router can route N^2 simultaneous connections[2] through itself, while the passive star can only route N simultaneous connections, as seen in figure 2.4. The passive router is mainly used as a mux/demux device.

- **The Active Switch**

The active switch can also support N^2 simultaneous connections like the passive router. The difference between them is that the active switch's routing matrix can be reconfigured on demand under electronic control. However, the active switch is not passive, therefore it needs to be powered. The active switch can also be referred to as a Wavelength Routing Switch (WRS). The main application of the active switch is in constructing wide area wavelength routed networks.

[2]Where N in Figure 2.4 is equal to 4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 The Construction of a Wavelength Routed Wide Area Optical Network

A wavelength routed optical network is shown in figure 2.5. As it can be seen the network consists of active switches connected by fiber links. Each access station is connected to an active switch through a fiber link, this combination is called a network node. Each node contains a tunable transmitter and receiver.

An example of communication between nodes is shown in figure 2.5. If there is a

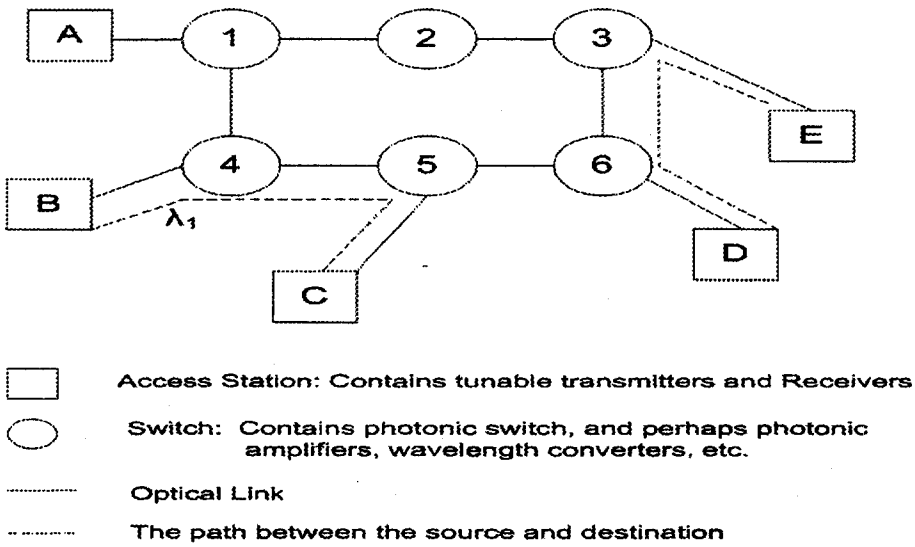

Figure 2.5: A Wavelength routed (wide area) optical WDM network

connection request between nodes B and C, it is shown that the light path is using the wavelength λ_1. If a wavelength conversion device is not available, then both nodes have to be tuned on the same wavelength. This is called the wavelength continuity property of the light path. If a wavelength converter is present as in the example in figure 2.5; the light path between nodes D and E does not have to use the same wavelength throughout the path.

A major requirement for wavelength routed optical networks is that if there are two or more light paths traversing the same fiber link, they must be on different wavelengths so interference does not occur.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.4 Network Control Management

Once a connection request is received at the node, the node is required to calculate the path to the destination and to reserve the required wavelength along the path. The node has to have/inquire the network state information in order to know which wavelengths are available. Therefore, there must be some kind of control mechanism in order to reserve and free the resources needed. The control mechanism can either be centralized or distributed.

- **Centralized**

  Centralized routing (White 2002) dictates that the routing information generated are stored at a central location within the network. Once a connection request is received at the node, the node needs to consult this centralized site and inquires as to the best route on which to transmit the data. The central site maintains a routing table. The primary advantage of using a centralized routing is that all routing information is kept at one node or site. Having only one routing table, minimizes routing table storage and eliminates the possibility of multiple routing tables with conflicting information. A major disadvantage of the centralized routing is that if the node holding the one and only routing table crashes, the entire network will have no routing information. Also if all other nodes must consult the one node holding the routing table, network congestion or a bottleneck may result at the routing table's node.

- **Distributed**

  There are two main distributed network control management approaches. The first approach is proposed in (Ramaswami and Segall, 1997), and is referred to as the "link-state approach" because it routes connections in the link-state fashion. The second approach is referred to as the "distance-vector routing approach" because it utilizes the distributed Bellman-Ford routing algorithm (Bellman 1958). The two are described below:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. **Link State**

   In the network, each node has information about the whole network topology, (*i.e.,* it knows which wavelengths are free and which paths to take). Upon a call request, the node chooses a path and a wavelength based upon the information it has. Once the route and the wavelength are chosen, the node attempts to reserve the wavelength on each link along the chosen path and sends an acknowledgment message back to the source. If all the reservations are successful, then it sends a setup message to each of the nodes, therefore the appropriate switches are then configured at each node, and the connections are established. If the reservation is not successful, then a takedown message is sent to the nodes to free up all the reserved wavelengths and the call is blocked. Consequently, when a call is accepted or blocked, each node sends an updated message to the rest of the nodes in the network. This message includes any changes in the status of the wavelengths being used on the node's outgoing links.

2. **Distance Vector**

   In the distance-vector approach, each node does not need to maintain complete link-state information on the network, but maintains a routing table instead. This routing table specifies the next hop and the distance (cost) to the destination for every destination and wavelength. This approach uses the Bellman-Ford algorithm to maintain the routing tables. This method requires the nodes to update their routing table whenever a connection is established or taken down. Therefore each node sends routing updates to their neighbouring nodes periodically or whenever the status of the node's outgoing links change. When a connection request is received, the source node looks up its routing table and selects the wavelength that has the least cost to the destination (if more than one wavelength have the same least cost, then the first fit approach is used for the wavelength assignment).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.5 The Routing and Wavelength Selection

The performance of the network is dependent on how the desired wavelength and path are chosen, which are known as the Routing and Wavelength Selection (RWA) problem. The search for the free wavelength is simple since any available wavelength can be assigned along the selected route. The selection process is crucial so as to maximize the wavelength utilization. Selection is further classified into sequential and combinatorial approaches, as presented in (Choi *et al.* 2000). The sequential approach sorts routes to be assigned and then assigns a wavelength to these routes. On the other hand, the combinational selection considers the inter-dependency of each selection. It is then broken down into optimal and heuristic approaches. The optimal approach is an NP-complete problem which is very difficult to apply in larger networks. However, in heuristic approaches, it is hard to reduce the search space to a smaller set of routes. The description of each approach is as follows:

- **Sequential Selection**
  - **Selection order**
    * Largest number of neighbors-first schemes, sort the routes according to the number of neighbors in an attempt to assign an available wavelength.
    * Largest available wavelength-first schemes, sort the routes in the order of available wavelengths.
    * Largest traffic-first schemes, sort the routes in order of traffic requirement.
    * Largest path-first schemes, sort routes in order of the number of hop counts for each route.
    * Shortest first schemes, sort the routes with the shortest number of hops first.
    * Random schemes sort routes in a random order.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- **Selection rule**
  - First fit schemes select the first available wavelength in numerical order.
  - Most used schemes attempt to allocate the most used wavelength first.
  - Least used schemes attempt to allocate the least used wavelength first.
  - Random schemes attempt to randomly allocate a wavelength.

- **Combinational Selection**
  - Optimal selections can be solved by excessive search, but they do not ensure that the algorithm can handle large networks.
  - Heuristic selection algorithms work very well with the network sorting problems and they are divided into genetic algorithms, simulated annealing algorithms and Tabu algorithms.

In summary, WDM optical network technology is introduced and examined from different perspectives. Major problems and setbacks that face the WDM network have been discussed in order to give the network designer the full picture.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

Path Selection Algorithms

Constraint-based path selection algorithms are needed to find a path that would satisfy a set of QoS constraints[1]. This problem is NP-complete (Garey and Johnson 1979), thus many heuristic algorithms have been proposed. Many of these algorithms are presented in (Kuipers *et al.* 2002). When selecting a path between a desired source and destination, in which more than one constraint is present, choosing a path can have dramatic effects on the QoS. Therefore, the key issue is to identify efficient paths that can satisfy the given QoS constraints. This is known as the QoS-based routing problem. The goal of this chapter is to shed some light on the myriad existing multi constraint path selection algorithms proposed for QoS-based uni-cast routing. These algorithms are DCCR (Guo and Matta 2003), LDP, and HZ_1 (Handler and Zang 1980).

Before exploring each algorithm, a definition of the notations used is needed. Let $G(N, E)$ represent the network topology, where N is the set or the number of nodes, and E is the set or the number of links. The source and destination nodes are represented by s and d, respectively. The number of QoS metrics is denoted by m. Each link is characterized by an m-dimensional link weight vector, consisting of m nonnegative QoS weights where $w_i(u, v), i = 1...m, (u, v) \in E$ as components. The constraints that are being considered are additive constraints, in which the QoS value

[1]The QoS constrains refers to the minimization of the delay, the cost, etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of a path is equal to the sum of the corresponding weights of the links along that path.

3.1 Delay Cost Constrained Routing (DCCR)

DCCR is presented in (Guo and Matta 2003). DCCR takes the same greedy strategy as in Dijkstra's algorithm[2] (Tanenbaum 1996). DCCR algorithm uses a nonlinear equation 3.1, in searching for the optimum path. When using a nonlinear weight function, only recording one best path from source to destination may lead to failure in finding an optimum path. DCCR solves this issue by applying Chong's k-shortest paths algorithm (Chong *et al.* 1995), which records k-shortest paths, listed in increasing weights for each node. By examining the k paths, the path with the lowest cost in the final step, can be chosen and returned as the optimum solution. The pseudo code for the DCCR algorithm as in (Guo and Matta 2003) is presented in 3.1.1.

$$Path\ Weight = \frac{\text{PathDelay}}{1 - \frac{\text{PathCost}}{\text{CostBound}}} \tag{3.1}$$

The k-shortest path algorithm is very similar to the Dijkstra algorithm. The basic idea of this algorithm is that it stores k number of paths in an array for each node. These paths are the current best paths from the source to this node. The algorithm then uses a heap, in order to store the nodes that have not yet been visited k times. The elements of the heap contain the following information: n_id, wgt and idx; where, n_id identifies the node and idx locates an element of the array $ND(n\_id)$ of the k-shortest paths associated with this node. As it can be seen in the pseudo code shown in Appendix A, the heap operations are based on node weight (wgt). The relaxation step in the DCCR algorithm is similar to the Dijkstra algorithm, however the difference appears in lines 17-23 in the DCCR algorithm when an unvisited node's weight is updated, then the corresponding element in the heap also needs to be updated. Also

[2]Dijkstra's algorithm is used to generate a path from the source to the desired destination subject to a single QoS constraint.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

if the weight of the current path is less than the weight of one of the k-paths recorded, the recorded path with the maximum weight is replaced by the new path.

The k-shortest path may return a path that contains loops. (Guo and Matta 2003) avoids this problem by using a non dominated strategy. A path P is said to be dominated by another path P', if and only if $D(P) > D(P')$ *and* $C(P) > C(P')$ (where $D(P)$ represent the delay of the path P and $C(P)$ represent the cost of the path P). By applying this method, and since delay and cost are additive metrics, a path that would contain a loop will always be dominated by the corresponding loop-free sub path. Consequently, the final path solution will not contain any loop.

3.2 HZ_1

The HZ_1 (Handler and Zang 1980) uses a linear weight function. The algorithm starts off with two paths; the Least Delay Path (LDP) and the Least Cost Path (LCP). They are calculated using Dijkstra's algorithm, with the weight function being the link delay and link cost respectively, which is shown in lines 1 and 2 (See Appendix B for pseudo code).

By examining the LCP, if the delay of the LCP is a feasible delay bounded path, then it is the optimal solution. If it is not feasible, therefore, at each iteration the algorithm keeps two paths, the current best feasible, which is delay bounded LDP, and the current best infeasible path LCP. Two parameters α and β are then defined in line 7. These parameters are used to construct a new linear path weight function, as seen in equation 3.2.

$$W(P) = \alpha \times D(P) + \beta \times C(P) \tag{3.2}$$

Using the linear function which include link cost and link delay, the HZ_1 algorithm tries to find the optimum (LWP) path, in order to reduce both cost and delay. A new variable is introduced γ, where γ is the current least path weight, in the algorithm $\gamma = D(LCP) \times C(LDP) - D(LDP) \times C(LCP)$. If finding the LWP is successful,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

i.e., $W(LWP) < \gamma$ and LWP is feasible ($D(LWP) \leq \Delta d$), LWP replaces the LDP to become the best feasible path, therefore, the weight given to link cost increases in the next cycle (lower cost paths are given more preference). If LWP is infeasible ($D(LWP) > \Delta d$), then the LWP replaces LCP in the next iteration. Therefore, this increases the weight given to link delay. When no more progress can be made, the algorithm stops and returns the best feasible path out of LWP and LDP as the optimum path.

3.3 Comparing HZ_1 and DCCR

3.3.1 Time Complexity

Lemma 1.(Guo and Matta 2003) *The time complexity of the DCCR algorithm is given as* $O(k|E|log(k|v|) + k^2|E| + t(A))$, *where A is any single metric shortest-path algorithm and* $t(A)$ *is the time complexity of A.*

Proof (Guo and Matta 2003):

In order to take out a minimum value from an array, it takes $O(log(k|V|))$. Since a maximum of $k|V|$ paths are considered in this algorithm. Therfore, $k|V|$ elements need to be taken out at worst case scenarios. This gives $O(k|V|log(k|V|))$. It can be seen that the most number of edges that can be used in the relaxation process is equal to $k|E|$, since each relaxation step would include taking out the largest value of elements from the k-array of the neighboring node ($O(k)$). Therefore, one heap search operation is equal to ($O(log(k|V|))$), and a heap modification is ($O(log(k|V|))$). Consequently, the total time spent for each relaxation step is $O(k|E|(log(k|V|) + k)) = O(k|E|log(k|V|) + k^2|E|)$. A cost bound needs to be calculated in the DCCR algorithm, which is achieved by running Dijkstra's algorithm to find the cost of the LDP in $t(A)$ time. If the priority queue is implemented as a binary heap, then the total time complexity of the DCCR algorithm is $O(k|E|log(k|v|) + k^2|E| + t(A))$.

Lemma 2.(Guo and Matta 2003) *The time complexity of the HZ_1 algorithm is*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$O(m(G)(|E| + t(A) + 2t(A))$, *where* $m(G)$ *is the total number of executed iterations,* A *is any single-metric shortest path algorithm and* $t(A)$ *represents the time complexity of* A.

Proof (Guo and Matta 2003):

In lines 7-18 (see appendix B) in the HZ_1 algorithm, which represent the outer loop runs $m(G)$ times, and at each iteration, the weight of at most $|E|$ edges, and the least weight path in $t(A)$ time need to be computed. Therefore, the total time is $O(m(G)(|E| + t(A)))$. A loop needs to run in order to calculate the LDP and LCP, this takes $2t(A)$, assuming Dijkstra's algorithm is used and the priority queue is implemented as a binary heap. Therefore, the time complexity becomes $O(m(G)(E|+ |E|log|V|) + 2)|E|log|V| = O(m(G)(|E| + t(A) + 2t(A))$.

3.3.2 Performance Comparison Under Different Network Topologies

A simulator (which is written in C) in which the source, destination and network topology are entered and depending on what algorithm is being used the optimum path is given. The program tests the path selection algorithms under different networks with varying number of nodes, in order to see . The positions of the nodes lie in a rectangular area. The source node s and the destination node d, are chosen such that the Manhattan distance between s and d is the longest possible distance in the graph, for example see figure 3.1.

The link delay function consists of the propagation delay T_p, the transmission delay T_t, and the queuing delay T_q. The transmission delay is ignored due to the assumption that high speed links are being used. The ratio between the queuing delay and the propagation delay is given by $\tau = T_q/T_p$, which reflects how busy the communication link is. Thus, the link delay is defined as $d(e) = (1 + \tau) \times T_p$.

In the simulation model, τ is uniformly distributed in $[0, T]$, where T is a parameter which reflects the maximum queuing delay allowed at each switch. T is set to be 10

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

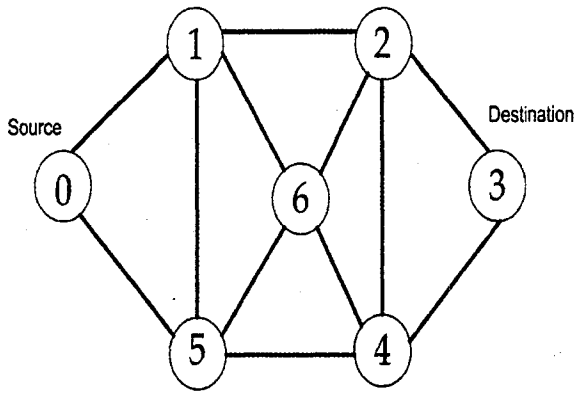

Figure 3.1: Example of the source and destination used in the simulation

in our simulations.

Link cost is generated such that it does not increase the difficulty in finding an optimum path. In the simulations there is a negative correlation between the delay and cost (*i.e.*, the more the cost, the less the delay). The link cost is given by $c(e) = M/(C+d(e))$, where M and C are parameters chosen so as to adjust the value of $c(e)$ within a reasonable range. In the simulations, $M = 1000$, $C = 1$ and $d(e)$ varies from 0.1 to 20.

The tightness of the delay bound is chosen based on the configuration of the graph. Each time a new graph is generated, Dijkstra's algorithm is used to find the LDP and LCP, then it computes the delay of these two paths, denoted by D(LDP) and D(LCP), respectively. The delay bound Δ_d is then defined to be $\Delta_d = D(LDP) + \rho(D(LCP) - D(LDP))$, where $\rho \in [0, 1]$, this is called the delay bound ratio, and thus reflects the tightness of the delay bound. In the simulations, ρ is set to 0.5[3]. Figure 3.2 shows the performance measures of the three algorithms for different network sizes. The least delay path is simply the path with the least delay and is found using Dijkstra's algorithm (see Appendix C). The negative correlation is shown in the LDP, as it costs the most in comparison with the HZ_1 and the DCCR algorithm. DCCR takes advantage of its non-linear function and the HZ_1 takes advantage of its less computational complexity, to find an optimum path. The

[3]These values were chosen in order to compare the results with those in (Guo and Matta 2003)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

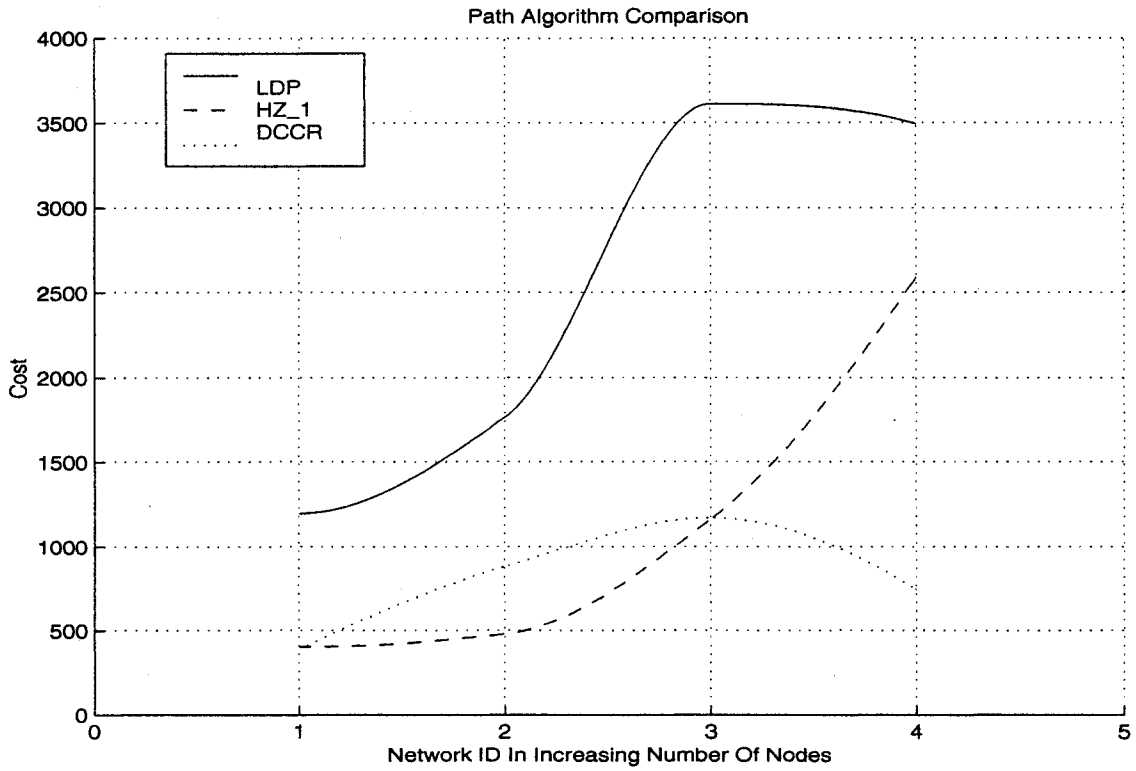

Figure 3.2: Comparison of the path selection algorithms with respect to cost

optimum paths that are found are much less than the LDP in terms of cost. The speed of the DCCR is slower than HZ_1, since the DCCR uses a non-linear weight function to obtain the optimal path(Guo and Matta 2003).

