PERFORMANCE EVALUATION OF ONLINE CALL ROUTING AND ADMISSION CONTROL ALGORITHMS

Yves Saintillan

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

December 1998

0-612-43558-X

Abstract

Performance Evaluation of Online Call Routing and Admission Control Algorithms

Yves Saintillan

We consider the problem of call routing and admission control in general topology networks. Given a network, a *call request* consists of an origin-destination pair, and a bandwidth requirement. For each request, the routing algorithm must find a path in the network satisfying the bandwidth requirement, and the admission control algorithm must decide whether or not to accept the call. If the call is to be accepted, the required bandwidth needs to be allocated on the path selected throughout the duration of the call. The goal of the admission control algorithm is to decide online which calls to accept, without prior knowledge of future calls, so as to maximize the network throughput over time.

By using a large set of simulation experiments, we evaluate the performance of online admission control algorithms proposed by several researchers in the context of competitive worst-case analysis. We first analyze the behavior of the EXP algorithm proposed by Gawlick *et. al* [25]. This algorithm, which is based on reserving bandwidth for calls using sufficiently cheap paths (according to a cost function exponential in the link utilizations), has been shown to outperform a simple greedy approach of accepting any calls for which there is a path with the required bandwidth. Our experiments confirm these results; however, the difference in performance was found to be reduced by varying parameters such as the bandwidth requirements, the routing algorithm and the traffic matrix. The performance advantage is also diminished when each request has an additional Quality of Service constraint, in particular, the maximum end-to-end delay. Furthermore, on the networks simulated, we show that the behavior of EXP is much more unfair than the behavior of the greedy algorithm along several dimensions. We propose small variations to the admission control algorithm that each greatly reduce the unfairness of EXP on a dense commercial network topology, without affecting the throughput. Additionally, we conducted experiments on the recently proposed KPP algorithm [42]. In our results, the KPP algorithm does not outperform EXP (despite its higher competitive ratio), but always achieves better fairness.

Acknowledgments

I would like first to thank my supervisor Dr. Lata Narayanan for her support, help, and constant encouragement throughout this thesis.

Part of this work was done at the National Institute of Standards and Technology. I am indebted to Dr. David Su, Manager of the High-Speed Networking Group, for his support.

I would like to thank Nada Golmie for providing the NIST ATM Network Simulator used in our experiments. I wish to thank also Rainer Gawlick for providing the traffic matrix and topology of the commercial network used in the simulations. I am grateful for the machine time donated by the Softeks lab at Concordia to run some of the early simulations reported in this thesis.

Finally I am grateful to all the members of my family for their affection and love.

Contents

List of Figures ix

List of Tables xiii

1 Introduction 1

1.1 Scope of Thesis 5

1.2 Contributions 6

1.3 Organization of Thesis 7

2 Survey of Scheduling, Routing and Admission Control Algorithms 9

2.1 Preliminaries 9

2.1.1 QoS Parameters 9

2.1.2 Connection Characteristics 10

2.1.3 Performance Measures 10

2.1.4 Competitive Analysis 11

2.2 Scheduling Algorithms 11

2.2.1 FIFO 12

2.2.2 Weighted Fair Queueing 12

2.2.3 Virtual Clock 14

2.2.4 Stop-and-Go 14

2.2.5 Other Scheduling Algorithms 15

2.2.6 Comparison of scheduling algorithms 15

2.3 Routing Algorithms 16

2.3.1 Routing in Telephone Networks 16

2.3.2 Routing in Traditional Data Networks 17

2.3.3 QoS Routing 18

2.4 Admission Control Criteria 21
2.4.1 Phone Networks 21
2.4.2 Known Traffic Matrix on General Topologies 21
2.4.3 Competitive Analysis Approach 23
2.4.4 Comparison of Admission Control Algorithms 26

3 Experimental Results **28**
3.1 Organization of Experiments 31
3.1.1 Routing Algorithms 31
3.1.2 Scheduling Criteria 33
3.1.3 Admission Control Criteria 34
3.1.4 Combination of Routing Algorithm with Admission Control and Scheduling Criterion 35
3.1.5 Type of Load 35
3.1.6 Type of Delay Constraint 36
3.1.7 Different Profit Measures 36
3.1.8 Topology and Traffic Matrix 36
3.2 Results of Experiments 39
3.2.1 Analysis of GREEDY and EXP Strategies 39
3.2.2 Evaluation of Other Admission Control Algorithms 52
3.2.3 Delay-Constrained Case 63
3.2.4 Effect of Job Size 68