Thus, it can be seen how the optimum path can be obtained when the link cost and link delay are negatively correlated. Obviously, this assumption increases the difficulty of finding the optimal path. The tradeoff between the two algorithms, HZ_1 and DCCR, is the computational delay depending on which weight function would be used (linear or non-linear weight function).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Congestion Control

Congestion Control is a process which is used in networks to avoid congestion. Congestion control is desired since network resources are limited. If uncontrolled, users can easily overload certain networking resources, thus making the network unusable. Furthermore, the complexity is increased since network traffic is a complex nonlinear and non stationary process which is significantly affected by immeasurable parameters and variables. Hence, a precise model of this process becomes increasingly difficult as the complexity of the process increases. Fuzzy modeling has been found to effectively describe a system with these properties. This area has been the interest of many researchers.

In (Loukas *et al.* 2000) a Random Early Discard[1] (RED) algorithm is presented which uses a fuzzy logic controlled RED queue. In order to implement this, the fixed max/min queue thresholds are removed from the RED queue for each class and replaced with dynamic network state dependant thresholds. The thresholds are then calculated using a fuzzy logic controller. In (Chen *et al.* 2000) a fuzzy autoregressive (AR) model is proposed for predicting actual traffic data in high speed networks. The model uses the fuzzy clustering algorithm. Based on the prediction of traffic congestion by this proposed fuzzy-Autoregressive model, a congestion control algorithm is developed to smooth the input arrival process through decreasing the peak bit rate.

[1]RED simply sets some min and max dropping thresholds for each class.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In (Al-Hammadi and Schormans 2001) a fuzzy congestion control (FCC) scheme has been presented to monitor both low and high priority buffers. In order to reduce the complexity of the proposed FCC scheme, a separate fuzzy associative memory (FAM) is used for every buffer in the time priorities switch. The proposed FCC scheme gives lower cell loss and delay when compared with the conventional explicit binary scheme. One logic approach (Chrysostomou *et al.* 2003) for congestion control in a Diff-Serv framework is implemented with marking capabilites[2]. The design (Chrysostomou *et al.* 2003) of the fuzzy knowledge base avoids the necessity of any special parameterization or tuning, aside from the linguistic interpretation of the system behavior. The proposed fuzzy explicit marking (FEM) technique can perform equally well using homogeneous or heterogenous traffic sources without any special tuning.

In (Liu and Guan 2004) a fuzzy controller has been designed to calculate the explicit rate (ER) value. The fuzzy controller algorithm consists of three parts, which determine the PID parameters. The advantage of this scheme is that it is robust to the uncertain round trip delay and available bandwidth for available bit rate services. In (Chrysostomou *et al.* 2004), a fuzzy logic based approach for delivering an improved and more predicatable congestion control implementation within the Diff-Serv architecture is presented. (Chrysostomou *et al.* 2004) uses fuzzy logic techniques to develop a new active queue management (AQM) scheme, which is implemented within the Diff-Serv framework using a two-class FEM controller (FEM In/out) to provide congestion control. In order to maintain both high utilization and low mean delay, the fuzzy control system proposed is designed to regulate the queue of IP routes at a predefined level, by achieving a specified target queue length (TQL).

In this chapter, we propose an algorithm based on the Fuzzy Logic approach in order to control congestion in a WDM network.

[2]Marking is performed by either dropping a packet or setting its explicit congestion notification bit.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.1 Motivating Example

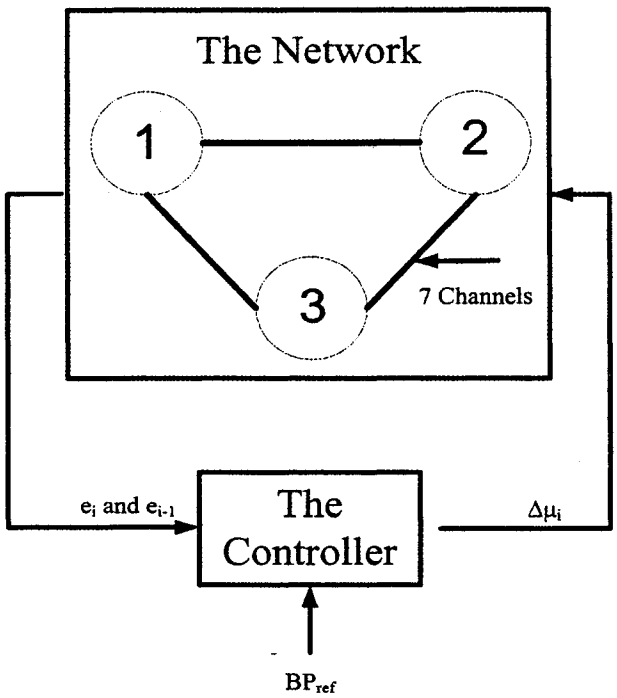

Figure 4.1: Motivating Example

We are given a small size WDM network see figure (4.1). In this example, the WDM network has seven channels on each link. The channels are chosen by the first fit wavelength routing scheme (see chapter 2). The calls' arrival times follow a Poison distribution, and the holding times follow an exponential distribution. The required task in this example is to meet a certain QoS measure. The QoS is given in terms of the Blocking Probability (BP). A BP reference (BP_{ref}) is given which represents the QoS the network has to meet. In order to satisfy this QoS a control scheme needs to be applied. The network checks its blocking probability at sampling periods (i), in order to see if this QoS is being met. The network calculates the error (e_i) at each sampling period, which is the difference between the BP_i and BP_{ref}. The control used in this scenario puts a specific delay on the incoming calls depending on the value of e_i and e_{i-1}. The network then enters the e_i and e_{i-1} values into the controller. The exact delay ($\Delta\mu_i$) is then obtained from the controller output, in order to give the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

network some time to free up its resources (see figure (4.1)), hence meet the required QoS. If for example at sampling period i, the network is not meeting its required QoS (*i.e.*, the network is congested) and the delay is calculated to be 4ms, then one of the following three things can happen at the next sampling period $i + 1$:

1. The network is not congested anymore, then the controller will takeout the unnecessary delay.

2. The network is not congested, but very close to the boundary of the BP_{ref}, then the controller will keep the delay on the network until it has fully recovered from the congestion.

3. The network is more congested than it was at sampling period i, then the controller will put more delay to give the network some time to free up more of its resources.

All of these steps will be explained in more detail throughout the chapter.

4.2 Fuzzy Systems

A fuzzy system is a system, static or dynamic, which uses fuzzy sets or fuzzy logic and the corresponding mathematical framework. Fuzzy systems can serve many functions, such as modeling, data analysis, prediction and control. One of the first applications of fuzzy systems were in automatic control. The fundamental idea of a fuzzy logic controller is to formulate the control strategy of a human operator, which can then be represented as "if-then" statements. These "if-then" statements are called rules. These rules have two parts, the first part of the rule is called antecedent, which identifies the conditions in which the rule would apply, the second part is called consequent, which contains linguistic terms (e.g., light, medium, high, etc.). These linguistic terms represent a quantitized approximation of the magnitude of the affected variables (Kosko 1997) and (Verbruggen and Babuska 1999).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In order to state the "if-then rules" appropriately, the linguistic terms that are used, the logical relationships that operate on them, and the representation of the "if-then" relations, need to be defined. These concepts include the fuzzy sets, fuzzy set operations, fuzzy relations, and the types of rule based fuzzy models.

4.2.1 Fuzzy Sets

Consider a variable that is used to describe the state of the network. The network administrator's knowledge about the magnitude of this variable may be represented by three linguistic terms, for instance: Light, Medium, or Heavy. A heavy state of the network is a set which may be defined such that the blocking ratio is greater or equal to some value P. In the conventional set theory, this set can be represented by a characteristic function $S_{Heavy}(T)$ which is defined as

$$S_{Heavy}(T) = \begin{cases} 1 & \text{if the blocking ratio} \geq P \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

According to this definition, a specific blocking rate value is either an element of the set (Heavy) or not. It makes more sense to assume that there is a gradual transition from non-heavy network states to heavy network states. This concept can be represented by a fuzzy set, which is defined by generalizing the characteristic function of conventional sets. A membership function is the characteristic function of the fuzzy set.

Fuzzy sets can be represented in discrete domains as a list of ordered pairs ($\{x_i$, $\mu(x_i)\}, x_i \in X$), and on continuous domain as an analytical formula of the membership function, for example ($\mu(x) = 1/(1+x^2), x \in \mathbb{R}$).

4.2.2 Fuzzy Set Operations

Fuzzy sets and logical connectives such as AND (conjunction), OR (disjunction) or NOT (complement), need to be combined in order to work with fuzzy rules. Conse-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

quently, the logical connectives from conventional Boolean logic have been extended to their fuzzy equivalents. If there are different domains such as, X and Y, fuzzy sets are then defined in each domain, therefore, the operator has to be applied in the Cartesian product space of X and Y (*i.e.*, for all possible pairs of x and y).

4.2.3 Fuzzy Relations

A fuzzy relation is a fuzzy set in a Cartesian product space of several domains, $X_1 \times X_2 \times ... \times X_n$. The membership degree represents how much an element is associated with the different domains X_i. An n-array fuzzy relation is a mapping

$$R : X_1 \times X_2 \times ... \times X_n \rightarrow [0, 1], \tag{4.2}$$

which assigns the membership grades to all n-tuples $(x_1, x_2, .., x_n)$ from the Cartesian product $X_1 \times X_2 \times ... \times X_n$. The fuzzy relations are used to represent the fuzzy "if-then" rules.

4.2.4 Standard Additive Model

As described before, in the rule based fuzzy systems, the relationships between variables is presented using "if-then" rules, which have the form **If** antecedent proposition **then** consequent proposition. The antecedent proposition is always a fuzzy proposition represented by a linguistic variable and a linguistic term. The form that is focused on is presented in (Zak 2003), it is the Standard Additive Model. Let X and Y be nonempty sets and let I and J be nonempty index sets[3]. If a collection of fuzzy sets $\{A_\alpha : \alpha \in I\}$ and $\{B_\beta : \beta \in J\}$ are given on X and Y respectively, and the rules are given in the form of

$$\text{IF } A_\alpha, \text{ THEN } B_\beta$$

[3]In mathematics, the elements of a set A may be indexed or labeled by means of a set J that is on that account called an index set.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The "IF-THEN" rules along with the collections of fuzzy sets form a function R from $\{A_\alpha : \alpha \in I\}$ to $\{B_\beta : \beta \in J\}$, which is called the fuzzy system associated with the rules.

A fuzzy system can be considered a "fuzzified" function from X to Y with fuzzy sets replacing points in the domain. Let us consider, for instance, the following two rules:

$$\text{IF } A_3, \text{ THEN } B_3,$$
$$\text{IF } A_4, \text{ THEN } B_2,$$

and assume that the point x_o is found in the support of A_3 and A_4. Therefore, according to the given rules, both B_3 and B_2 have to be considered. The process by which these actions are combined is called defuzzification. The outcome of defuzzification is a numerical value. The defuzzifier used is called the standard additive model (SAM). A schematic representation of a SAM with two inputs, and a single output is given in figure 4.2.

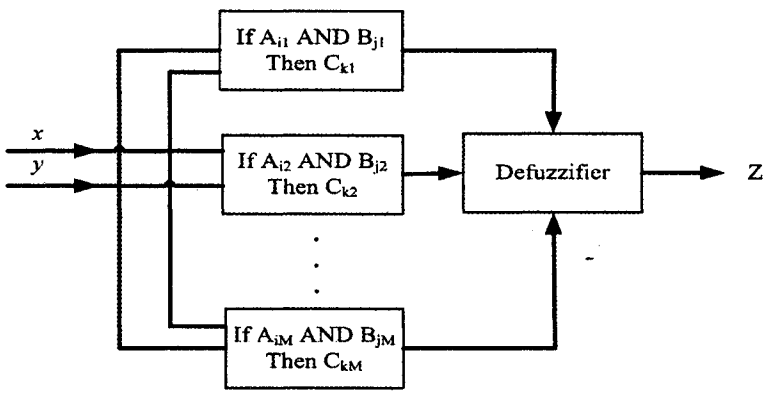

Figure 4.2: A diagram of the closed-loop system with a fuzzy logic controller

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.3 Congestion Control Using Fuzzy Logic

4.3.1 The Network Model

The model that is worked on is a small size WDM Network in which all the call requests go through a server. It is assumed that the server's queue can take up to k number of calls at a time. The calls arrive in the queue following a Poisson distribution, and the duration for each call is exponentially distributed. The network that is used is a WDM network, where the bandwidth of each link is divided into n channels, (*i.e.*, $[\lambda_o, \lambda_1, ..., \lambda_n]$). The paths are predetermined by the server using Dijkstra's algorithm, and the path is calculated based on the least number of hops. The server also calculates a backup path for each incoming call request, in case the primary path fails. The backup path is calculated by the second least number of hops. The server receives the following information about a call: source, destination, arrival time, and duration time. It is also assumed that each call needs only one channel. The same channel is reserved, throughout the path from the source to destination, for the duration of this call. If a call request does arrive but there are no channels to support it, the call is blocked and registered as a blocked call due to lack of resources to support it. Blocked calls are either handled as lost calls or they are put a side until resources are available. For the purpose of this research these calls will be classified as lost calls. The metric that is used to evaluate the network is the blocking probability which is calculated using the following equation:

$$BP(N_i, NBC_i) = \frac{\mathrm{NBC_i}}{\mathrm{N_i}} \tag{4.3}$$

where N_i is the total processed calls upto sampling index i where the sampling period is fixed at a certain number of calls, and NBC_i represents the number of blocked calls at sampling index i. The goal is to minimize the blocking probability as much as possible.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The server checks the state of the network based on the QoS[4] that needs to be achieved in terms of the blocking rate. The server then uses this information in order to adjust the delay, which is measured in milliseconds. If the network state is light and has not exceeded its QoS measure, then it will decrease the delay (if a delay has been put on these calls). However, if the network state has exceeded its QoS measure, then it will add a certain delay on the calls so that it will give the network time to free up its resources for incoming calls. Let μ_i represent the delay at sampling period i and μ_o represent the initial delay of the server's queue. The initial conditions for the delay (μ_o) in all the simulations is zero. The new delay is given by the following equation:

$$\mu_{i+1} = \mu_i + \Delta\mu_i \tag{4.4}$$

where $\Delta\mu_i$ is the delay difference that will be added on μ_i. As it can be seen while $\Delta\mu_i$ can be either positive or negative, μ_i **must** be always greater than or equal zero.

How will the server know by how much to decrease or increase its queue delay, and how sensitive will it be? A controller needs to be applied such that it would be able to translate the *BP* values that are obtained from the network, into something that would best describe the state of the network. This is where the fuzzy controller will be most useful and this is explained in more detail in the next section. Before discussing the design of the controller, the attributes of the system that is being used need to be known. First, is knowing how the system responds to different server delay settings in terms of the blocking ratio. This is done by modeling the network using a network modeling software (such as OPNET, or C++), and applying different delays, then observing its effect on the blocking probability. This will give a better understanding of how long it will take for the system to recover from congestion or if the network is light, by how much should the delay of the server decrease without congesting the network. Secondly, which QoS measure the system is working under or needs to satisfy and make a judgment whether this QoS can be satisfied using the current

[4]The QoS mentioned here refers to a specific blocking rate that needs to be achieved. This will be explained in more detail later in the chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

network model and its available resources. The QoS measure will be represented by a set point, which is the desired BP and is represented by BP_{ref}. The fuzzy controller will take the error e_i and the difference Δe_i as the inputs and $\Delta\mu_i$ will be the output of the controller. The error and the error-difference are given by:

$$e_i = BP_{ref} - BP(N_i, NBC_i) \tag{4.5}$$

$$\Delta e_i = e_i - e_{i-1} \tag{4.6}$$

where e_i represents the error at the current sampling period (i), and e_{i-1} represents the error at the previous sampling period $(i-1)$. Thus, by examining equations (4.5 and 4.6), we can see that if $\Delta e_i > 0$ then the error is increasing, and if $\Delta e_i < 0$ then the error is decreasing. In both of these cases the controller needs to act in terms of the delay $(\Delta\mu_i)$ that is going to be put on the system. In order to give the controller an idea on where the error is heading, Δe_i, which plays the role of the error time-derivative $\dot{e}_i$, is the input which gives this information. The inputs e_i and Δe_i are used to generate the "IF THEN" rules (see table 4.1), for example:
IF e_i is Large Negative and Δe_i is Large Negative then the output is Medium Positive. The table will be discussed further in the next section.

4.3.2 Fuzzy Controller

A block diagram of a closed-loop system with a fuzzy logic controller (FLC) and the fuzzy logic control architecture are shown in figure (4.3). The controller uses the error e_i and the rate of change of the error Δe_i, as its inputs. As it can be seen in figure (4.3), there are five principal elements to the fuzzy logic controller (Cirstea *et al.*2002):

- Fuzzification module (fuzzifier).
- Knowledge base.
- Rule base.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

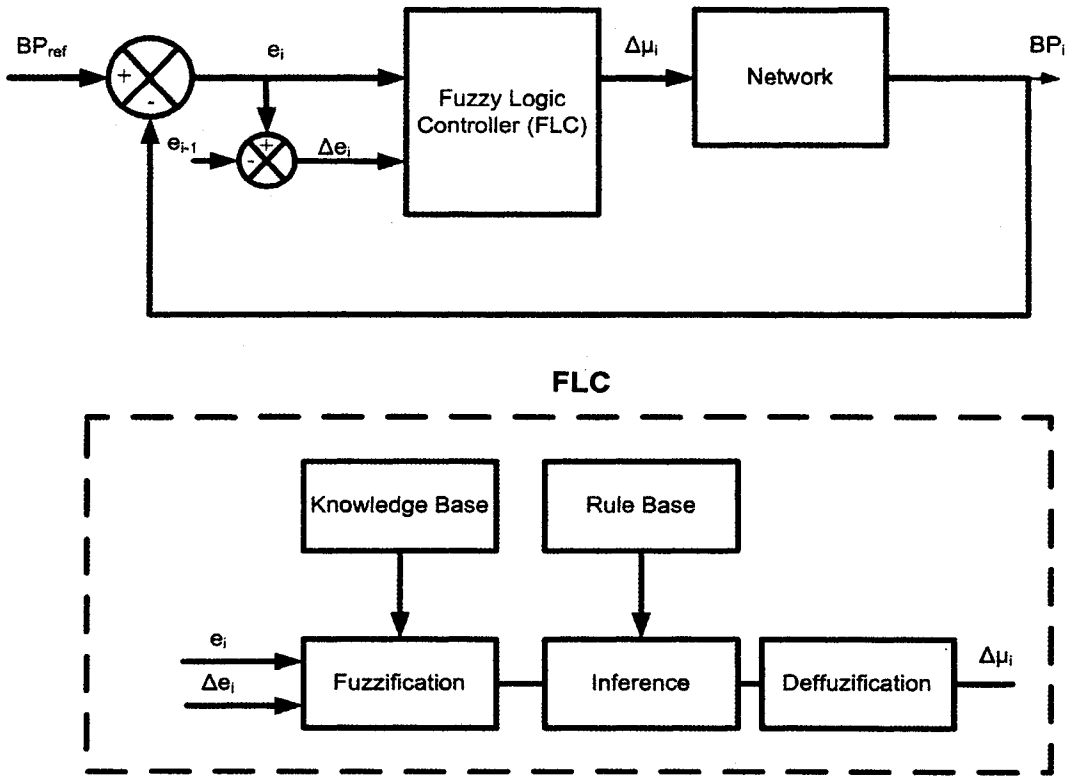

Figure 4.3: A diagram of the closed-loop system with a fuzzy logic controller

- Inference engine.
- Defuzzification module (defuzzifier).