4 Simulator Implementation **73**
4.1 The NIST ATM Network Simulator 73
4.1.1 Functionalities 73
4.1.2 Architecture 73
4.2 Changes Made to the Simulator 74
4.2.1 New Functionalities 74
4.2.2 Changes Made to the Architecture/Design Document 76
4.2.3 Tests and Validation 79

5 Conclusions and Future Work **80**

A Traffic Matrices **82**

A.1 Matrix for NSFNet Topology 82

A.2 Matrix for ComNet Topology 83

B Algorithm Implementation **84**

B.1 MIN-HOP-1 Algorithm 84

Bibliography **86**

List of Figures

1 Admission control event trace 30
2 NSFNet topology 37
3 Existing commercial network topology 38
4 Varying the critical utilization on NSFNet for EXP: mean blocking probability (left) and standard deviation of blocking probabilities (right) 39
5 Varying the critical utilization on ComNet for EXP: mean blocking probability (left) and standard deviation of blocking probabilities (right) 40
6 GREEDY vs EXP: num hops vs load 40
7 GREEDY vs EXP: average path length vs load 41
8 GREEDY vs EXP: calls vs length at load 850 41
9 Distribution of the percentage of rejected requests for node pairs on the NSFNet topology for GREEDY (left) and EXP (right) 43
10 Distribution of the percentage of rejected requests for node pairs on the ComNet topology for GREEDY (left) and EXP (right) 43
11 Utilization of bottleneck link 5 to 6 for GREEDY (left) and EXP (right) 47
12 Utilization of bottleneck link 6 to 7 for GREEDY (left) and EXP (right) 48
13 Utilization of bottleneck link 10 to 11 for GREEDY (left) and EXP (right) 48
14 Effect of traffic matrix on the NSFNet topology: random matrix . . . 50
15 Effect of traffic matrix on the NSFNet topology: 80% of all calls are 1-hop calls (left) and 2-hop calls (right) 50
16 Effect of traffic matrix on the NSFNet topology: 80% of all calls are 3-hop calls (left) and 4-hop calls (right) 50
17 Effect of traffic matrix on the NSFNet topology: 80% of all calls are 5-hop calls 51

18 Effect of traffic matrix on the behavior of EXP on the NSFNet topology with 80% of all calls being 5-hop calls: varying the critical utilization at load 400 Erlangs (left) and at load 1100 Erlangs (right) 51

19 GREEDY vs EXP: 4_5_hops traffic 51

20 Effect of the routing algorithms on the NSFNet topology for GREEDY (left) and EXP (right) 53

21 Effect of routing algorithms on the Commercial Network Topology for GREEDY (left) and EXP (right) 53

22 Varying the critical utilization for POWER(k) on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 55

23 Varying the critical utilization for POWER(k) on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 55

24 Comparison of POWER(10)(with a critical utilization of 93%) with EXP (critical utilization of 94%) on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) . . . 55

25 Comparison of POWER(15)(with a critical utilization of 99%) with EXP (critical utilization of 97%) on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) . . . 56

26 Distribution of the percentage of rejected requests for each switch pair for POWER(15) at load 3600 Erlang on the ComNet topology 56

27 Varying μ for different ρ (or critical utilization) values for EXP on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 57

28 Comparison of EXP 94% with EXP using $(\mu, \rho) = (19972, 1.05e6)$ on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 58

29 Comparison of EXP 97% with EXP using $(\mu, \rho) = (1.08e10, 1.72e10)$ on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 58

30 Distribution of the percentage of rejected requests for each switch pair for EXP using $(\mu, \rho) = (1.08e10, 1.72e10)$ at load 3600 Erlang on the ComNet topology 59