The fuzzification module converts the crisp values[5] of the control inputs into fuzzy values, so that they are compatible with the fuzzy set representation in the rule base. The knowledge base provides all the necessary definitions for the fuzzification process such as membership functions, and fuzzy set representation of the input variables. Figures (4.4) and (4.5), show the fuzzy sets and the membership functions for both e_i and Δe_i.

The "IF-THEN" rules are presented in the form of a matrix presented in table 4.1.

We use seven membership functions for the controller's inputs and its output.

[5]A crisp value is a precise numerical value such as 2, 3, or 7.34. The crisp value is obtained after the defuzzification.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

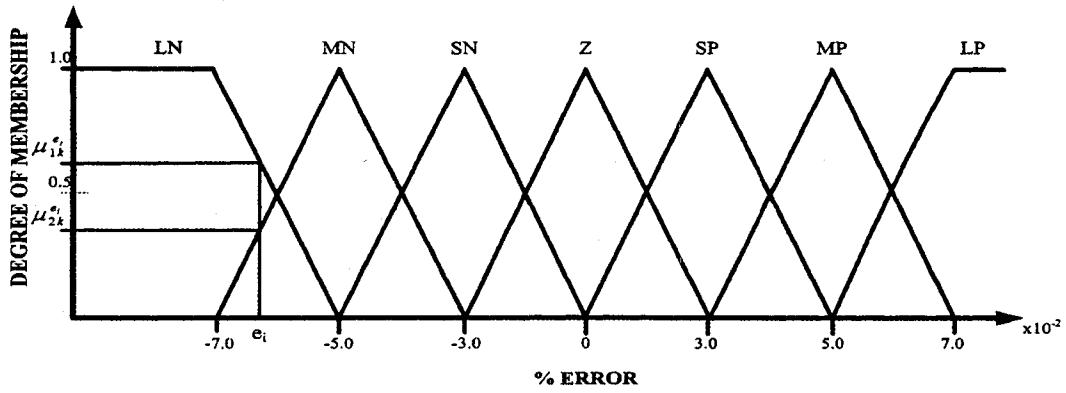

Figure 4.4: Fuzzy numbers of the linguistic variable e_i

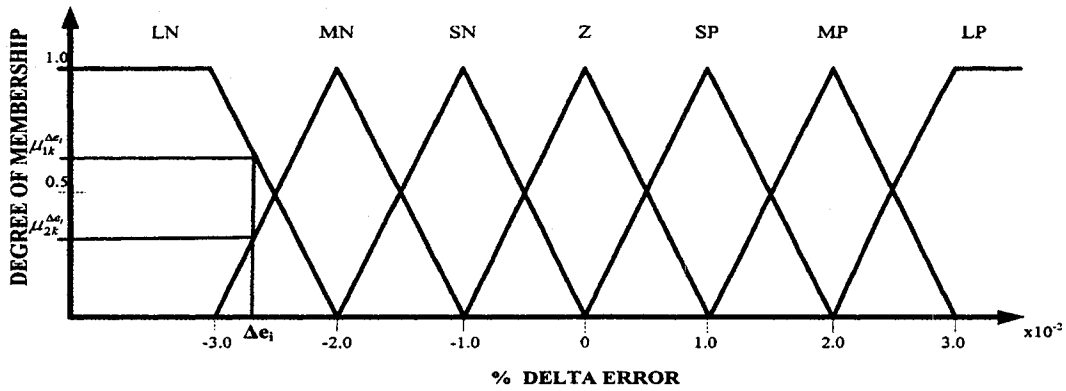

Figure 4.5: Fuzzy numbers of the linguistic variable Δe_i

We then generate the rules. The set of rules are shown in table (4.1), where LN denotes large negative, MN means medium negative, SN is small negative, Z is zero, SP is small positive, MP is medium positive, and LP is large positive. The next step involves choosing an inference engine and a defuzzifier. We use the product inference rule and the modified center average defuzzifier which is presented as follows (Zak 2003).

$$\Delta\mu_i = \frac{\sum_{k=1}^{49} \mathrm{m_k^o p_k / S_k^o}}{\sum_{k=1}^{49} \mathrm{p_k / S_k^o}} \tag{4.7}$$

for $k = 1, 2, ..., 49$, following the rules shown in table (4.1), we compute

$$p_k = \mu_{ik}^{e_i} \mu_{jk}^{\Delta e_i}, \qquad i, j = 1, 2, ..., 7 \tag{4.8}$$

The functions $\mu_{ik}^{e_i}$, $\mu_{jk}^{\Delta e_i}$ are the membership functions corresponding to the linguistic variables e_i, and Δe_i (see figure 4.4 and 4.5), respectively. The variables i, j corre-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Δe_i / e_i	**LN**	**MN**	**SN**	**Z**	**SP**	**MP**	**LP**
LN	MP(m_1^o)	SP(m_2^o)	SP(m_3^o)	MN(m_4^o)	MN(m_5^o)	MN(m_6^o)	LN(m_7^o)
MN	MP(m_8^o)	MP(m_9^o)	SP(m_{10}^o)	SN(m_{11}^o)	SN(m_{12}^o)	SN(m_{13}^o)	MN(m_{14}^o)
SN	LP(m_{15}^o)	MP(m_{16}^o)	SP(m_{17}^o)	Z(m_{18}^o)	SN(m_{19}^o)	SN(m_{20}^o)	MN(m_{21}^o)
Z	LP(m_{22}^o)	MP(m_{23}^o)	SP(m_{24}^o)	Z(m_{25}^o)	SN(m_{26}^o)	SN(m_{27}^o)	SN(m_{28}^o)
SP	LP(m_{29}^o)	MP(m_{30}^o)	SP(m_{31}^o)	SP(m_{32}^o)	Z(m_{33}^o)	Z(m_{34}^o)	SN(m_{35}^o)
MP	LP(m_{36}^o)	LP(m_{37}^o)	MP(m_{38}^o)	SP(m_{39}^o)	SP(m_{40}^o)	SP(m_{41}^o)	SP(m_{42}^o)
LP	LP(m_{43}^o)	LP(m_{44}^o)	MP(m_{45}^o)	MP(m_{46}^o)	SP(m_{47}^o)	SP(m_{48}^o)	Z(m_{49}^o)

Table 4.1: The Rules Matrix

spond to the elements' address in table (4.1) (*i.e.*, $i = 1$, $j = 1$ then it corresponds to "MP" in the table). The parameter m_{ki}^o corresponds to the controller output membership function. The output depends on the results obtained from table 4.1. Depending on the values of e_i and Δe_i, $\mu_{ij}^{e_i}$ and $\mu_{jk}^{\Delta e_i}$ are calculated. S_k^o is the spread of the corresponding controller output function (S_k^o is equal 1 for all k=1,...,49). Although the output values range between 1...49, but because we have seven recurring cases we only need seven different outputs.

The values of the output are chosen depending on the nature of the system that is being controlled. Of course since this is a continuous function the output is a mixture of all of the output values depending on the degree of membership in each of the cells in table (4.1). This process is called the Defuzzification process. The m_k^o values are obtained by trial and error to see how fast the system can operate under different conditions and if it can meet the required QoS (setpoint) desired. An example is given below to show how the results are obtained:

Example:

if after a run of the network, the following values are obtained:

- $\%e_i$=-6.5 see figure (4.4) and $\%\Delta e_i$=-2.75 see figure (4.5)
- In figure (4.4) we can see that -6.5 intersects with the LN and MN lines. There-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

fore $\mu_{1k}^{e_i}$ and $\mu_{2k}^{e_i}$ are obtained from the equations of the line.

- In figure (4.5) we can see that -2.75 intersects with the LN and MN lines. Therefore $\mu_{1k}^{\Delta e_i}$ and $\mu_{2k}^{\Delta e_i}$ are obtained from the equations of the line.

- Therefore p_k can be calculated using equation (4.8) and $\Delta\mu_i$ is calculated using equation (4.7).

In the next section, simulation results will be presented for different set points, so that we can examine how the controller reacts under different situations.

Algorithm Summary: For $i = 0$ till $i = M$, where M is the total number of calls

- N1: The network is run on the delay μ_i.

- N2: $BP(N_i, NBC_i)$ is calculated using equation (4.3).

- N3: e_i and Δe_i are calculated using equations (4.5) and (4.6) respectively, and then inputed in the controller.

- N4: The controller output $\Delta\mu_i$ (the delay difference) is then obtained from the controller.

- N5: The network is then run with delay difference added.

- N6: If $i = M$ then stop, otherwise repeat steps N1-N5.

4.3.3 Simulation Results

The network topology used is as follows:

The network model is working under the following parameters: The duration times for the calls follow an exponential distribution with mean equal to 0.1 msec, and inter arrival times follow a Poisson distribution with mean equal to 0.019msec. Hence, the network is working under 50% load. If the network load is increased then the controller needs to be reconfigured all over again. The network shown in figure (4.6), is a WDM network with eight channels on each link $(\lambda_0, \lambda_2, ..., \lambda_7)$. The simulations

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

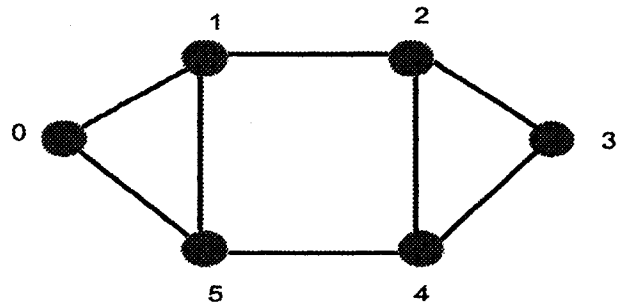

Figure 4.6: The Network Model

are performed for different setpoints (0.01,0.03,0.05,0.07,0.1) which are in terms of the blocking ration in percent. The sampling period used in the simulations is based on the number of processed calls. These simulations are performed with different sampling periods to show the controller performance with respect to the sampling period. The sampling periods that are used are 100, 200, and 250 calls. The total number of calls generated were 10,000 and 14,000, to see how this change may effect the response. The parameters for m_k^o are as follows:

- For LP is 13.63ms.
- For MP is 13.13ms.
- For SP is 12.62ms.
- For Z is 0.0ms.
- For SN is -5.05ms.
- For MN is -10.1ms.
- For LN is -15.15ms.

These values were chosen by examining the network under different values to see how fast does the controller need to react in order to be able to meet the required set-point. Figures (4.7 to 4.46) show the evolution of tracking error versus the number

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of calls. Figures (4.47-4.51) show the evolution of the control input versus the number of calls. The simulations were done when the control was applied and without to see how much the performance of the network would deteriorate without the presence of a controller.

As mentioned earlier the results shown are for setpoints (0.01%, 0.03%, 0.05%, 0.07%, 0.1%). It can be seen that the controller's worst case scenario in terms of both control and achieving the QoS requirement would be in the case where the setpoint is equal to 0.01% (*i.e.,* only 1 call can be blocked every 10,000 and 1.4 calls can be blocked every 14,000). It is shown that as the setpoint increases to 0.1% the performance of both the 10,000 and 14,000 improves in meeting the QoS measure.

The results are looked at from two perspectives, from the controller and the networking QoS point of view. In figures (4.7-4.14), where the setpoint is set for 0.01%, when the sampling is 100 calls, the controller exerts a lot of effort so that it does not just meet but also exceeds the QoS requirement by not blocking any calls throughout the 10,000 calls. Thus, giving a steady error of 0.01%, but even by increasing the number of calls to 14,000 as shown in figure (4.10) it does not change, still no calls were blocked. In figure (4.8) the controller succeeds in making the error converge to zero, thus meeting the desired setpoint. The main difference between the 100 call sampling and the other two is that the 100 call sampling is very small that it does not give the system a chance to have a drop in terms of error as shown in figures (4.8) and (4.9). The first drop is lower than the other drops in all the figures (in terms of the error e) because the blocked call happens when not as much calls have been processed, therfore the drop is much higher (*i.e.*, one blocked call in 600 gives a blocking probability of 0.16% which is high with respect to the setpoint). In the results where no control was put on the network, (see figures (4.13) and (4.14)), they start out the same as the controlled plot, until they hit the drop where the first blocked call occurs. The uncontrolled system has no control, therefore the error never converges anywhere near the zero axis.

In figures (4.15-4.22) the setpoint is changed to 0.03%. In these figures where the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

total number of calls is set to 10,000 the controller exceeds the QoS requirement but the error does not converge to zero. In the case where the number of calls are increased to 14,000 the controlled system does not meet the desired QoS measure, but is very close to zero in terms of the error.

The reason for this error not converging to zero in all the cases of the 10,000 calls, is that there is more control being applied, which is good in terms of the QoS measure so that it guarantees that the QoS will be met, even if sudden traffic changes occur. This controller was first designed for a server queue that can hold up to 10,000 calls, but the reason we did simulations with 14,000 is to see the limitations of the controller. This is shown in figures (4.18-4.20) where the error settles in the negative, thus not achieving the desired QoS. Another point that also contributes to the error settling in the negative is that sometimes the number of calls is not divisible by the setpoint.

Peaks appear in later simulations as the setpoint increases to 0.1% (see figures 4.39-4.46). These peaks occur once a call or more has been blocked, thus a great effect on the error can be seen. The controller here decreases the unnecessary $\Delta\mu$ because the setpoint permits that, therefore more of those peaks are shown.

The output which is the $\Delta\mu$ is shown in figures (4.47-4.51). The output is shown for different setpoints at a sampling of 200 calls. The $\Delta\mu$ indicates what the controller is doing, thus $\Delta\mu$ equal to zero means that no control is being applied. It can be seen that the $\%e$ and $\Delta\mu$ plots work together following the fuzzy rules that are being applied. For example in figures (4.8) and (4.47) the delay increases to force the error to increase and keeps a steady $\Delta\mu$ until the error converges to zero. It takes the system some time to react to the $\Delta\mu$ (control) that is being applied on the system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

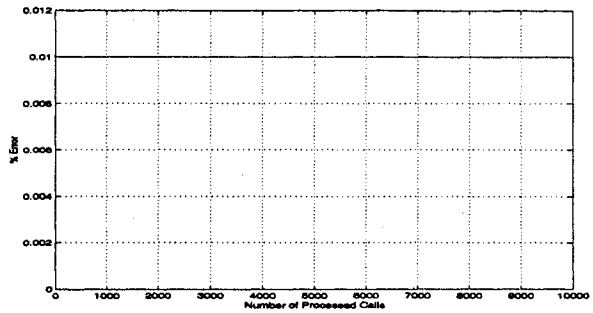

Figure 4.7: Sampling of 100 calls and Setpoint=0.01% (For 10,000 calls)

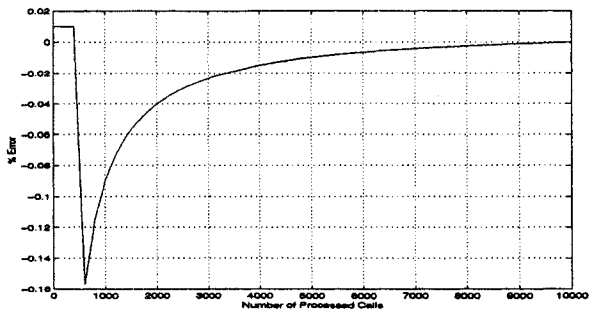

Figure 4.8: Sampling of 200 calls and Setpoint=0.01% (For 10,000 calls)

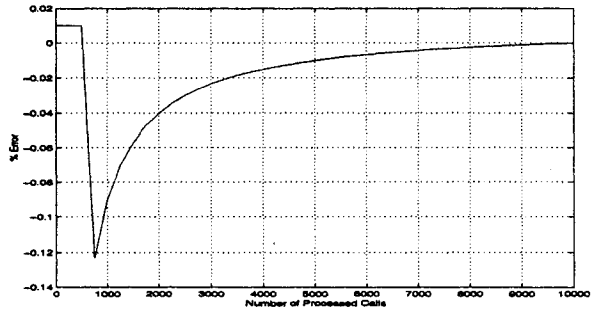

Figure 4.9: Sampling of 250 calls and Setpoint=0.01% (For 10,000 calls)

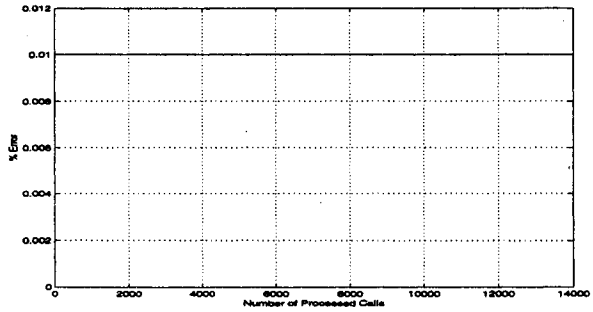

Figure 4.10: Sampling of 100 calls and Setpoint=0.01% (For 14,000 calls)

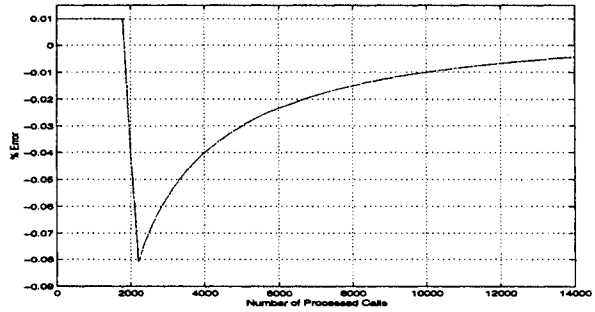

Figure 4.11: Sampling of 200 calls and Setpoint=0.01% (For 14,000 calls)

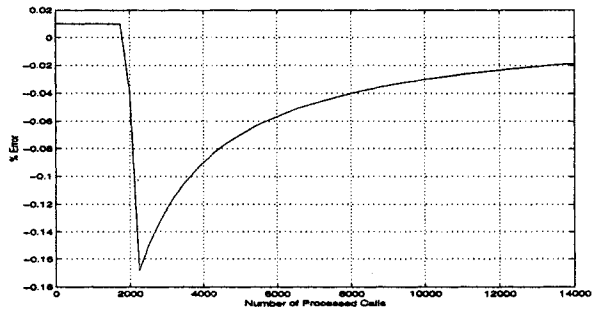

Figure 4.12: Sampling of 250 calls and Setpoint=0.01% (For 14,000 calls)

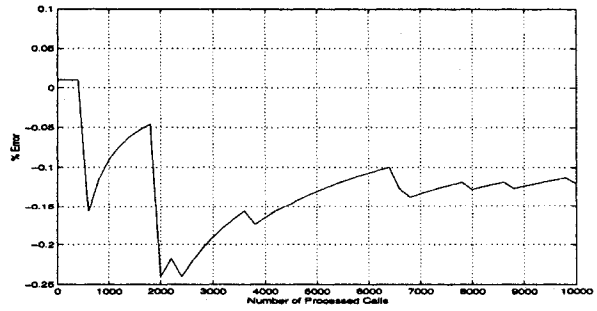

Figure 4.13: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.01% (For 10,000 calls)