31 Varying the parameter a for the KPP algorithm on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 60
32 Varying the parameter a for the KPP algorithm on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 60
33 Comparison of EXP (94%) with KPP^* on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 61
34 Comparison of EXP (97%) with KPP^* on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right) 61
35 Distribution of the percentage of rejected requests for each switch pair for KPP^* at load 3600 Erlang on the ComNet topology 61
36 Effect of 2-hop delay constraints on NSFNet for GREEDY (EXP has identical performance) 64
37 Effect of 3-hop delay constraints on NSFNet for GREEDY (left) and EXP (right) 64
38 Effect of 4-hop delay constraints on NSFNet for GREEDY (left) and EXP (right) 65
39 Effect of 5-hop delay constraints on NSFNet for GREEDY (left) and EXP (right) 65
40 Effect of mix delay constraints on NSFNet for GREEDY (left) and EXP (right) 65
41 Effect of 2-hop delay constraints on ComNet for GREEDY (left) and EXP (right) 66
42 Effect of 3-hop delay constraints on ComNet for GREEDY (left) and EXP (right) 66
43 Effect of 4-hop delay constraints on ComNet for GREEDY (left) and EXP (right) 66
44 Effect of 5-hop delay constraints on ComNet for GREEDY (left) and EXP (right) 67
45 Effect of mix delay constraints on ComNet for GREEDY (left) and EXP (right) 67
46 Effect of big jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right) 70

47 Effect of big jobs on NSFNet at low loads with 4-hop delay constraint for GREEDY (left) and EXP (right) 70
48 Effect of small jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right) 71
49 Effect of heterogeneous jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right) 71
50 Effect of big jobs on ComNet with 4-hop delay constraint for GREEDY (left) and EXP (right) 71
51 Effect of small jobs on ComNet with 4-hop delay constraint for GREEDY (left) and EXP (right) 72
52 Architecture of the simulator 74
53 Simulator screen 75
54 Interfaces 78

List of Tables

1 Default parameters 38
2 Percentage of rejected requests for calls at 1 hop distance 45
3 Percentage of rejected requests for calls at 2 hop distance 45
4 Percentage of rejected requests for calls at 3 hop distance 46
5 Percentage of rejected requests for calls at 4 hop distance 46
6 Percentage of rejected requests for calls at 5 hop distance 47
7 Utilization of Links 47
8 Critical utilizations (in %) for the ComNet topology 52
9 μ as a function of the critical utilization 54
10 Comparison of admission control algorithms on the ComNet topology at load 3600 Erlangs 62
11 Critical utilizations for the NSFNet topology, given for each delay constraint value. "2h" refers to the case of two-hop delay constraint . . . 63
12 Critical utilizations for the ComNet topology 64
13 Critical utilizations (in %) for small, large, and heterogeneous jobs on the NSFNet topology 69
14 Critical utilizations (in %) for small and large jobs on the ComNet topology 69
15 Switches of command line 77
16 Traffic matrix for the NSFNet topology 82
17 Traffic matrix for the commercial network topology 83

Chapter 1

Introduction

Computer networks are becoming increasingly faster, with up to gigabits per second rates, due to advances in transmission technologies such as fiber optics [47, 61], and driven by high demand for multimedia services, such as teleconferencing, medical imaging and video-on-demand. These new applications have inherent requirements of network performance, not needed by traditional applications. High-speed networks, such as Broadband-ISDN and in particular ATM [49, 57], need to support these new applications with heterogeneous requirements. The *Quality of Service* parameters of applications can include the end-to-end cell transfer delay (CTD), the cell delay variation (CDV) also called delay *jitter*, the cell loss ratio (CLR), and the bit error rate (BER) [57]. For example, traditional applications such as voice services require a low CTD. Services which do not have critical real-time requirements (eg. File Transfer), called *best effort services*, require only a low CLR. In contrast, multimedia applications, such as teleconferencing, need low CTD and low CDV, but are tolerant to loss.

The problem of traffic control for such networks is an important area of research. In particular the congestion that can occur due to the limitation of network resources may not be acceptable for some applications. In order to prevent congestion, it is necessary to reserve resources and to perform a kind of *admission control* that decides if a connection request can be accepted or not. The admission control is very dependent on the nature of the applications' requirements. For best-effort traffic no admission control and no resource reservation is actually needed. For applications that do not need strict performance guarantees, a dynamic admission control (also called measurement-based) has been proposed [40, 56, 12, 62]. In this scheme, resources are allocated dynamically by measuring the input traffic. However, when the requirements are stringent, a static admission control is needed. It consists of allocating a fixed amount of resource at connection setup. Such reservations can be found in ATM or integrated services IP. In this thesis, we are specifically interested in the static admission control, and the interaction of its three main components: the *routing* component, the *scheduling* component, and the *admission control* criterion. The routing component implements the path calculations necessary to find a path on which to set up the connection. The scheduling component manages queues at intermediate nodes and specifies whether there are enough resources at each node and on the entire path to satisfy the QoS requirements. If the path satisfies the scheduling constraints, locally and end-to-end, it is said to be *QoS-permissible* [52]. The admission control criterion maximizes some notion of profit, for example the network throughput. If the route satisfies this criterion, it is said to be *AC-permissible.* A connection is accepted if and only if there is a path which is at the same time QoS-permissible and AC-permissible. These components are not always independent of each other. For instance, the routing component often strongly depends on the admission control criterion used.