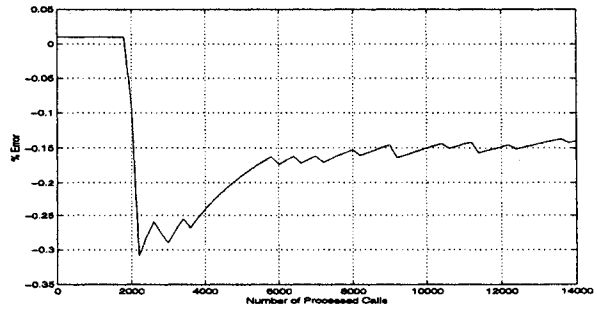

Figure 4.14: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.01% (For 14,000 calls)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

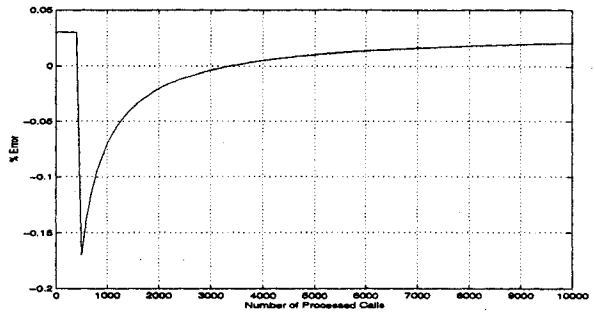

Figure 4.15: Sampling of 100 calls and Setpoint=0.03% (For 10,000 calls)

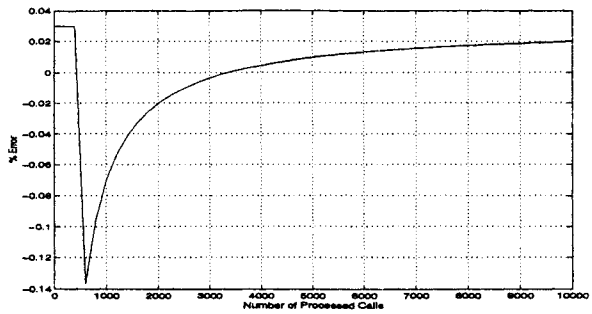

Figure 4.16: Sampling of 200 calls and Setpoint=0.03% (For 10,000 calls)

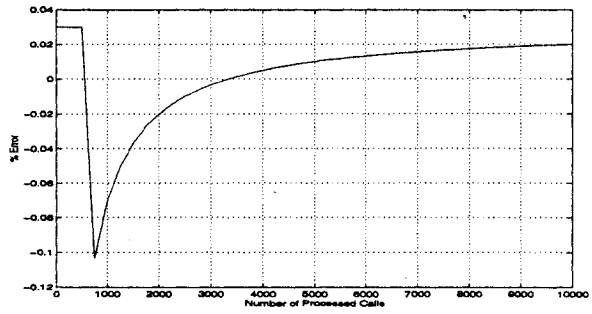

Figure 4.17: Sampling of 250 calls and Setpoint=0.03% (For 10,000 calls)

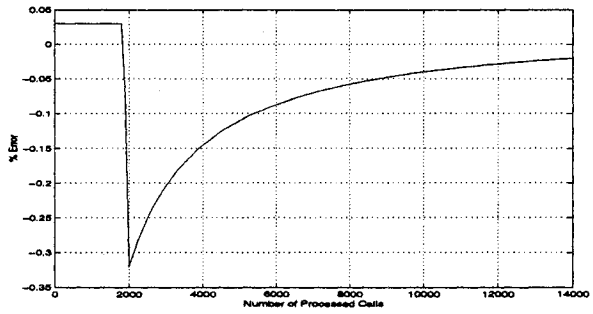

Figure 4.18: Sampling of 100 calls and Setpoint=0.03% (For 14,000 calls)

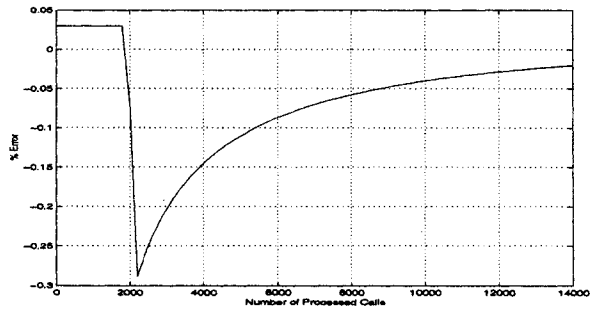

Figure 4.19: Sampling of 200 calls and Setpoint=0.03% (For 14,000 calls)

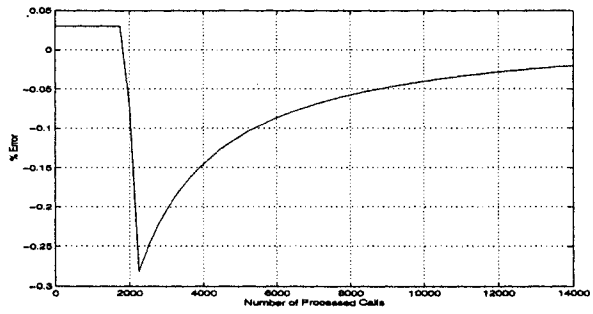

Figure 4.20: Sampling of 250 calls and Setpoint=0.03% (For 14,000 calls)

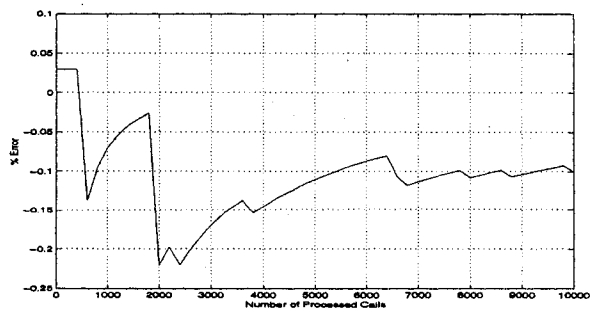

Figure 4.21: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.03% (For 10,000 calls)

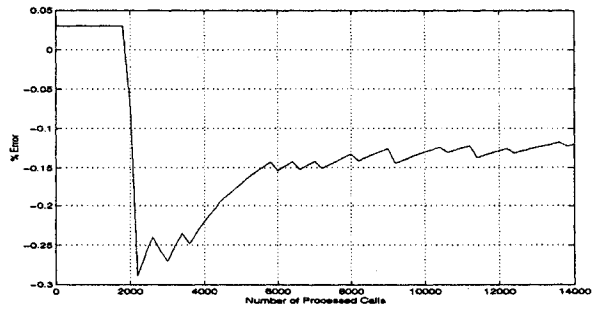

Figure 4.22: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.03% (For 14,000 calls)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

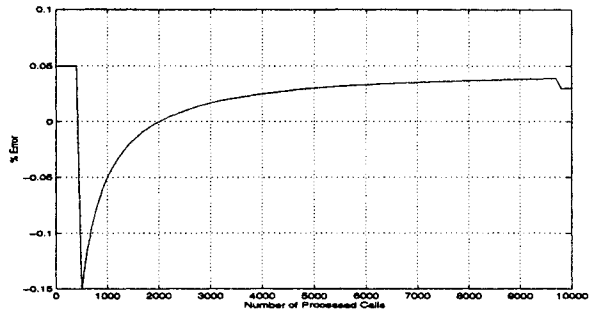

Figure 4.23: Sampling of 100 calls and Setpoint=0.05% (For 10,000 calls)

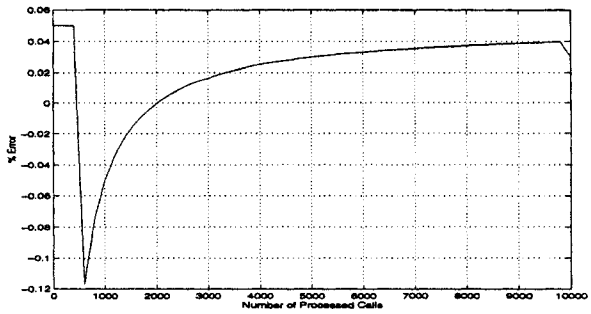

Figure 4.24: Sampling of 200 calls and Setpoint=0.05% (For 10,000 calls)

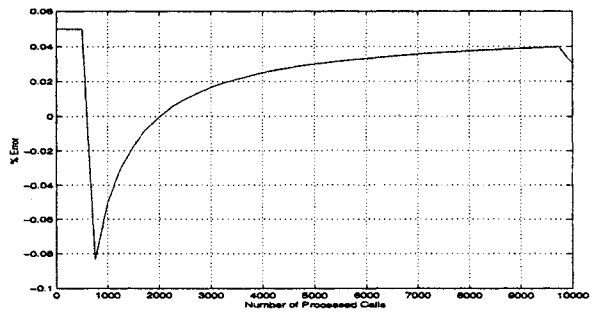

Figure 4.25: Sampling of 250 calls and Setpoint=0.05% (For 10,000 calls)

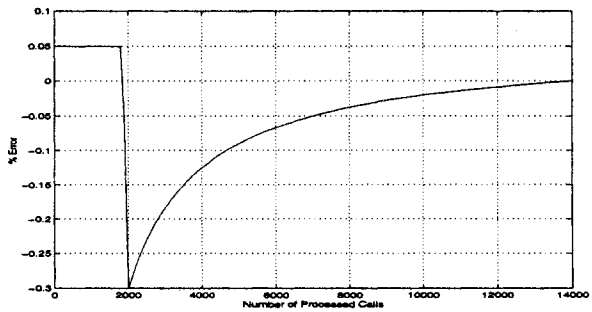

Figure 4.26: Sampling of 100 calls and Setpoint=0.05% (For 14,000 calls)

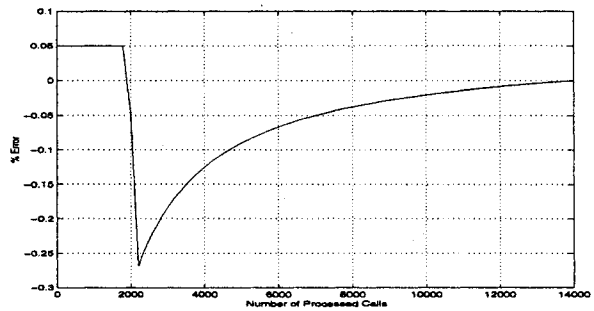

Figure 4.27: Sampling of 200 calls and Setpoint=0.05% (For 14,000 calls)

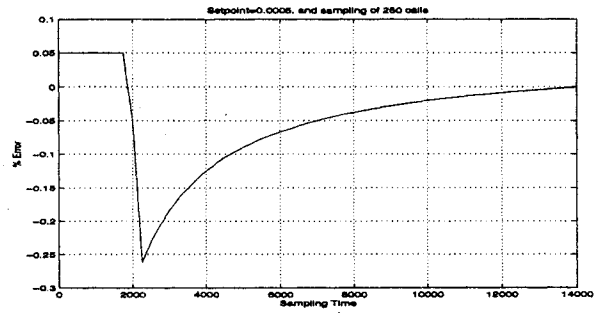

Figure 4.28: Sampling of 250 calls and Setpoint=0.05% (For 14,000 calls)

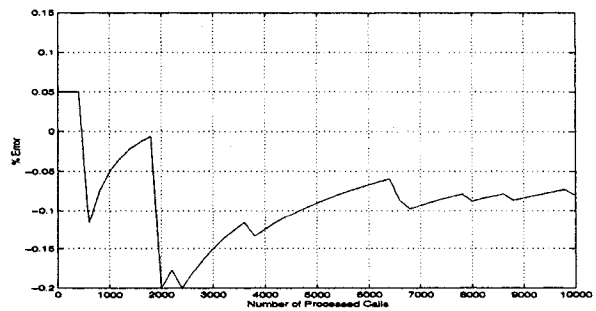

Figure 4.29: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.05% (For 10,000 calls)

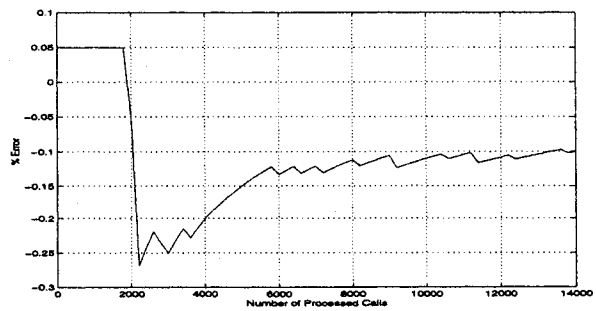

Figure 4.30: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.05% (For 14,000 calls)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

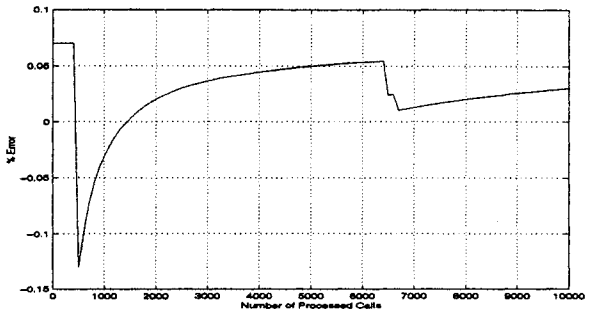

Figure 4.31: Sampling of 100 calls and Setpoint=0.07% (For 10,000 calls)

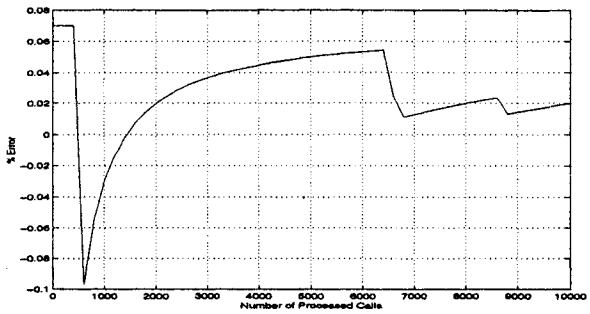

Figure 4.32: Sampling of 200 calls and Setpoint=0.07% (For 10,000 calls)

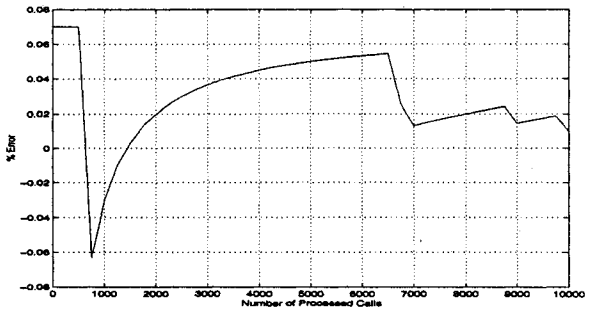

Figure 4.33: Sampling of 250 calls and Setpoint=0.07% (For 10,000 calls)

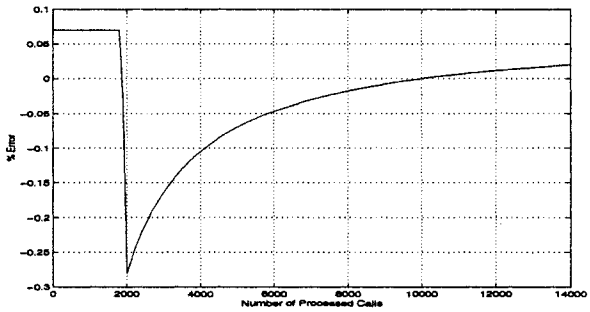

Figure 4.34: Sampling of 100 calls and Setpoint=0.07% (For 14,000 calls)

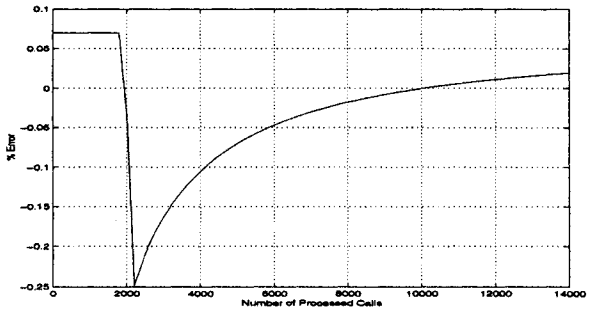

Figure 4.35: Sampling of 200 calls and Setpoint=0.07% (For 14,000 calls)

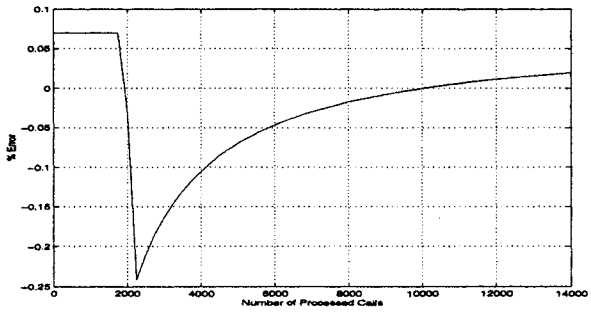

Figure 4.36: Sampling of 250 calls and Setpoint=0.07% (For 14,000 calls)

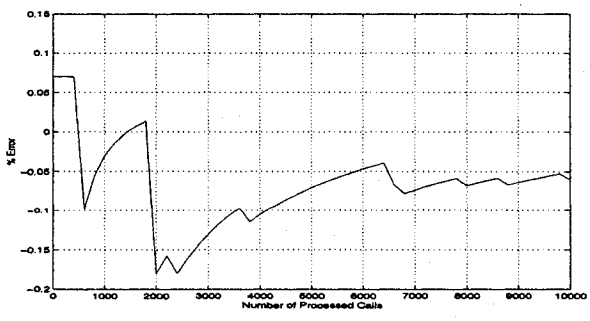

Figure 4.37: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.07% (For 10,000 calls)

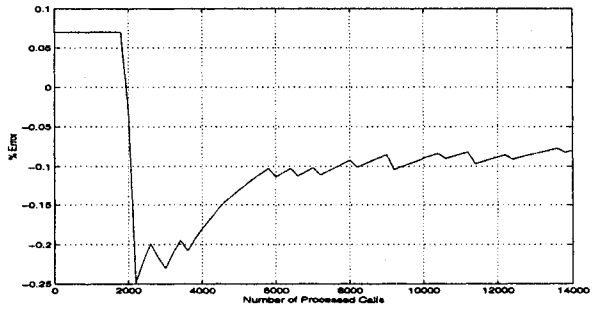

Figure 4.38: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.07% (For 14,000 calls)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

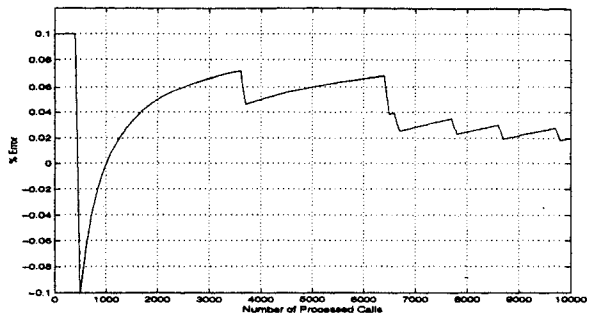

Figure 4.39: Sampling of 100 calls and Set-point=0.1% (For 10,000 calls)

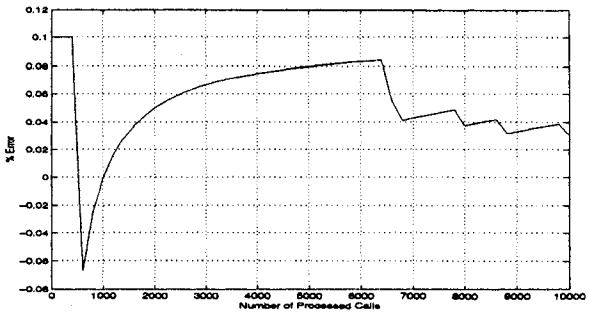

Figure 4.40: Sampling of 200 calls and Set-point=0.1% (For 10,000 calls)

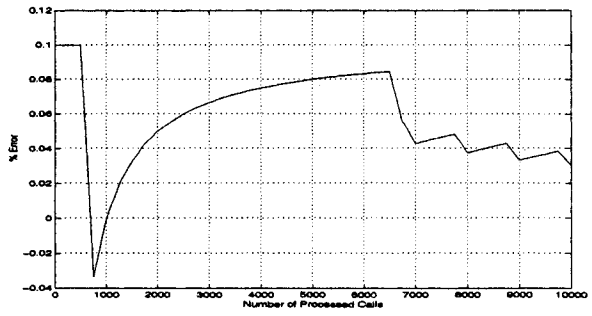

Figure 4.41: Sampling of 250 calls and Set-point=0.1% (For 10,000 calls)

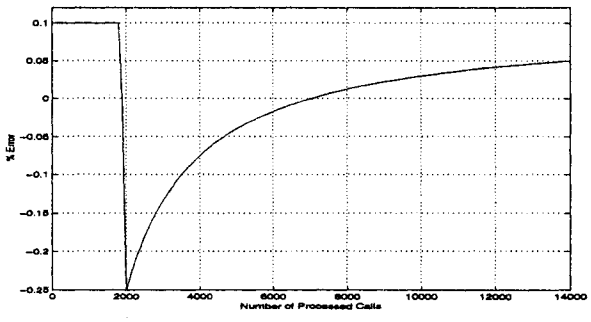

Figure 4.42: Sampling of 100 calls and Set-point=0.1% (For 14,000 calls)

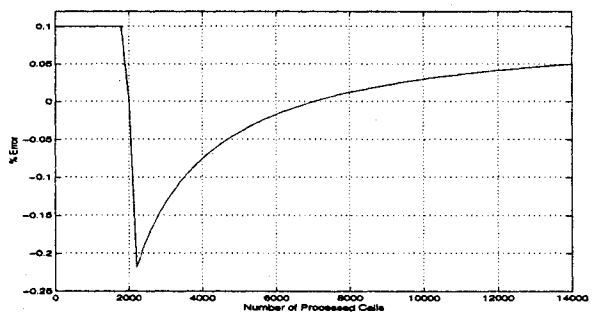

Figure 4.43: Sampling of 200 calls and Set-point=0.1% (For 14,000 calls)

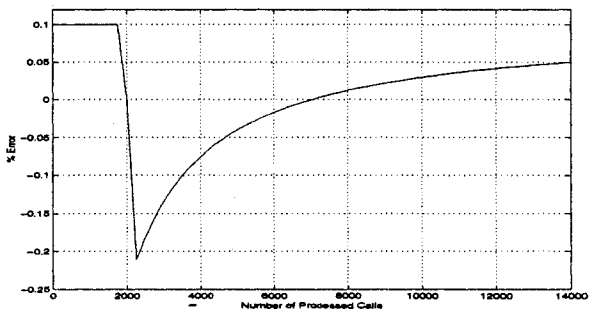

Figure 4.44: Sampling of 250 calls and Set-point=0.1% (For 14,000 calls)

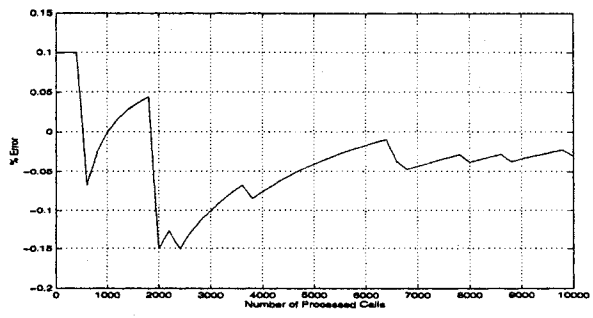

Figure 4.45: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.1% (For 10,000 calls)

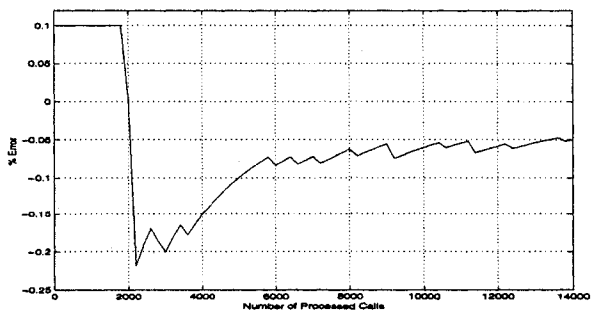

Figure 4.46: Sampling of 200 calls, WITH NO CONTROL and Setpoint=0.1% (For 14,000 calls)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

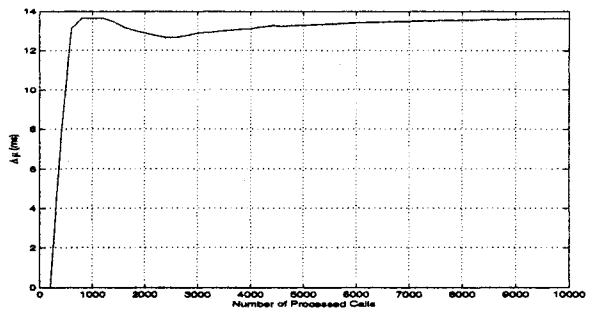

Figure 4.47: The controller output $\Delta\mu$ at Setpoint=0.01% (For 10,000 calls) and sampling of 200 calls

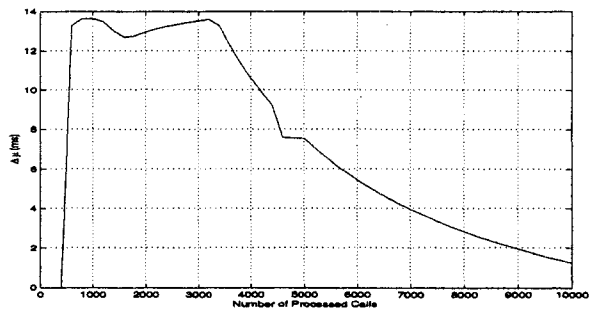

Figure 4.48: The controller output $\Delta\mu$ at Setpoint=0.03% (For 10,000 calls) and sampling of 200 calls

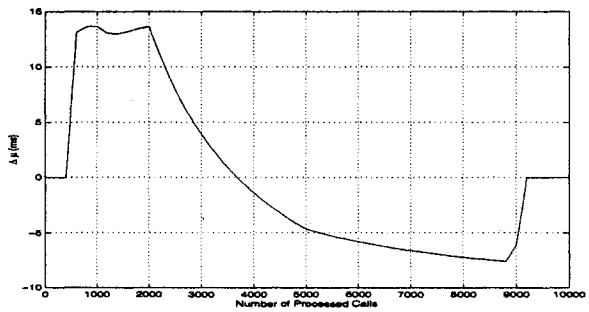

Figure 4.49: The controller output $\Delta\mu$ at Setpoint=0.05% (For 10,000 calls) and sampling of 200 calls

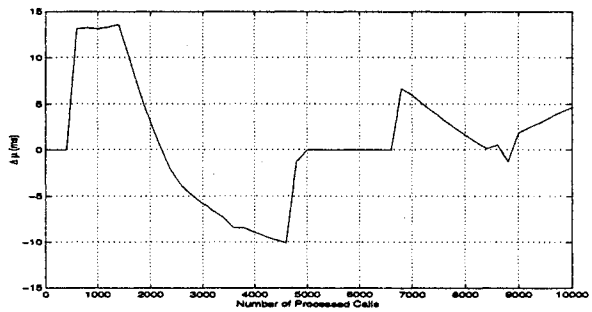

Figure 4.50: The controller output $\Delta\mu$ at Setpoint=0.07% (For 10,000 calls) and sampling of 200 calls

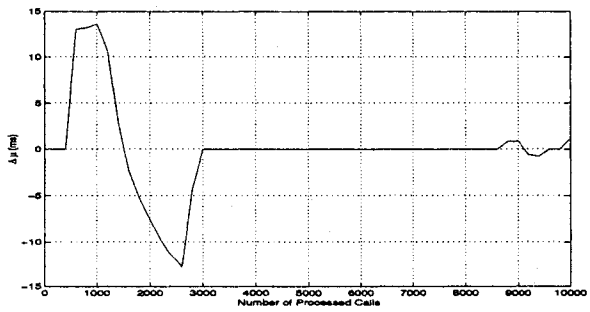

Figure 4.51: The controller output $\Delta\mu$ at Setpoint=0.1% (For 10,000 calls) and sampling of 200 calls

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

Adaptive Online Admission Control

In circuit switched networks, during the duration of a call, a path between the communication nodes has to be established and has to meet the QoS requirements (*i.e.*, short delays and slow variances). In circuit switched networks, resources[1] have to be reserved at each node along the path between the source and destination. However, if a resource for the incoming call can not be found the call is blocked. Blocked calls are assumed to be lost by the system or in some cases, they are queued until resources are available. Performance measures of such networks include blocking probability and throughput rate. In an uncontrolled network, a call is always accepted as long as there are resources available to support such a call, if resources are unavailable then the call is blocked. In order to meet the QoS requirements, call admission control has been established, which is the decision to accept or reject a call. The importance of admission control is that even when there are resources available the acceptance of certain calls can have drastic measures on the performance of the network, and also on the currently active calls.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

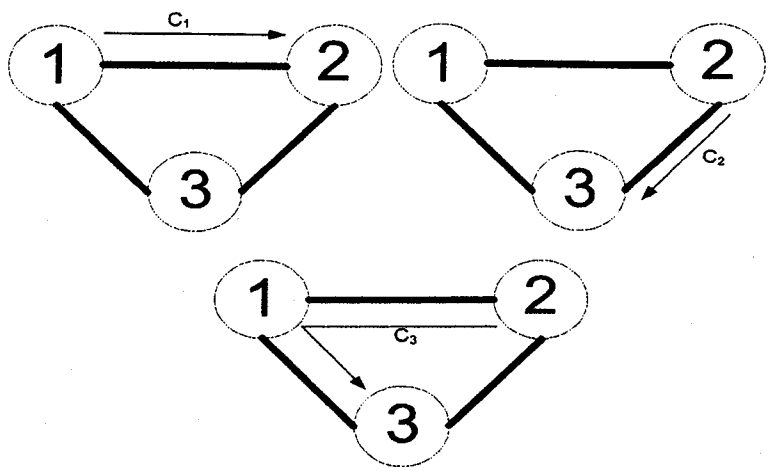

Figure 5.1: Motivating Example

5.1 Motivating Example

In this example we are given a WDM optical network with seven channels on each link. Each circuit c_i[2] (see figure (5.1)) is given a certain number of channels that it can use. The number of channels of each c_i is presented by t_i, therefore in this case we have t_1, t_2, and t_3. These values are stored in a vector called the threshold vector T[3]. The goal of this algorithm is to find the optimum threshold vector T, which is represented by T^*[4]. In this example we are given the initial vector $T = [4, 5, 3]$ and the calculated blocking probability is 4%. In order to reach the required T^*, some iterations will take place. The iterations are done based on the sensitivity of each circuit to the incoming traffic (*i.e.*, depending on the number of channels each circuit needs). In this example, after checking the sensitivity (this will be explained in more detail later in the chapter) of each circuit we find that:

- t_1 needs 3 channels instead of 4,
- t_2 needs 4 channels instead of 5,

[1]Resources in this case refer to available wavelengths, because we are dealing with WDM networks.

[2]The source destination pairs have been predetermined and they are viewed as circuits denoted by c_i. This will be explained in more detail in the next section.

[3]Where $T = [t_1, t_2, t_3]$

[4]The vector T^* is the threshold vector with the minimum blocking probability.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- t_3 needs 4 channels instead of 3.

After the iteration, the new threshold vector is $T = [3, 4, 4]$ with a blocking probability of 3%. The iterations repeat in order to find T^*. These iterations follow the algorithm that will be discussed in more detail later in this chapter. Please note that $t_1 + t_2 \leq 7$ because each link has only seven channels.

5.2 Adaptive Online Admission Control

(Gokbayrak and Cassandras 2002) deals with circuit switched networks, in which the source destination pairs have been predetermined and they can be viewed as circuits denoted by c_i. A call that uses the route c_i will be a call of type i. Generally, circuit switching is implemented either by Wave Division Multiple Access (WDMA), Frequency Division Multiple Access (FDMA) or Time Division Multiple Access (TDMA). In other words, multiplexing is dividing the total capacity into n logical channels. For each channel there exists a transceiver at the node. Therefore, a call of type i must reserve a channel/transceiver for the duration of the call. For simplicity reasons, (Gokbayrak and Cassandras 2002) assumes that all calls use the same capacity and each call uses one channel for the duration of the call. Therefore, in order for a call of type i to be accepted, there has to be one less than T_i calls currently active over circuit i, where T_i is the threshold of the calls of type i. The goal of (Gokbayrak and Cassandras 2002) is to calibrate this threshold online as the state of the network changes. The advantages of having a Threshold-based call admission control policy are:

1. The call admission algorithm can be done using local information at the source.
2. No distributional information is required in order to determine the threshold value.
3. This is an adaptive process in the sense that the optimal values of the thresholds are automatically adjusted as the operating conditions of the network change.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The drawback is that at lower network utilization there will be insufficient use of resources, since this policy works on the class of complete partitioning policies.

(Gokbayrak and Cassandras 2002) tackles the admission control as follows: it formulates the call admission control problem in order to transform it into a resource allocation problem, it then minimizes the weighted network call blocking probability by developing a specific methodology for the online optimization of the thresholds, which is presented in (Gokbayrak and Cassandras 2001). This algorithm is used to determine the optimum value of the threshold vector T, and it transforms the original discrete problem into a continuous optimization problem. The online surrogate problem[5] Methodology for stochastic discrete resource allocation problems and the adaptive call admission control algorithms will be discussed in detail in the next two sections, respectively.

5.3 Online Surrogate Problem Methodology for Stochastic Discrete Resource Allocation Problems

In (Gokbayrak and Cassandras 2001), the authors consider a discrete optimization problem, where the decision variables are positive integers. An example of this resource allocation problem is found in communication networks, where there is a fixed number of channels to optimize some performance metric.

For the problems that are considered in (Gokbayrak and Cassandras 2001), let r represent the decision vector. The set of vectors of r is represented by A_d. In a typical resource allocation setting, r_i denotes the number of resources that user i is assigned, therefore:

$$A_d = \{r : \sum_{i=1}^{N} r_i = k\} \tag{5.1}$$

where $\mathbf{r} = [r_1, r_2, ..., r_N]^T$, k is the total number of resources and N is the number of users (but in this case it is the number of connections).

Let $L_d(r, w)$ be the cost incurred over a specific sample path w when the state is r,

[5] A surrogate problem is the substitution of the original discrete problem into a solvable continuous problem. This will be explained in more details in the next sections.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and let $J_d(r) = E[L_d(r)]$ be the expected cost of the system operating under r. In the future, w will be dropped from $L_d(r, w)$ and, unless otherwise noted, all costs will be over the same sample path. Then the discrete optimization problem is to find r^* which belongs to A_d. Therefore:

$$J_d(r^*) = \min_{r \in A_d} J_d(r) = \min_{r \in A_d} E[L_d(r)] \tag{5.2}$$

When the system operates in an environment, where calls are coming in a random order, the duration is also random and when no closed form expression of $E[L_d(r)]$ is available, the problem is further complicated. Generally, this requires simulations or direct measurements made on the actual system. Most of the known approaches are based on some kind of random research, with the added difficulty of having to estimate the cost function at every step.

It can be expected that much faster improvements can be made, if the scheme is allowed to relocate multiple resources from users who have smaller cost sensitivities to users with higher cost sensitivities. The questions that (Gokbayrak and Cassandras 2001) try to answer are: Is it possible to transform a discrete optimization problem, as in equation (5.2) into a surrogate continuous optimization problem, then proceed to solve the latter using standard gradient based approaches, and finally transform its solution into a solution of the original problem? Moreover, is it possible to design this process for online operation? That is, at every iteration step in the solution of the surrogate continuous optimization problem, is it possible to immediately transform the surrogate continuous state into a feasible discrete state r?

In (Gokbayrak and Cassandras 2001), the authors transform the original discrete set A_d into a continuous set over which a surrogate optimization problem is defined and solved subsequently. An important feature of the proposed approach is that every state r in the optimization process remains feasible, so that our scheme can be used online to adjust the decision vector as the operation conditions change over time. Then the key issue is to show that, when and if an optimal allocation is obtained in the continuous state space, the transformed discrete state is in fact r^* in equation (5.2).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3.1 Basic Approach for Online Control

In the next sections, the following notations will be used: Let r_i indicate the ith component in r, r^j indicate the jth vector in the set A_d, the index n denotes the iteration steps. Since equation (5.2) is a non linear integer programming problem, one common method to solve this problem is to relax the integer constraint on all r_i so that they can be regarded as continuous (real valued) variables and then apply standard optimization techniques. Therefore:

- A_c is the relaxed set of the convex hull[6] A_d.
- L_c is the cost function over a specific sample path.

The resulting surrogate problem is to find ρ^* that minimizes the surrogate expected cost function $J_c(\rho^*)$ over the continuous set A_c. Therefore:

$$J_c(\rho^*) = \min_{\rho \in A_c} J_c(\rho) = \min_{\rho \in A_c} E[L_c(\rho)] \tag{5.3}$$

Where $\rho \in \mathbb{R}_+^N$ is a real valued state. Assuming an optimal solution ρ^* can be determined, this state must then be mapped back into a discrete vector by some means, usually using some form of truncation.

In (Gokbayrak and Cassandras 2001), a different approach is proposed which is intended to operate online. A relaxation is still invoked. A formulation of the surrogate continuous optimization problem with some state space $A_c \subset \mathbb{R}_+^N$ and $A_d \subset A_c$ is presented. However, at every step n of the iteration scheme involved in solving the problem, both the continuous states and discrete states are simultaneously updated through a mapping of the form $r_n = f_n(\rho_n)$. This has two advantages:

1. The cost of the original system is continuously adjusted.
2. It allows the user to make use of information typically needed to obtain the cost sensitivities from the actual operating system at every step of the process.

[6]The convex hull of a set of points is the intersection of all convex sets which contain the points.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The surrogate system is set to be equal to the actual system, *i.e.*:

$$\rho_o = r_o \tag{5.4}$$

Therefore, at the *nth* step of the process, let $H_n(r_n, w_n)$ denote an estimate of the sensitivity of the cost $J_c(\rho_n)$ with respect to ρ_n obtained over a sample path w_n of the actual system operating under allocation r_n. Therefore, two sequential operations are then performed at the *nth* step:

N1: The continuous state ρ_n is updated through

$$\rho_{n+1} = \rho_n - \eta_n H_n(r_n, w_n) \tag{5.5}$$

Where $\rho_{n+1} \in A_c$ and η_n is a step size parameter.

N2: the newly determined state ρ_{n+1} of the surrogate system is transformed into an actual feasible discrete state of the original system through

$$r_{n+1} = f_{n+1}(\rho_{n+1}) \tag{5.6}$$

Where f_{n+1}: $A_c \rightarrow A_d$ is a mapping of feasible continuous states to feasible discrete states which must be selected appropriately (*i.e.*, it has to satisfy a certain constraint), this will be explained later in more detail. It is shown that equation (5.5) generates the sequence $\{\rho_n\}$ and equation (5.6) is an additional operation, which converges to r^* in equation (5.2).

Note that $\{r_n\}$ corresponds to feasible realizable states, based on which one can evaluate $H_n(r, w)$ from observable data, (*i.e.*, a sample path of the actual system under r_n, not the surrogate state ρ_n).

Before addressing the issue of obtaining estimates, $H_n(r_n, w_n)$ is necessary for the optimization scheme described above to work. There are two other crucial issues that form the corner stones of the proposed approach:-

1. f_{n+1} in equation (5.6).

2. $L_c(\rho, w)$ and its relationship to $L_d(r, w)$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.3.2 Continuous to Discrete Transformation

In this section f_{n+1} in equation (5.6) is needed, in order to retrieve the r_{n+1}, which is our discrete variable.