In the rest of this chapter, we will give a brief description of the scheduling, routing and admission control components involved in the QoS performance guarantees framework, and then present the scope of this thesis.

The goal of the scheduling component is to provide performance guarantees by the use of an appropriate strategy for queue management. Depending on the amount of bandwidth reserved, scheduling can provide a deterministic or statistical multiplexing service. In deterministic multiplexing, the peak bandwidth of the application is reserved. Clearly, this ensures that no queueing will ever happen and the queueing delay is thus reduced to zero. However, this approach is quite conservative. Indeed for variable bit rate applications, the resources may not be fully utilized. The other approach, which takes advantage of ATM multiplexing capabilities, consists of reserving an amount of bandwidth that is smaller than the peak rate but greater than the average rate. This quantity is usually referred to as the *effective* bandwidth. The scheduling algorithm then, needs to guarantee that the QoS constraints are respected.

The scheduling algorithm is often associated with traffic characterization. Most of the scheduling algorithms provide guarantees under the assumption that a certain form of traffic characterization or *traffic shaping* [66] is used, such as, for instance, the Token Bucket or the deterministic bounding interval dependent scheme (D-BIND) [45]. Traffic shapers can be used at the network entry only, or at every node in the path [26].

Another issue is what kind of guarantee is provided. The guarantee can be *deterministic* or *statistical*, if the performance criterion has a strict upper bound, or if it is guaranteed to hold with a certain probability (respectively). For example, a well-known strategy called PGPS, which we describe in Chapter 2, can provide either statistical or deterministic guarantees depending on the particular traffic shaping used. Conversely, if a traffic characterization such as D-BIND is used, one can derive a scheduling strategy to provide deterministic [45] or statistical [46] guarantees.

Another way of classifying scheduling disciplines is whether they are *work-conserving* or not. A work-conserving algorithm is one in which the server is always busy whenever there is some packet in the queue. Work-conserving algorithms optimize the utilization of network resources, but on the other hand they can sometimes lead to poor values of delay jitter. The advantage of the disciplines that are not work-conserving is that they can be used to bound the delay jitter. Usually there are tradeoffs between strong guarantees, complexity of implementation, and optimal utilization of network resources.

In our terminology, the role of the routing component is to find a route, or a set of routes, between a specified source and a destination, or between a source and many destinations in the case of multicast. For example, it may find the shortest path which has the maximum available bandwidth. In this thesis, we are interested in routing algorithms for virtual circuit networks, such as in ATM. In contrast to hop-by-hop routing, all the traffic of one connection is routed through the same path. Also, once a connection is established, typically no-rerouting is allowed.

There are several issues related to virtual circuit routing. First, a routing algorithm can be required to find a path between one source and one destination (unicast connections) or between one source and many destinations (multicast connections). Both are found in traditional data networks, but multicast connections are increasingly used to carry multimedia traffic. Another issue is whether the routing algorithm is centralized or distributed, and how often the network state information is updated. The routing is said to be *pre-computed* when paths between all destination pairs are calculated at regular intervals (at the times the network state information is updated).

The routing in *on-demand* when the routing calculation is performed on a per-request basis.

In traditional data networks (i.e. ARPANET, SNA), the routing algorithms mainly consist of a shortest-path calculation, using either Dijkstra's algorithm as in the Open Shortest-Path First (OSPF) protocol, or the Bellman-Ford algorithm. The costs of the paths are usually based on a metric, such as delay, link utilization, or number of hops. The problem of routing for connections with QoS requirements, also called *QoS routing* [76], is more complicated and not well understood. Typically, QoS routing consists of finding a path, or a tree in the case of multicast, satisfying more than one constraint [57]. For example, constraints may be expressed as a combination of bandwidth, cell loss and cell delay requirements. For arbitrary metrics, this problem is NP-hard [76]. Heuristics have been proposed for unicast [69] or multicast [77] connections. However in some cases of metrics the routing problem can be solved in polynomial time. For example, unicast QoS routing techniques have been proposed as an extension of OSPF in the case of delay constraints.

The role of the admission control algorithm is to decide whether to accept or reject a connection request. If there is no path satisfying the QoS requirements, the request is obviously rejected. However if there are one or more feasible paths, two strategies are possible. The simplest consists of always accepting the request. This is called the *greedy* strategy. On the other hand, for the *non-greedy* strategies, the request may be rejected if the *costs* of the feasible paths are too high, where the cost of a path is a function of the goals of the admission control algorithm. Typically, the goal of the admission control algorithm may be the maximization of the profit accrued by the network, which is often proportional to the percentage of accepted calls. Other goals may be to provide fairness to users, or to minimize the average end-to-end delay [57]. If the request is finally accepted, the admission control algorithm needs to choose which of the feasible paths to use, again depending on the performance goal adopted.

The problem of admission control has been extensively studied in the area of circuit-switching networks for telephony [1]. In this context, the only constraint is the bandwidth requirement and the goal is to maximize the number of active calls in the network. Non-greedy strategies, referred to as *reservation-based* strategies, have been proposed and have been shown to outperform greedy strategies. Roughly speaking,

[1]Note that due to the close dependency of routing and admission control, these algorithms are usually denoted as "routing algorithms"

the admission control tries first to route the call on the direct link between the end nodes. If it is not available, alternate paths are tried, but an alternate path is accepted only if the utilization of all the links is below a threshold (or reservation level). The actual order of trial of alternate paths may differ depending on the algorithm.

However, algorithms developed for circuit-switching networks are based on fully-connected topologies, and assume that the arrival pattern of requests is known in advance. Since high-speed networks such as ATM (or integrated services IP) may not be fully-connected, these algorithms are not applicable directly, but some concepts may be re-used. The problem of admission control on general topology networks has been investigated by several researchers. One issue that distinguishes these approaches is whether or not something is known about the arrival patterns. In [71], an alternate routing strategy is proposed, but the knowledge of the traffic matrix is used to compute primary paths. In contrast, a strategy independent of any assumption on the traffic matrix has also been proposed based on a competitive analysis. Awerbuch *et al.* [5] consider the setting where profit is proportional to the bandwidth-duration product. They give an algorithm which is $O(\log LT)$-competitive with the best off-line algorithm (see the definition of competitiveness in Section 2.1.4) , where L is the maximum path length and T is the ratio of the maximum over the minimum circuit holding time. They also give a matching lower bound. In practice, this approach has been found to outperform the greedy strategy [25]. However, the above approaches are concerned only with allocation of bandwidth and do not take into account QoS requirements, such as end-to-end delay requirements.

Finally, it should be noted that non-greedy strategies may not be applicable in some cases. For example, in Permanent Switched Virtual Circuits (such as Virtual Path connections in ATM), where the holding times of circuits are considered to be infinite, only the greedy approach can be employed [24], since the goal in this case is to maximize the amount of bandwidth routed.

1.1 Scope of Thesis

In this thesis, we investigate the performance of routing and admission control algorithms for switched virtual circuits. We assume that the algorithm has no *a-priori* knowledge of the traffic pattern, unless otherwise specified, and that the network can have any general non-hierarchical topology. Our main goal is to analyze the flexibility of the competitive routing and admission control algorithm proposed in [25], to study the performance of other algorithms such as the KPP algorithm [42], and to evaluate the performance of the algorithms in the context of QoS constraints. As a performance measure we use the proportion of rejected requests. However, we also consider *fairness*, a term that has been used in the literature [71, 65] to measure the variation of blocking probabilities over origin-destination pairs.

These simulations are based on two network topologies of different sizes used as benchmarks in the literature [71, 25]. Note that these topologies are based on real networks that do not provide bandwidth guarantees, but carry for example IP traffic. However, broadband networks supporting virtual circuit routing, corresponding to these topologies, will likely be used in the not too distant future to carry multimedia traffic. Other parameters, such as call arrival rates, durations, and link capacities were set to reflect projected video traffic. The traffic matrices reflect the variety of load patterns on the currently existing networks [71]. In our simulations, the model is the same as in [25], except when delay constraints are considered. Rejected requests are considered to be lost. Connections are unicast and the admission control and routing algorithms are implemented in a centralized fashion. Our simulations are run on a modified version of the NIST ATM Network simulator [30].