A_c is defined as

$$A_c = \{\rho : \sum_{i=0}^{N} \rho_i = K\} \tag{5.7}$$

where ρ is a positive real number, (Gokbayrak and Cassandras 2001) begins by specifying a set F_ρ of mappings $f(\rho)$ as in equation (5.6). It first defines:

$$I_\rho = \{i \mid \rho_i \in Z_+\} \tag{5.8}$$

Where I_ρ is the set of i that corresponds to the elements in the continuous set, which are integers. Next, (Gokbayrak and Cassandras 2001) defines the following:

$$f_i^+(\rho) = \begin{cases} \rho_i \;\; if \;\; i \in I_\rho \\ \lceil \rho \rceil \;\; otherwise \end{cases} \tag{5.9}$$

$$f_i^-(\rho) = \begin{cases} \rho_i \;\; if \;\; i \in I_\rho \\ \lfloor \rho \rfloor \;\; otherwise \end{cases} \tag{5.10}$$

where $\lceil \rho_i \rceil$ denotes the ceiling (smallest integer $\geq f_i$) and $\lfloor \rho_i \rfloor$ denotes the floor (largest integer $\leq f_i$). Therefore:

$$F_\rho = \{f_i | f_i : A_c \rightarrow A_d, \forall i, f_i(\rho) \in \{f_i^+(\rho), f_i^-(\rho)\}\} \tag{5.11}$$

For all $f \in F_\rho$ and $r \in A_d$, $f(r) = r$. The purpose of $f \in F_\rho$ is to transform some continuous state vector $\rho \in A_c$ into a neighboring discrete state vector $r \in A_d$ obtained by seeking $\lceil \rho_i \rceil$ or $\lfloor \rho_i \rfloor$ for each component $i = 1, ..., N$, with this definition of continuous to discrete state transformation it can be defined as $\aleph(\rho)$ which is the set of all feasible neighboring discrete states of $\rho \in A_c$. Another definition presented in (Gokbayrak and Cassandras 2001) is the set of all feasible neighboring states of $\rho \in A_c$ is

$$\aleph(\rho) = \{r | r = f(\rho), \text{ for } f \in F_\rho\} \tag{5.12}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A more convenient and explicit characterization of equation (5.12) is by defining a residual vector $\tilde{\rho} \in [0,1)^N$ of the continuous state ρ, is given by:

$$\tilde{\rho} = \rho - \lfloor \rho \rfloor \tag{5.13}$$

where $\lfloor \rho \rfloor$ is the vector whose components are $\lfloor \rho \rfloor_i = \lfloor \rho_i \rfloor$. In the case of equation (5.7), we set

$$m_\rho = \sum_{i=1}^{N} \tilde{r}_i^j \quad and \quad \tilde{r}_i^j = 0, \quad for \quad i \in I_\rho \tag{5.14}$$

Therefore, $\tilde{r}^j(\rho)$ is an N dimensional vector component which is either 0 or 1 summing up to m_ρ. It follows that for all $f^j \in F_\rho$

$$f^j(\rho) = \lfloor \rho \rfloor + \tilde{r}^j(\rho) \tag{5.15}$$

In (Gokbayrak and Cassandras 2001) it states that any $\rho \in A_c$ can be expressed as a convex combination of points $r \in \aleph(\rho)$. (Gokbayrak and Cassandras 2001) asserts that every $\rho \in A_c$ belongs to the convex hull of all feasible neighboring state set $\aleph(\rho)$ defined in equation (5.12).

5.3.3 Optimization Algorithm

N1: Perturb ρ_n so that $I_{\rho n} = 0$.

N2: Select f_n such that $r_n = f_n(\rho_n)$.

N3: Operate at r_n to evaluate $\nabla L_c(\rho_n)$ using perturbation analysis.

N4: Update the continuous state, $\rho_{n+1} = [\rho_n - \eta_n \nabla L_c(\rho_n)]$.

N5: If some stopping condition is not satisfied, then repeat steps for $n+1$. Else $\rho^* = \rho_{n+1}$.

In (Gokbayrak and Cassandras 2001), a method is presented for solving stochastic discrete optimization problems, where the decision variables are non negative integers. It does this, by first, transforming the discrete problem into a surrogate continuous

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

optimization problem, which is solved using gradient based techniques. The solution of the original problem is found as an element of the discrete state neighborhood of the optimal surrogate state. A key advantage of this solution is that it is performed online, thus giving the system an online control nature, based on actual data from the system.

5.4 Adaptive Call Admission Control Algorithm

5.4.1 Call Admission Control Problem Formulation

(Gokbayrak and Cassandras 2002) considers an N node network with fixed routing specified by C circuits. The circuit i is denoted by the vector $c_i = [c_{i1}, c_{i2}, .., c_{iN}]$, where $c_{ij} = 1$ if circuit i passes through node j and $c_{ij} = 0$ otherwise. Thresholds are the number of channels assigned to each circuit, therefore, the goal is to determine the number of transceivers at each node. Let $t_i(j)$ be the number of transceivers assigned to circuit i at node j. The capacity constraint is now equal to $\sum_{i=1}^{c} t_i(j)c_{ij} \leq n$ for all $j = 1, ..., N$. Upon the establishment of a call, the transceivers that are used are reserved for the duration of the call. Therefore, a circuit constraint $t_i(j) = t_i(k) = T_i$, for all nodes j, k which belong to circuit i. A threshold vector is introduced whose elements hold all the circuit constraints *i.e.*, $T = [T_1, .., T_c]$.

Let $L_i(T)$ represent the cost of circuit i observed along the sample path associated with threshold vector T. Therefore, the resource partitioning problem is handled as a discrete optimization problem, where the objective is to determine the vector T to minimize a weighted sum of the expected costs $E[L_i(T)]$ over all circuits. In (Gokbayrak and Cassandras 2001), $L_i(T)$ represents the fraction of blocked type i calls over some given time interval and depends only on T_i. Therefore, what is required is the minimization of a weighted blocking probability over all call types, which is represented in (Gokbayrak and Cassandras 2002) by **(P1)**

$$\textbf{(P1)} \min_{T} \sum_{i=1}^{C} \beta_i E[L_i(T_i)] \tag{5.16}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This is subject to the resource capacity constraints:

$\sum_{i \epsilon D_j} T_i \leq n$ for all nodes $j = 1, ..., N$;

$T_i \epsilon \{0, 1, ...\}$ for all circuits $i = 1, ..., C$

where β_i is the weight associated with type i calls and $Dj = \{i : c_{ij} = 1; i = 1, ..., C\}$

For solving (P1), (Gokbayrak and Cassandras 2002) uses an online surrogate problem methodology for stochastic discrete resource allocation problems, which is discussed in detail later in this chapter. (Gokbayrak and Cassandras 2002) considers the transceivers at each node as discrete resources allocated to the different circuits in the network. By relaxing the threshold constraints, it transforms **(P1)** into a continuous surrogate optimization problem **(P2)**. This is then solved online through a stochastic approximation type algorithm, which updates the original system as the surrogate system is updated. Therefore, a sensitivity estimation is required.

5.4.2 Sensitivity Estimation

(Gokbayrak and Cassandras 2002) presents an online sensitivity estimation algorithm for the effect of a change in the threshold parameters on the cost criterion, which in this case is the weighted sum of all call probabilities over all call types.

WDM Based Modeling

Time Division Multiple Access (TDMA) is the most common circuit switching technique used, where multiplexing identifies an n-slot period for each network node, such that each call is assigned a slot in the frame; but for our interest this will be applied in a WDM optical network where each call will be assigned a channel on a link, therefore multiplexing identifies a wavelength channel on each link and for each channel there

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

exists a transceiver on each node. Depending on the bandwidth of the call, the size of the channel is determined. (Gokbayrak and Cassandras 2002) assumes uniform bandwidth requirement (*i.e., each call needs one channel*). Once a call is assigned a channel, it reserves this channel for the duration of the call.

The channel that is assigned to the call, will be reserved for the duration of this call through out the path of the call. In other words, if the call is between nodes 0 and 2 and it takes the path (0,1,2) and it reserves the channel i at the source node, which is 0, then the channel i has to be reserved at the intermediate nodes, respectively (at node 1). This channel assignment procedure results in allocating T_i channels to type i calls in each link. The quality of service measure that will be used, is that if a call can not be assigned a channel on a link, it will be blocked.

The arrival of various call types is assumed to be randomly distributed. The jth type i call is characterized by the following pair (A_j^i, ξ_j^i), where A_j^i represents the call arrival and ξ_j^i represents the call duration.

The system is described in (Gokbayrak and Cassandras 2002) from the point of view of an incoming call of type i to node q. Therefore, at node q, it has channels which are assigned to call type i and other channels which are assigned to other call types. The channel assigned type i is called a transmission channel, while all those remaining are called vacant channels. (Gokbayrak and Cassandras 2002) represents the number of free channels by f_i, therefore $0 \leq f_i \leq T_i$. If $f_i = 0$, the call is blocked. The call will be using the assigned channel for ξ_j^i, and once the call is terminated f_i is incremented by one.

(Gokbayrak and Cassandras 2002) uses the blocking probability for each type i call as a performance metric. The algorithm observes a sample path of the system and lets b_i represent the number of type i calls that are blocked. Therefore, if K type i calls are observed, then the estimate call blocking probability is given by $(b_i(K))/K)$. This is denoted by P_i and is given by

$$P_i = \frac{\mathrm{b_i(K)}}{\mathrm{K}} \tag{5.17}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Consequently, a change in the parameter T_i, will have a great effect on b_i observed on the sample path. (Gokbayrak and Cassandras 2002) uses a technique to predict the effect of removing or adding a channel from the allocation to type i calls, using only data observed on the sample path.

The "Marked/Phantom" Sensitivity Estimation Algorithm

In order to answer the questions presented in the previous section, (Gokbayrak and Cassandras 2002) views the channel *i.e.,* slot allocated for a call as a marked slot, and begins to evaluate the number of blocked calls that would have resulted had this been a vacant slot instead. This is accomplished based on data directly obtained from the sample path that is observed. The second approach considers a vacation slot and views it as a phantom slot, in which case the objective is to evaluate the number of blocked calls that would have resulted had this been a transmission slot instead.

The observed system is referred to as the nominal system and the system that would have resulted from marking a slot, is referred to as the Marked or Phantomized system. To mark or phantomize a slot is the same as marking or phantomizing a transceiver. Let us define the following sample path quantities:

- b_i: The total number of blocked type i calls in the nominal system.
- b_i^m: The total number of blocked type i calls in the marked system.
- b_i^P: The total number of blocked type i calls in the Phantomized system.
- $a_i(k)$: The total number of type i call arrivals during the k^{th} period (*i.e.,* period) of the nominal system.
- $d_i(k)$: Number of type i call completions during the k^{th} period of the nominal system.
- $d_i^m(k)$: Number of type i call completions during the k^{th} period of the marked system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- $d_i^P(k)$: Number of type i call completions during the k^{th} period of the Phantomized system.

- $f_i(k)$: Number of transmission slots available to type i calls in the k^{th} period of the nominal system. Clearly $0 \leq f_i(k) \leq T_i$.

- $f_i^m(k)$: Number of transmission slots available to type i calls in the k^{th} period of the marked system. Clearly $0 \leq f_i^m(k) \leq T_i - 1$.

- $f_i^P(k)$: Number of transmission slots available to type i calls in the k^{th} period of the Phantomized system. Clearly $0 \leq f_i^P(k) \leq T_i + 1$.

The objective is to evaluate b_i^m and b_i^P using only quantities observed along a sample path of the nominal system. The value will be used in evaluating the sensitivity of the blocking probability for type i calls.

$$\frac{\Delta \mathrm{b_i^m}}{\mathrm{K}} = \frac{\mathrm{b_i^m - b_i}}{\mathrm{K}} \tag{5.18}$$

or

$$\frac{\Delta \mathrm{b_i^P}}{\mathrm{K}} = \frac{\mathrm{b_i^P - b_i}}{\mathrm{K}} \tag{5.19}$$

After analyzing the nominal path, the result is:

$$f_i(k+1) = [f_i(k) + d_i(k) - a_i(k)]^+, f_i(0) = T_i \tag{5.20}$$

where $[x]^+ = max(0, x)$. Therefore, the number of free slots in a period is initially given by the threshold parameter T_i. The $f_i(k)$ is incremented by the number of the call completions $d_i(k)$ and decremented by the number of new calls $a_i(k)$.

Construction of the Marked System Sample Path:

The equation to determine the free slots of the marked system can be written as follows:

$$f_i^m(k+1) = [f_i^m(k) + d_i^m(k) - a_i(k)]^+, f_i^m(0) = T_i - 1 \tag{5.21}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In equation (5.22), $a_i(k)$ is obtained directly from the nominal system. (Gokbayrak and Cassandras 2002) introduces another variable allowing to test if the system is cost sensitive or insensitive to removing a slot: $u = min\{k : f_i(k) + d_i(k) \leq a_i(k), k = 0, 1, ...\}$. If such a u does not exist, then the system is insensitive. If u does exist then $f_i^m = f_i(k) - 1$ for all $k = 0, ..., u$ and $d_i^m(k) = d_i(k)$ for all $k = 0, ..., u$.

(Gokbayrak and Cassandras 2002) defines a call to be a tagged call, when it is accepted in the nominal system, but blocked in the marked system. (Gokbayrak and Cassandras 2002) introduces $z_i(k)$ as an additional binary variable, which indicates a tagged call at the beginning of the k^{th} period, therefore $z_i(0) = 0$. A binary variable z_i is set as $z_i(u+1) = 1$, therefore:

$$f_i(u+1) = f_i^m(u+1) = 0 \tag{5.22}$$

This call is terminated at some period $l > u$. Therefore, for the duration of this tagged call, both the nominal and marked systems, see the same number of blocked calls. Thus, it can be seen that the available slots available will be: $f_i^m(l) + d_i^m(l) = f_i(l) + d_i(l) - 1$. Therefore, two cases need to be considered:

- $f_i(l) + d_i(l) \leq a_i(l)$: In this case, the slot is freed up by the completion of the tagged call, and will be used by a new tagged call, therefore $z_i(l+1) = 1$. The process will repeat with the following initial condition $f_i(l+1) = f_i^m(l+1) = 0$.

- $f_i(l) + d_i(l) > a_i(l)$: In this case, the slot freed up by the completion of the tagged call is not used by a new one, therefore $z_i(l+1) = 0$. The process will repeat with the following initial condition $f_i(l+1) > 0$ and $f_i^m(l+1) = f_i(l+1) - 1$.

In order to formally define the dynamics of $z_i(k)$, (Gokbayrak and Cassandras 2002) introduces one more binary variable $y_i(k)$, as the indicator of the completion of the tagged call of type i within the kth period. If the tagged call is completed in the kth period, then $y_i(k) = 1$. The following balance equation is given.

$$z_i(k+1) = \begin{cases} 1 \;\; if \;\; f_i(k) + d_i(k) \leq a_i(k) \\ 0 \;\; if \;\; f_i(k) + d_i(k) > a_i(k) \;\; \text{and } y_i(k) = 1 \\ z_i(k) \;\; \text{otherwise.} \end{cases} \tag{5.23}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Note that $z_i(k)$ is completely determined from the observable quantities along the nominal sample path. There can be at most one tagged call of type i in the marked system at any instant, since only one transmission slot is removed in the marked system.

Therefore, the final step is the evaluation of the sensitivity of the blocking probability for type i calls, $(\Delta b_i^m/K)$. As defined, it is the ratio of the total number of tagged calls over the observed sample path to the total number of observed arrivals. Let $\mathbf{1}[.]$ be the usual indicator function (in which if the conditions inside the function are satisfied then it is a binary true which is equal to 1 otherwise false which is equal to 0). Thus, it is given by:

$$\frac{\Delta \mathrm{b_i^m}}{\mathrm{K}} = \frac{1}{\mathrm{K}} \sum_{k=0}^{F_i-1} \mathbf{1}[f_i(k) + d_i(k) \leq a_i(k)] \times \mathbf{1}[z_i(k) = 0 \; or \; y_i(k) = 1]. \tag{5.24}$$

In practice, Δb_i^m is simply incremented by 1 with every transition of $z_i(k)$ from (0 to 1). *i.e.,* a tagged call is terminated in that period and another tagged call is accepted.

Construction of the Phantomized System Sample Path:

The equation for the free slots in the Phantomized system is:

$$f_i^p(k+1) = [f_i^p(k) + d_i^m(k) - a_i(k)]^+, f_i^p(0) = T_i + 1 \tag{5.25}$$

Let $u = min\{k : f_i(k)+d_i(k) < a_i(k), k = 0, 1, ...\}$, if u does not exist, then the system does not block any calls of type i. Assuming that u does exist, the Phantomized and the nominal system will accept every call until the uth period. Thus $f_i^p(k) = f_i(k)+1$ for all $k = 0, ..., u$ and $d_i^p(k) = d_i(k)$ for all $k = 0, ..., u$. A call is defined as a phantom call, when it is blocked in the nominal system but accepted in the Phantomized system. A binary variable is introduced to indicate the presence of a phantom call $z_i(u+1)$ and is set to $z_i(u+1) = 1$.

In the uth period, the accepted number of calls in both systems is represented by: $f_i^p(k) + d_i^p(k) = f_i(u) + d_i(u) + 1$ calls. Therefore, for the duration of the phantom call, both the nominal and Phantomized systems, see the same number of blocked calls.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Therefore, in the *lth* period, the nominal system and the Phantomized system will have $f_i(l)+d_i(l)$ and $f_i^p(l)+d_i^p(l) = f_i(l)+d_i(l)+1$ available slots, respectively. Two cases need to be considered:

- $f_i(l) + d_i(l) < a_i(l)$: In this case the slot is freed up by the completion of the phantom call, and will be used by a new phantom call, therefore $z_i(l+1) = 1$. The process will repeat with the following initial condition $f_i(l+1) = 0 = f_i^p(l+1)$.

- $f_i(l) + d_i(l) \geq a_i(l)$: In this case, the slot freed up by the completion of the tagged call is not used by a new one, therefore, $z_i(l+1) = 0$. The process will repeat with the following initial condition $f_i(l+1)+1 = f_i^p(l+1) > 0$.

In order to formally describe the dynamics of $z_i(k)$, another binary variable is introduced $y_i(k)$ as an indicator of termination of a phantom call of type i within the *kth* period. The following balance equation is given:

$$z_i(k+1) = \begin{cases} 1 \;\; if \;\; f_i(k)+d_i(k) < a_i(k) \\ 0 \;\; if \;\; f_i(k)+d_i(k) \geq a_i(k) \;\; \text{and } y_i(k) = 1 \\ z_i(k) \;\; \text{otherwise.} \end{cases} \tag{5.26}$$

Note that $z_i(k)$ is completely determined from the observable quantities along the nominal sample path. There can be, at most, one phantom call of type i in the Phantomized system at any instant, since only one transmission slot is added in the Phantomized system.

Therefore, the final step is the evaluation of the sensitivity of the blocking probability for type i calls, $(\Delta b_i^p/K)$. As defined, it is the ratio of the total number of phantom calls over the observed sample path to the total number of observed arrivals. Thus, it is given by:

$$\frac{\Delta \mathrm{b_i^P}}{\mathrm{K}} = \frac{1}{\mathrm{K}} \sum_{k=0}^{F_i-1} \mathbf{1}[f_i(k)+d_i(k) < a_i(k)] \times \mathbf{1}[z_i(k) = 0 \; or \; y_i(k) = 1]. \tag{5.27}$$

In practice, Δb_i^p is simply incremented by 1 with every transition of $z_i(k)$ from (0 to 1). *i.e.*, a tagged call is terminated in that period and another tagged call is accepted.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Optimal Call Admission Threshold Determination

In this section, we return to the original problem, where a threshold vector T, which satisfies the capacity constraints at the nodes is sought to minimize a weighted sum of blocking probabilities. With this in mind, the algorithm presented in the previous section (Gokbayrak and Cassandras 2001) is applied here. The original discrete feasible set is transformed into a continuous feasible set, over which a surrogate optimization problem is defined and solved. It has been shown in the previous section, that when an optimal threshold vector τ^* is obtained in the continuous state space, it is transformed to the optimal threshold vector $\mathbf{T}^*$. In other words, once a solution to the surrogate optimization problem is found, it is transformed in order to obtain a solution to the original discrete stochastic optimization problem.

First, by relaxing the constraint T_i, which is an integer for $i = 1, ..., C$. The relaxed problem can be formulated as follows:

$$\min_{\tau} \sum_{i=1}^{C} \beta_i E[L_i(\tau_i)] \tag{5.28}$$

Also subject to the resource capacity constraints $\sum_{i \in D_j} \tau_i \leq n$ for nodes $j = 1, ..., N$, $\tau_i \geq 0$ for all circuits $i = 1, ..., C$. In this formulation, τ_i is the surrogate variable (real valued threshold) for type i calls, β_i is the weight associated with type i calls.

In order to determine the optimal threshold the following steps are taken

1. Evaluate:

$$\beta_i = \frac{\lambda_{\mathrm{i}}}{\sum_{\mathrm{j}=1}^{\mathrm{C}} \lambda_{\mathrm{j}}} \tag{5.29}$$

   where λ_i is the Poisson arrival rate for circuit i and β_i is the weight associated with type i calls.

2. Operate at T_n in order to evaluate the sensitivity estimate H_n where

$$(H_n)_i = \begin{cases} -\beta_i \frac{\Delta \mathrm{b}_{\mathrm{i}}^{\mathrm{P}}}{\mathrm{K}} & if \quad (T_{m+1})_i < (\tau_{m+1})_i \\ -\beta_i \frac{\Delta \mathrm{b}_{\mathrm{i}}^{\mathrm{m}}}{\mathrm{K}} & if \quad (T_{m+1})_i > (\tau_{m+1})_i \end{cases} \tag{5.30}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Evaluate the perturbed variable τ_{n+1}

$$\tau_{n+1} = \tau_n - \eta H_n \tag{5.31}$$

where η is a step size parameter.

4. Evaluate the blocking probability

$$P_b = \frac{\sum_{i=1}^{C} \lambda_i P_i(T_i)}{\sum_{j=1}^{C} \lambda_j} \tag{5.32}$$

where $P_i(T_i)$ is the expected circuit i call blocking probability with assigned threshold.

Before calculating $P_i(T_i)$, first the circuit loads $\rho_i = (\lambda_i/\mu_i)$ need to be calculated where μ_i is the mean of the calls' holding time which follow an exponential distribution, and

$$P_i(T_i) = \frac{\rho_i^{T_i}/T_i!}{\sum_{j=0}^{T_i} \rho_i^{j}/j!} \tag{5.33}$$

Summarizing the Algorithm Steps:

- Let $T_o = \tau_o$
- Perturb τ_n (if necessary) so that all components are non integer.
- Operate at T_n to calculate the sensitivity of the threshold vector.
- Calculate τ_{n+1} and obtain T_{n+1}
- Calculate the blocking probability, if it satisfies a certain condition, then the optimum $\tau^* = \tau_{n+1}$, else repeat the steps for $n + 1$.

Thus, it can be seen that the threshold based call admission policies for circuit switched networks has been considered and a scheme has been developed for adjusting the threshold parameters online, where the goal is to minimize a weighted sum of call blocking probabilities. The main advantages of this control scheme lie in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

its implementational simplicity and the fact that it is completely distributed in nature. It is adaptive in the sense that it can automatically adjust the thresholds as the operating conditions change and it does not require any explicity distributional modeling assumptions. The seed of this algorithm is the marked/phantom slot algorithm, developed for online estimation of the sensitivity of the call blocking metric defined above with respect to the thresholds. The reason is that this algorithm is based on directly observable network data that does not require any special distributional modeling assumptions.

5.4.3 Simulation Results

In this section, we present the call admission approach and its features by considering the following network: The objective is to determine the optimal thresholds T_i^*

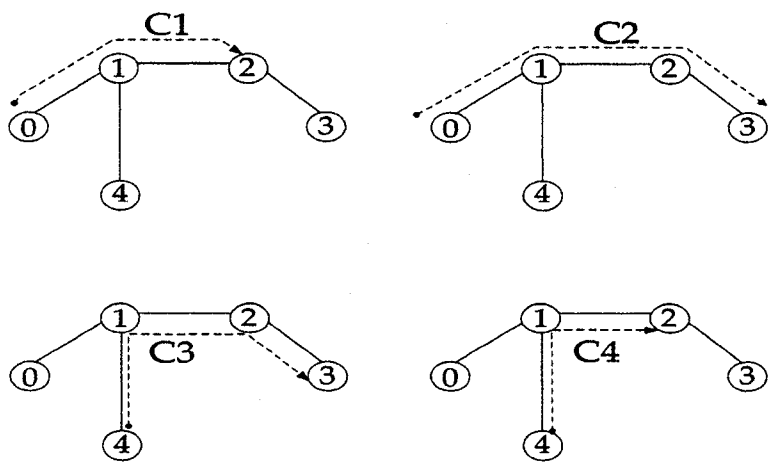

Figure 5.2: Network Topology

(in the call admission policy) so as to minimize the weighted network call blocking probability (equation (5.32)). The choice of the value for the step size parameter η is very crucial, and it can only be obtained by trial and error.

In the simulations, we assume $n = 20$ channels per link. The arrival of the calls follow a Poisson distribution with mean $\lambda = 0.02$ and the holding times follow an exponential distribution with mean $\mu = 0.01$. All the call types have the same arrival and holding time means. The circuit loads $\rho_1 = \rho_2 = \rho_3 = \rho_4 = 0.2$, and β_i can be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

calculated using equation (5.29). Therfore the corresponding separable optimization problem in equation $\min_{T\in\mathbb{Z}_+^N} \sum_{i=1}^{4} \beta_i P_i(T_i)$, subject to:

- $T_1 + T_2 \leq 20$;
- $T_1 + T_2 + T_3 + T_4 \leq 20$;
- $T_2 + T_3 \leq 20$;
- $T_3 + T_4 \leq 20$;

All the call types are of equal importance with the mentioned system parameters. To illustrate the approach, we performed a single run optimization of the threshold in the call admission policy with different constant step size parameters (η). The gradient projection method (gradients in this case are directly calculated using equations 5.30 and 5.32, so that there is no estimation noise) with a constant step size parameter η. Simulations were done using different values for the constant step size parameter η, with a constant initial value for the threshold T which is [4, 4, 4, 4] and [3, 4, 2, 4]. These are presented in the form of tables which are composed of the calculated threshold vector T at each sampling period k (which is taken every 25 calls) and the corresponding weighted network probability (P_b). As it can be seen every vector T does not exceed the total available number of channels which is 20.

In the simulations choosing any initial threshold vector is straightforward, but choosing a step size parameter η is very tricky in the sense that it is different for each network and the chosen initial threshold vector. By examining equation (5.31), it can be seen that the choice of η effects τ_{n+1}, if η is too small then τ_{n+1} will take more iterations to reach the desired value, but if η is large then τ_{n+1} will change too fast thus skipping potential values. Thus the choice of η can effect the threshold choice, therefore effecting the resulting weighted network probability. The results shown in table (5.1-5.4) used different values of η. For each set of results we get an optimum vector T^* at which the minimum P_b is achieved. The best T^* in terms of the P_b is when $\eta = 25$ at k=1. At $k = 1$ at the case of η=25, $T^* = [5, 4, 6, 4]$. It only took one

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

iteration to obtain the minimum P_b. It can be seen that the optimum value of η in this case is 25. Consequently, the user has to run the algorithm on different step size parameters in order to make sure that the results obtained are optimum.

k	T	P_b
0	4,4,4,4	5.45822×10^{-5}
1	4,4,5,4	4.14825×10^{-5}
2	4,4,6,4	4.09548×10^{-5}
3	4,4,7,4	4.09371×10^{-5}
4	4,4,8,4	4.09366×10^{-5}
5	4,4,8,4	4.09366×10^{-5}
6	4,4,8,4	4.09366×10^{-5}
7	4,4,8,4	4.09366×10^{-5}
8	4,4,8,4	4.09366×10^{-5}

Table 5.1: At $\eta = 10$

k	T	P_b
0	4,4,4,4	5.45822×10^{-5}
1	5,4,6,4	2.78551×10^{-5}
2	5,3,9,3	5.46397×10^{-4}
3	5,2,10,3	4.37183×10^{-3}
4	5,2,10,3	4.37183×10^{-3}
5	5,2,10,3	4.37183×10^{-3}
6	5,2,10,3	4.37183×10^{-3}
7	5,2,10,3	4.37183×10^{-3}
8	5,2,10,3	4.37183×10^{-3}

Table 5.2: At $\eta = 25$

k	T	P_b
0	4,4,4,4	5.45822×10^{-5}
1	6,3,8,3	5.45487×10^{-4}
2	5,3,9,3	5.46397×10^{-4}
3	5,3,9,3	5.46397×10^{-4}
4	5,3,9,3	5.46397×10^{-4}
5	5,3,9,3	5.46397×10^{-4}
6	5,3,9,3	5.46397×10^{-4}
7	5,3,9,3	5.46397×10^{-4}
8	5,3,9,3	5.46397×10^{-4}

Table 5.3: At $\eta = 50$

k	T	P_b
0	4,4,4,4	5.45822×10^{-5}
1	6,2,9,3	4.37130×10^{-3}
2	6,2,9,3	4.37130×10^{-3}
3	6,2,9,3	4.37130×10^{-3}
4	6,2,9,3	4.37130×10^{-3}
5	6,2,9,3	4.37130×10^{-3}
6	6,2,9,3	4.37130×10^{-3}
7	6,2,9,3	4.37130×10^{-3}
8	6,1,10,3	4.193696×10^{-2}

Table 5.4: At $\eta = 75$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Another set of values are presented with an initial T vector of [3, 4, 2, 4]. The results are presented in tables (5.5-5.8). Again different values of η are simulated to see which η would give the optimum solution in terms of the weighted blocking probability. As it can be seen in table (5.6) at $k = 2$ the optimum result for the P_b is 2.7551×10^{-5} at the threshold value T^* [6, 4, 5, 4]. This is the same P_b achieved in the previous results. Note that we have two different vectors with the same result. This is because we have $\rho_1 = \rho_2 = \rho_3 = \rho_4$, then any combination of the elements in the vector would give this P_b (see equations (5.32) and (5.33)).

k	T	P_b
0	[3,4,2,4]	4.39858×10^{-3}
1	[3,4,3,4]	5.73143×10^{-4}
2	[3,4,3,4]	5.73143×10^{-4}
3	[3,4,4,4]	3.13862×10^{-4}
4	[3,4,5,4]	3.00763×10^{-4}
5	[3,4,6,4]	3.00235×10^{-4}
6	[3,4,7,4]	3.00217×10^{-4}
7	[3,3,10,4]	5.59497×10^{-4}
8	[3,3,11,3]	8.18777×10^{-4}

Table 5.5: At $\eta = 10$

k	T	P_b
0	[3,4,2,4]	4.39858×10^{-3}
1	[4,4,5,4]	4.14825×10^{-5}
2	[6,4,5,4]	2.78551×10^{-5}
3	[6,3,7,4]	2.8659×10^{-4}
4	[5,3,9,3]	5.46397×10^{-4}
5	[5,2,10,3]	4.37183×10^{-3}
6	[5,2,10,3]	4.37183×10^{-3}
7	[5,2,10,3]	4.37183×10^{-3}
8	[5,2,10,3]	4.37183×10^{-3}

Table 5.6: At $\eta = 25$

k	T	P_b
0	[3,4,2,4]	4.39858×10^{-3}
1	[5,3,9,3]	5.46397×10^{-4}
2	[7,1,11,1]	8.33333×10^{-2}
3	[7,0,11,2]	0.254098
4	[7,0,11,2]	0.254098
5	[8,0,11,1]	0.291667
6	[8,0,11,1]	0.291667
7	[9,0,11,0]	0.5
8	[9,0,11,0]	0.5

Table 5.7: At $\eta = 50$

k	T	P_b
0	[3,4,2,4]	4.39585×10^{-3}
1	[5,2,11,2]	8.19727×10^{-3}
2	[5,3,10,2]	4.37183×10^{-3}
3	[6,3,10,1]	4.19396×10^{-2}
4	[9,2,9,0]	0.254098
5	[10,2,8,0]	0.254098
6	[10,2,8,0]	0.254098
7	[11,1,8,0]	0.29167
8	[11,1,8,0]	0.29667

Table 5.8: At $\eta = 75$

Threshold-based call admission policies have been considered for circuit switched networks and an algorithm has been developed for adjusting the threshold parameters online, the objective being to minimize a weighted sum of call blocking probabilities. In addition to this, the main advantages of this threshold-based admission control scheme lie in its implementational simplicity, and the facts that: it is completely

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

distributed in nature; it is adaptive in the sense that it can automatically adjust the thresholds as the operating conditions change; and it does not need any explicit distributional assumptions.

Central to this admission control scheme is the Marked/Phantom channel algorithm developed in section 5.2.2 for online estimation of the sensitivity of the call blocking metric defined earlier with respect to the thresholds. An advantage that the Marked/Phantom Channel algorithm has is that no special distributional modeling assumptions are required, because this algorithm is based on directly observable network data. Another advantage is that this approach is not just limited to the call blocking probability metric. Similar admission control problems can be formulated with more general cost functions or with multiple objectives if sever traffic classes are to be explicitly modeled.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 6

The Network Model and Simulation

The network model which is used for our simulations is based on routing decisions. The simulator is then composed of a Traffic Generator, Shortest Path Routing, Wavelength Assignment, and Discrete Event Simulation (will be explained in more detail later in this chapter). The traffic generator uses the Poisson distribution for call arrivals, and exponential distribution for holding times.

6.1 Network Model

The communication between nodes is done of peer-to-peer type. All nodes in the network share a global memory module that holds the states of the communication links between network nodes. On the other hand, the routing decisions are distributed. Having a central node computing the shortest path or secondary path for each message on each node would cause the network to suffer from a considerable delay. Each node plans the path on its own using the shared information in the global memory module. That way, the frequently updated states in the global memory and the distributed routing decisions together would avoid the problem of uncertainty in network states and the processing delays for path planning.

Advantages

The advantage of having this global shared memory module is that all nodes would have up to date information about the different network state parameters.

Disadvantages

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The disadvantage is the deadlock that may happen while having this module being accessed from more than one node. Even if the deadlock is prevented, there still may be a bottleneck accessing this memory module. However in this case, the number of nodes in the optical network is not high enough to cause hazardous events that actually affects the performance of the network. Another disadvantage is the complexity of implementing the connection database.

6.1.1 Dijkstra's Algorithm

One possible method for selecting a route through a network is to use a least cost algorithm, which chooses a route that minimizes the sum of the costs of all the communication paths along that route. If one could find the route with the smallest sum, one would find the least cost route. An algorithm such as Dijkstra's (Tanenbaum 1996) is performed by each node, and the results are stored at the node and sometimes shared with other nodes (for pseudo code see appendix C).

6.1.2 Network Simulator Flow Chart

The flow chart of the C++ network simulator is presented in figure (6.1). The program first starts by reading the network graph that is entered. The network graph consists of the number of nodes and the links connecting each node. A wavelength database is built for each link. On each link there exists N number of wavelengths (channels). A variable describes the nature of each wavelength on each link (*i.e.*, either reserved or free). A traffic generator is used to generate the traffic (M number of calls). For each call there exists the following information: 1. The source, and destination, 2. The arrival time and duration of the call. Each call is entered in the Dijkstra's algorithm, in order to obtain a path (route) between the source and destination. A message is sent from the source to the destination to determine which wavelengths are available on the specified route obtained by Dijkstra. Upon receiving the message by the destination node, the free wavelength[1] is selected and a message

[1]The wavelength is selected by first fit scheme (please see chapter 2 for definition).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is sent back to the source, and reserves the specified wavelength on all the links it passes. If there is no channel available, then the call is blocked and registered as a blocked call. Before the program moves on to process the next call request, it first checks if there are current calls that have ended. If there are, then the channels that were reserved by the ended call get released for other calls to use.

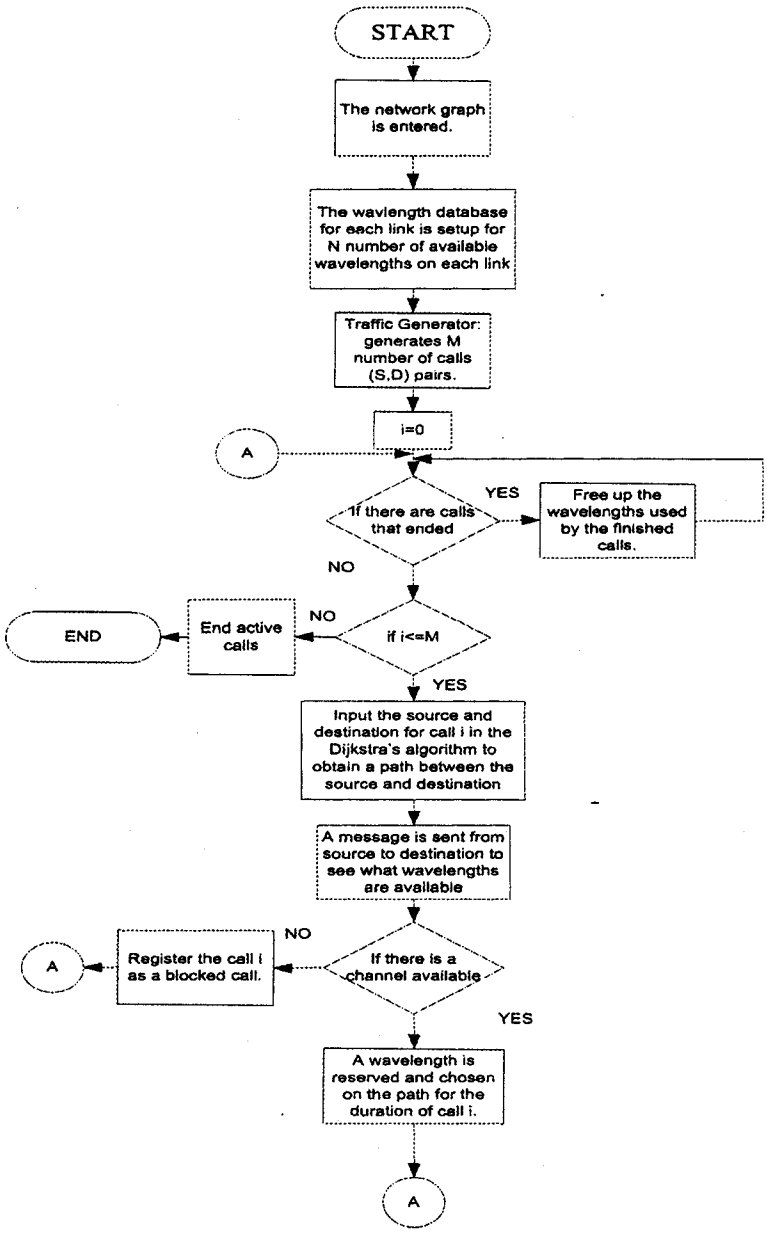

Figure 6.1: Network Simulator Flow Chart

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2 Random Variables

A random variable can be thought of as the numeric result of operating a non deterministic mechanism or experiment to generate a random result. Mathematically, a random variable is defined as a measurable function from a probability space to some measurable space. This measurable space is the space of possible values of the variable, and it is usually taken to be the real numbers. For a continuous random variable X, the probability that X lies in the interval $[a, b]$ is given by

$$P(a \leq X \leq b) = \int_a^b f(x)dx \tag{6.1}$$

Where $f(x)$ is the probability density function.

6.3 Poisson Distribution for Call Arrivals

Poisson distribution is given by equation 6.2.

$$p(x) = \begin{cases} \frac{e^{-\alpha}\alpha^x}{x!}, x = 0, 1, ... \\ 0, \quad otherwise \end{cases} \tag{6.2}$$

Where, α is the number of occurrences, and $p(x)$ is the probability that x occurs.

Traffic, in general, may be very busy and has to slow down and wait, or it may be very light with little slowing or blockage. Facilities, such as roads, telephone lines, toll booths, service agents, and bank tellers, may be either under or over utilized causing costly idle time or poor service to customers. The Poisson distribution explains that call arrivals will always tend to be clumped together and will not arrive in an even manner.

A relationship exists which links the inter-arrival time and the arrival time characteristics. In the case where the inter-arrival time is Exponentially distributed, then the arrival time would be Poisson distributed. The actual arrival time is obtained by adding the inter-arrival time to the previous arrival time. The inter-arrival time is obtained using the following equation:

$$Y_i = \frac{1}{\phi} \times log\frac{1}{1 - x}, i = 0, 1, 2, ... \tag{6.3}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Where ϕ^{-1} is the mean, i represents the number of calls on hand and x is the random variable and Y_i is the ith inter-arrival time.