1.2 Contributions

Our results can be divided into four parts. In the first part, we offer an in-depth comparison of the greedy algorithm based on minimum-hop routing and the non-greedy EXP algorithm proposed in [25]. We look at the switch-to-switch blocking probabilities and evaluate the effects of the reservation level and the traffic matrix. EXP is found to be more unfair than the greedy algorithm on an Origin-Destination pair basis. In terms of overall blocking probability, EXP outperforms Greedy over a wide range of reservation levels, but this advantage is reduced when the traffic matrix is not well matched.

In the second part, we compare the performance of other routing and admission control algorithms. It is shown that for the greedy admission control criterion simulated, the routing function based on costs exponential in the link utilization has the best performance. Note that the advantage of combining greedy admission control with exponential cost routing was mentioned in [24] in the context of permanent virtual circuits, but we differ from the work of [24] in considering switched virtual circuits. In addition we present for the non-greedy approach a new class of algorithms, POWER(k), which consists of defining the cost as a power of the link utilization. We show that instances of this class can outperform EXP in terms of blocking probability and fairness when appropriate parameters are chosen. We show that the generic version of EXP (where $\mu \neq \rho$) can be used to reduce the variation of blocking probabilities. We also simulated the KPP algorithm. Overall, KPP achieves much better fairness than EXP. In terms of overall percentage of rejected requests, KPP is outperformed by EXP on a small topology, but achieves the same performance as EXP on the big topology.

The third part of the results consists of a performance evaluation of the EXP and Greedy algorithms when a QoS parameter, specifically the CTD, is added to the connection request. The CTD is expressed in terms of the maximum number of hops that the route for the connection can have. We say that such a connection request is *delay-constrained.* It is shown that for greedy or non-greedy approaches, the routing algorithm that works the best is the one that picks the path with minimum exponential cost among the ones satisfying the delay constraint. However, the performance advantage of non-greedy over greedy algorithms is reduced when delay constraints are added.

In the last part, we compare the performance of EXP and Greedy when the bandwidth requirement is varied. We observe that the performance advantage of EXP is reduced on both topologies when the size of the bandwidth requirement is increased. However, when the bandwidth requirement is decreased, EXP behaves differently depending on the topology.

1.3 Organization of Thesis

The rest of this thesis is organized as follows: an overview of the related work on scheduling, routing and admission control algorithms, is presented in Chapter 2. In Chapter 3 we present our results. Chapter 4 describes the simulation model and the changes made to the NIST ATM Network Simulator. Conclusions and directions for future work are presented in Chapter 5. Finally, in Appendix A we reproduce the traffic matrices used for the simulations, and in Appendix B we give the pseudocode we propose for the routing algorithm proposed in this thesis.

Chapter 2

Survey of Scheduling, Routing and Admission Control Algorithms

In this chapter, we first give a few definitions to clarify the terminology and issues considered in this thesis. We proceed by presenting a brief survey of the research in the area of admission control algorithms. We describe separately the algorithms proposed in the literature for the scheduling and routing components and the admission control criterion.

2.1 Preliminaries

2.1.1 QoS Parameters

The following QoS parameters are relevant to the discussion in this chapter:

CTD (Cell Transfer Delay) Delay between the departure of the first bit of the cell from the source to the arrival of the last bit to the destination. The CTD is the sum of the *propagation delay* due to the speed of the links, and the *queueing delay* at the nodes. In this thesis, the propagation delay is considered negligible.

CDV (Cell Delay Variation) (or Delay Jitter) Variance of the CTD.

CLR (Cell Loss Ratio) Ratio of the number of cells lost over the total number of cells sent for a given time interval.

2.1.2 Connection Characteristics

In this chapter, we will use the terminology defined by the ATM Forum to classify B-ISDN services.

CBR (Constant Bit Rate) Category of connections sending at a constant cell rate, also called Peak Cell Rate (PCR). CBR connections are assumed to have maximum CTD, maximum CDV, and CLR requirements.

VBR (Variable Bit Rate) Category of connections with variable bit rate, characterized by a PCR, a Sustainable Cell Rate (SRC), and a Maximum Burst Size (MBS). This category can be divided into the Real-Time VBR (rt-VBR) and Non-Real-Time VBR (nrt-VBR). The rt-VBR connections are assumed to have maximum CTD and CDV requirements.

ABR (Available Bit Rate) Category of rate-based flow-controlled connections, characterized by a PCR and a Minimum Cell Rate (MCR). ABR connections do not have any delay requirement, but expect low cell loss.