6.4 Exponential Distribution for Holding Times

The holding time[2] resembles the exponential distribution, because call lengths vary they can be short or long, but very seldom that they will be close to the average length of all connections' times experienced in a day. Therefore, by using the exponential delay, one can accurately simulate a holding time for a connection. The Exponentially distributed random variable is obtained using the following equation:

$$y = e^{-x\lambda} \tag{6.4}$$

Where, x is the random variable with normal distribution and λ is the mean. This is then used to generate the holding time.

The random variables are generated using the C++ program. First, a normal distribution variable is generated using the built in commands "srand" and "rand". After getting a random number with a normal distribution, it then uses that to get the Poisson distributed and Exponentially distributed random variables.

6.5 Discrete Event Simulation

Discrete event simulation is a construction method for simulation programs. The simulated time runs through a sequence of moments when discrete events occur. The method is considered to be more efficient, but on the other hand, more difficult to handle than time step simulation[3]. It has been given preference in most modern discrete simulation programs.

The discrete event simulation (Banks *et al.* 1996) divides the process to be emulated into a sequence of events. An event is a change of a process state that is modeled by

[2]The holding time of a call refers to the duration of the call.

[3]The idea of time step simulation is to divide the simulated time into intervals of the same length and to recalculate all model variables at the end of each of these intervals.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a sudden change of a model variable. Events are organized in a list that contains the time of each event in ascending order. In this way, sub-processes that run parallel are represented with a sequential structure. The selected time unit can be freely chosen without influence on the number of calculations.

Discrete simulation systems today predominantly use the principle of the discrete event simulation. It is more universal than time step simulation. In fact, the time step simulation can be regarded as a special case of discrete event simulation.

6.6 Implementation Approach

The evaluation of the congestion control and call admission control algorithms is based on an event driven simulator consisting of two parts; the first deals with the routing and resource reservation, the second deals with the congestion control and admission control.

- A call connection request arrives with the following information (Source, Destination, Duration).
- The source node then calculates the route using the desired path selection algorithm which in this case is the Least Cost Path which is calculated using Dijkstra's algorithm (see chapter 3).
- The source node sends a control message to check the available resources on that route.
- Upon receiving this control message by the destination node, the destination node then chooses the available channel. In this case, the wavelength assignment was following the first fit criteria (see chapter 2).
- The message is then sent back down the route, and reserves the channel on all the links on this route for the duration of the call.
- The source begins transmission upon receiving the control message.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- If there is no channels available then the call is blocked.
- When the call ends, the channel that was used by the call is freed up to be used by other incoming call requests.
- **For the admission and congestion control**, depending on which is being applied, the congestion control is implemented at the server level in which the server will communicate with the nodes via a dedicated system channel, while the admission controller is applied at each node.

This is done for every call request. For statistical data collection whenever a call is blocked it is recorded.

In summary, the complete simulator that is built using C++, has been presented in this section. The arrival time and holding time have been calculated using the formulas presented. It has been explained in detail how the call is handled from the moment it arrives till it ends.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 7

Conclusion and Future Work

The WDM optical network is shown to be the promising technology which can meet the demands for more communication bandwidth and network resources. This thesis examines the WDM optical network technology from different perspectives. The RWA has been presented, and it is shown that the method of choosing the wavelength can affect the network performance.

Three path selection algorithms (LDP, HZ_1, and DCCR) are closely examined, in which two of them (DCCR and HZ_1), can operate under multiple constraint paths. It is shown that the optimum path chosen between the source and destination can be obtained, even when constraints are negatively correlated. The trade off between the two algorithms, HZ_1 and DCCR, is the computational time depending on which weight function is used (linear or non-linear weight function).

The congestion controller presented uses fuzzy logic to decrease the blocking ratio, at the expense of adding more delay to the incoming calls. It is shown through a series of simulations in chapter 4, that the controller decreases the blocking ratio in order to meet the desired setpoint. The drawback of this controller is that it does not ensure full link utilization, and that there is a need for the controller to be reconfigured every time the network changes. The price to pay when using this controller, is the delay it puts on the calls. In some cases, this option is not valid due to the sensitivity of the traffic to the slightest delay. Another point to keep in mind is that the network is running on 50% traffic load.

The admission controller presented uses threshold-based call admission policies for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

circuit-switched networks. An algorithm is presented for adjusting the threshold parameters online. The goal of this algorithm is to minimize the weighted sum of call blocking probabilities. The main advantage of this algorithm lies in its implementational simplicity. The optimizing approach used is based on the surrogate problem method as described in chapter 5, section 5.1. This approach requires the estimate of the gradients with respect to the surrogate control parameters, which is achieved by applying the marked/phantom slot algorithm. The downfall of this algorithm is that, it does not ensure the value obtained from the blocking ratio to be the least minimum solution. Other minimum solutions might appear, as shown in the graphs in chapter 5. The main advantages that this controller achieves are that it ensures full link utilization; it is adaptive in the sense that it will give the proper threshold vector, even if traffic changes; it ensures that resources are not wasted; and it follows a certain fairness policy.

An interesting area for future work would be to modify the congestion controller using the fuzzy logic, in order to insure full link utilization and configure it so that the network is run on 85 or 90% traffic load. The combination of the admission and congestion controllers in one network might lead to interesting results, especially in affecting the network blocking probability. An area of vast interest is the full link restoration and path restoration techniques in WDM networks, which could be explored and implemented with the admission and congestion controllers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 8

Appendices

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A

The DCCR Algorithm Pseudo Code

RoutingDCCR($G(V,E), s, d, \Delta_d, \Delta_c$,D,C,k)

1. /*Each node u has k records, $(D, C, W, \pi$nd,πidx,mark), which is stored in $ND(u, idx)$, where πidx points to the predecessor's record of that path. A min-heap MH is maintained by increasing weight order, each heap item has the form (n_id, wgt, idx)*/
2. Set $C_{best} \leftarrow \infty, P \leftarrow nil$
3. **InitializeSingleSource**(G, s, ND, MH, k)
4. **HeapInsert**(MH,(s,0,1)) /*Searching start from s */
5. **while** $MH \neq 0$
6. $(u, wgtu, idx_u) \leftarrow$ **HeapEXtractMin**(MH)
7. $ND(u, idx_u)$.mark=VISITED
8. **if** $u = d$ /*found a path p*/
9. $C(p) \leftarrow \sum_{l \in P} c(l)$ /*trace back this new path p and compute its cost */.
10. $P \leftarrow p \bigcup P$
11. **if** $C(p) < C_{best}$
12. $C_{best} \leftarrow C(p), p_{best} \leftarrow p$
13. **if** $u = d$ and $|P|$=k /*Tried k shortest paths*/
14. **Return** p_{best}
15. **for** each vertex $v \in Adj[u]$ /*relaxation*/
16. $(W(v), D(v), C(v)) \leftarrow$ **ComputeWeight**(u, idx_u, v)
17. $(idx_v, w_{max}) \leftarrow$ **FindMax(ND,v)** /*Find path to v with max weight*/

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

18. **if** $W(v) < w_{max}$ and path idx_v is not dominated

19. ND(v,idx_v) $\leftarrow (D(v), C(v), W(v), u, idx_u$,UNVISITED)

20. i$\leftarrow$ **HeapSearch**(MH, v, idx_v)

21. **if** $i \neq nil$ /*Update heap records*/

22. **HeapReplace**$(MH, i, (v, W(v), idx_v))$

23. **else HeapInsert**$(MH, (v, W(v), idx_v))$

InitializeSingleSource(G, s, ND, MH, k)

1. **for** each node u $\in G$

2. **for** $i \leftarrow 1$ to k **do**

3. $ND(u, i).(D, C, W, \pi_{nd}, \pi_{idx}, mark) \leftarrow (\infty, \infty, \infty, nil, nil,$ UNVISITED)

4. $ND(s, 1).W = 0$

ComputeWeight(G, s, ND, MH, k)

1. $D(v) \leftarrow ND(u, idx).D + d(u, v), C(v) \leftarrow ND(u, idx).C + C(u, v)$

2. Compute $W(v)$ as defined in Equation 3.1

3. Return $(W(v), D(v), C(v))$

FindMax(ND, u)

1. **Return** $(idx, ND(u, idx).W)$ where idx is the index of the path with maximum weight and $ND(u, idx).mark$=UNVISITED

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix B

HZ_1 Algorithm Pseudo Code

The HZ_1

INPUT

$G(V, E)$=graph, S=source node,

d=destination node,

Δ=application specified delay bound,

D=link delay function,

C=link cost function.

DJK=Shortest Path Algorithm (e.g. Dijkstra's algorithm)

OUTPUT

A delay bounded path from source s to destination d.

RoutingHZ$(G(V, E), s, d, \Delta, D, C, DJK)$

1. Call DJK(G, s, d, D) to compute the least delay path, store it in LDP
2. Call DJK(G, s, d, C) to compute the least cost path, store it in LCP
3. **if** $D(LDP) < \Delta$
4. **Return** FAILED.
5. **if** $D(LCP) \leq \Delta$
6. **Return** LCP. /* LCP is a feasible path */
7. $Set \alpha \leftarrow C(LDP) - C(LCP), \beta \leftarrow$D(LCP)-D(LDP)

   $\gamma \leftarrow D(LCP) * C(LDP) - D(LDP) * C(LCP)$.

   compute $w(e) \leftarrow \alpha * d(e) + \beta * c(e)$ for each $e \in E$.

   Call DJK(G, s, d, W) to compute the least weight path, store it in LWP.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8. **if** $W(LWP)=\gamma$

9. **if** $D(LWP) \leq \Delta$

10. **Return** LWP

11. **else** /* $D(LWP) > \Delta$

12. **Return** LDP

13. **if** $W(LWP) < \gamma$

14. **if** $D(LWP) \leq \Delta$

15. $LDP \leftarrow LWP$

16. **else** /* $D(LWP) > \Delta$ */

17. LCP $\leftarrow LWP$

18. Go to step 7.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix C

Dijkstra's algorithm pseudo code

```
#define MAX_NODES 1024
#define INFINITY 10000000000
int n, dist[MAX_NODES]{MAX_NODES};
void shortest_path(int s, int t, int path[ ])
{struct state {
        int predecessor;
        int length;
        enum{permanent, tentative}label;
 }state[MAX_NODES];
 int i, k,min;
 struct state *p;
 for (p=&state[0]; p<&state[n]; p++){
        p->predecessor=-1;
        p->length=INFINITY;
        p->label=tentative;
 }
 state[t].length=0; state[t].label=permanent;
 k = t;
 do {
        for (i=0;i<n;i++)
          if (dist[k][i]!=0 && state[i].label==tentative {
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            if(state[k].length+dist[k][i]<state[i].length) {
                state[i].predecessor=k;
                state[i].length=state[k].length+dist[k][i];
            }
        }
        k=0;min=INFINITY;
        for(i=0; i<n; i++)
            if(state[i].label==tentative&&state[i].length<min){
                min=state[i].length;
                k = i;
            }
        state[k].label=permanent;
    }while (k!=s)
    i=0; k = s;
    do {path[i++]=k;k=state[k].predecessor;}while(k≥0);
}
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

Al-Hammadi A. and J. Schormans (2001). Fuzzy congestion controller in a prioritized ATM switch. *IEEE Proc.-Commun.*, Vol. 148, No. 2, April, pp. 57-62.

Banks J., J. S. Carson II and B. L. Nelson (1996). *Discrete-Event System Simulation, Second Edition*, New Jersey: Prentice-Hall.

Bellman R. (1958). *On a routing problem*, Q Appl Math, pp 87-90.

Boudriga N. and M. S. Obaidat (2003). Admission Control and Resource Management of Uncertain Duration Traffic in Optical Networks. *IEEE Electronic, Circuits and Systems* Vol. 3, December, pp. 1018-1021.

Bruni C., F. D. Riscoli, G. Koch and S. Vergari (2005). Traffic management in a band limited communication network: an optimal control approach. *International Journal of Control* Vol. 78, No. 16, 10 November, pp. 1249-1264.

Chen Bor-Sen, Sen-Chueh Peng, and Ku-Chen Wang (2000). Traffic Modeling, Prediction, and Congestion Control for High-Speed Networks: A Fuzzy AR Approach. *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 5, October, pp. 491-508.

Chen S. and K. Narstedt (1998). On Finding Multi-Constrained Paths. *1998 IEEE conference on communications* Vol. 2, June, pp. 874-879.

Choi J. S., N. Golmie, F. Lapeyrere, F. Mouveaux, and D. Su (2000). A Functional Classification of Routing and Wavelength Assignment Schemes in DWDM networks: Static Case. *Proceedings of the 7th International Conference on Optical Communications and Networks, OPNET* January, Paris, France, pp. 1109-1115.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chong E.I., S. Maddila and S. Morley (1995). On Finding Single Source Single Destination k Shortest Paths. *Proceedings of the International Conference on Computing and Information*, pp. 40-47.

Chrysostomou C., A. Pitsillides, A. Sekercioglu, and M. Polycarpou (2003). Fuzzy Explicit Marking for Congestion Control in Differentiated Services Networks. *Proceedings of the Eighth IEEE International Symposium on Computers and communication*, Vol. 1, pp. 312-319.

Chrysostomou C., A. Pitsillides, G. Hadjipollas, M. Polycarpou, and A. Sekercioglu (2004). Congestion Control in Differentiated Services Networks using Fuzzy Logic. *43rd IEEE Conference on Decision and Control*, Vol 1, pp. 549-556.

Cirstea M. N., A. Dinu, J.G. Khor, M. McCormick (2002). *Neural and Fuzzy Logic Control of Drives and Power Systems*, Esevier Science, Linacre House, Jordan Hill, Oxford.

Deb Supratim and R. Srikant (2004). Congestion Control for Fair Resource Allocation in Networks with Multicast Flows. *IEEE/ACM Transactions on Networking*, Vol. 12, No. 2, pp. 274-285.

Davoli F. and P. Maryni (2000). A Two-Level Stochastic Approximation for Admission Control and Bandwidth Allocation. *IEEE Journal on Selected Areas in Communications* Vol. 18, No. 2, February, pp. 222-233.

Garey M.R. and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

Gokbayrak K. and C. G. Cassandras (2001). Online Surrogate Problem Methodology for Stochastic Discrete Resource Allocation Problems. *Journal of Optimization Theory and Applications* Vol. 108, No. 2, pp. 349-376.

Gokbayrak K. and C. G. Cassandras (2002). Adaptive Call Admission Control in Circuit-Switched Networks. *IEEE Transactions on Automatic Control* Vol. 47, No. 6, pp. 1004-1015.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Guo L. and I. Matta (2003). Search Space Reduction in QoS Routing. *Computer Networks*, Vol. 41, pp. 73-88.

Haas Z. (1991). Adaptive Admission Congestion Control. *Computer Communication Review*, Vol. 21, No. 5, pp. 58-76.

Handler G.Y. and I. Zang (1980). A Problem Dual Algorithm for the Constrained Shortest Path. *Networks*, Vol. 10, pp. 293-310.

Imer O.C., T. Basar, and R. Srikant (2000). A Robust Adaptive Algorithm for ABR Congestion Control in ATM Networks. *9th IEEE International Conference on Computer Communication and Network*, Las Vegas, Nevada.

Kim D. K., P. Park, and J. W. Ko (2004). Output-Feedback $H\infty$ Control of Systems Over Communication Networks Using a Deterministic Switching System Approach. *Automatica*, Vol. 40, Issue 7, July, pp. 1205-1212.

Kim H. S., S. Y. Shin, and W. H. Kwon (2004). Feedback Control for QoS of Mixed Traffic in Communication Networks. *Control Engineering Practice*, Vol. 12, pp. 527-536.

Korkmaz T. and M. Krunz (1999). A Randomized Algorithm for Finding a Path Subject to Multiple QoS Constraints. *Global Telecommunications Conference-Globecom'99*, pp. 1694-1698.

Korkmaz T. and M. Krunz (2001). Multi-Constrained Optimal Path Selection. *IEEE INFOCOM*, pp. 834-843.

Kosko B. (1997). *Fuzzy Engineering*, Upper Saddle River, N.J: Prentice Hall.

Kuipers F., P. V. Mighem, T. Korkmaz, and M. Krunz (2002). An Overview of Constraint-Based Path Selection Algorithms for QoS Routing. *IEEE Communications Magazine*, December, pp. 50-55.

Leonardi E., F. Neri, M. Gerla, and P. Palnati (1996). Congestion Control in Asynchronous, High-Speed Wormhole Routing Networks. *IEEE Communications Magazine*, November, pp. 58-69.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Liu Zhixin and Xinping Guan (2004). A Flow Congestion Control Scheme of ATM Networks Based on Fuzzy PID Control. *Proceedings of the* 5^{th} *World Congress on Intelligent Control and Automation*, June, pp. 1466-1469.

Liu G. and K.G. Ramakrishnan (2001). A*Prune: An Algorithm for Finding K shortest Paths Subject to Multiple Constraints. *IEEE INFOCOM*, pp. 743-749.

Liu H. K., J. B. Wilson, and J. Y. Wei (2000). A Scheduling Application for WDM Optical Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 18, October, pp. 2041-2050.

Loukas Rossides, Stefan Kohler, Pitsillides Andreas, Tran-Gia Phuoc (2000). Fuzzy RED: Congestion control for TCP\ IP Diff-Serv. *IEEE* 10^{th} *Mediterranean Electrotechnical Conference*, Vol. I, pp.19-22.

Ma M. and M. Hamdi (2000). Providing Deterministic Quality of Service Guarantees on WDM Optical Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 10, October, pp. 2072-2083.

Mascolo S. (1999). Congestion Control in High-Speed Communication Networks Using the Smith Principle. *Automatica*, Vol. 35, pp. 1921-1935.

Mosharaf Kayvan, J. Talim, I. Lambadaris (2004). A call Admission Control for Service Differentiation and Fairness Management in WDM Grooming Networks. *IEEE Proceedings, BroadNets*, pp. 162-169.

Mukherjee B. (2000). WDM Optical Communications Networks: Progress and Challenges. *IEEE Journal on selected areas in communications*, October, Vol. 18 No. 10, pp. 1810-1824.

Murthy C. and M. Gurusamy (2001). *WDM Optical Networks: Concepts, Design, and Algorithms.* Prentice Hall PTR.

Ramaswami R. and A. Segall (1997). Distributed network control for optical networks. *IEEE/ACM Trans. Networking*, Vol. 5, pp. 936-943.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Smiljanic Aleksandra (2002). Flexible Bandwidth Allocation in High-Capacity Packet Switches. *IEEE/ACM Transactions on Networking*, April, Vol. 10, No. 2, pp. 287-293.

Sycamore Networks. www.sycamorenet.com/index.asp.

Tan L. and S.H.Yang (2002). Rate-Based Congestion Controllers for High-Speed Computer Networks. *Proc. of the Int'l Federation of Automatic Control 15th IFAC World Congress 2002.*

Tanenbaum A. (1996). *Computer Networks, Third Edition.* Prentice-Hall.

Verbruggen B. H. and R. Babuska (1999). *Fuzzy Logic Control: Advances in Applications*, Sigapore; River Edge, NJ: World Scientific series Vol. 23.

White Curt (2002). *Data Communications & Computer Networks, A Business User's Approach, Second Edition*, Course Technology, United States of America, Second Edition.

Yuan X. (2002). Heuristic Algorithms for Multi-constrained Quality-of-Service Routing. *IEEE/ACM Transactions on Networking*, Vol. 10, NO. 2, pp.244-256.

Zak Stainslaw H. (2003). *Systems and Control*, New York, Oxford University Press.

Zhao W. and W. Jia (1999). Efficient Adaptive Connection Admission Control Algorithms for Real-Time ATM LANs. *Control Engineering Practice*, Vol. 7, pp. 1525-1532.

Zhou B. and H. Mouftah (2002). Adaptive Least Loaded Routing for Multi-fiber WDM Networks Using Approximate Congestion Information. *IEEE International Conference on Communications*, Vol. 5, May, pp. 2745-2749.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.