UBR (Unspecified Bit Rate) Category of non real-time connections, such as File Transfer, that do not have any quality of service requirement.

2.1.3 Performance Measures

In the evaluation of routing and admission control algorithms, the most common performance measure is the *percentage of rejected requests*, expressed also as a *blocking probability*, or *rejection ratio*. Algorithms with provable performance ratio are sometimes given in the more general context of *profit maximization*. Each request could be associated with some notion of profit, and the goal of the algorithm would be to maximize the total amount of profit due to accepted requests. The profit may be proportional to the bandwidth-duration product [5]. If the profit of each request is 1, then the profit maximization becomes equivalent to minimizing the percentage of rejected requests.

Another performance issue that has been studied in the area of routing and admission control algorithms is the one of *fairness*. Let us suppose that connection requests are divided into several classes. In [65], a network is said to be *fair* for instance "if all classes of calls have equal likelihood of being accepted". The definition of fairness depends on what we define as a class of calls. A class could be the set of all requests for the same node pair or the set of all requests for pairs at the same distance, for instance. Thus, the fairness could be measured by:

- looking at individual blocking probabilities. In [71], fairness of an algorithm is measured by the variation in blocking probabilities.
- looking at the blocking probabilities for node pairs at the same distance. For example, among origin-destination pairs at a distance of 3 hops, are there requests between pairs that are more likely to be accepted than others?
- looking at the average path length of accepted calls, as is done in [65].

2.1.4 Competitive Analysis

A wide range of problems (for example problems related to scheduling, load balancing, and admission control) are *online* problems. Computations and decisions need to be made online, without full knowledge of future events. A way of characterizing an online algorithm is to compare its performance in the worst-case with respect to the best offline algorithm. An online algorithm is *k-competitive*, or has a *competitive ratio* k, with respect to the performance measure P, if the ratio of the performance of the algorithm over the performance of the offline algorithm over all input sequences is bounded by k.

2.2 Scheduling Algorithms

This section reviews scheduling algorithms, and in particular the bounds that have been derived to support QoS applications[1]. An exhaustive survey of scheduling algorithms can be found in [2, 23]. As stated earlier, a scheduling algorithm can be work-conserving or not. We will first present three main work-conserving algorithms, FIFO, WFQ, and Virtual Clock, then provide a brief description of the non-work-conserving Stop-and-Go algorithm, and finally give an overview of other scheduling algorithms proposed in the literature.

[1]We include the discussion about scheduling algorithms for completeness. However, we will not examine the scheduling component in great detail in this thesis.

2.2.1 FIFO

The FIFO (also called First-Come-First-Served) algorithm is simple to implement and work-conserving. However it does not provide isolation between flows. A session's packets may be delayed by a burst of other sessions, and it is not possible to give a delay bound independent of other sessions. A FIFO server may give the same maximum delay guarantee to each session, if the admission control function takes into account the input rates of all sessions, as shown in [78].

A modified version of FIFO, called FIFO+, was proposed by Clark *et al.* in [13] to reduce the delay jitter. An offset field is stored in the packet headers and is updated at each hop to measure the difference of its delay with the average delay of the packet class. Using this field, the server schedules the packets in order of the arrival times increased by the packet offset. Although it decreases the delay jitter, this algorithm does not provide end-to-end delay guarantees.

2.2.2 Weighted Fair Queueing

The WFQ scheduling algorithm was proposed by Demers *et al.* [15]. In WFQ, each connection i sharing a switch output port is associated with a parameter ϕ_i and is guaranteed to receive at least a proportion $r_i = \phi_i / \sum_j \phi_j$ of the output bandwidth. The packet-based version of WFQ was analyzed by Parekh and Gallager in [59]. They show that this version, which they refer to as Packet Generalized Processor Sharing (PGPS), provides a close approximation of the fluid flow (bit by bit) model.

WFQ can be implemented by serving packets in order of increasing *timestamps*. The timestamp of the k^{th} packet of connection i, of length L_i^k and of arrival time t, is defined as [59]:

$$f_i^k = \max(f_i^{k-1}, V(t)) + \frac{L_i^k}{\phi_i} \tag{1}$$

where $V(t)$ is a virtual time updated at each arrival and packet departure event t_j in the following manner. Let b_e denote the bandwidth of the outgoing link, and B_j the set of current active connections. The virtual time at t_j is:

$$V(t_j) = V(t_j - 1) + \frac{t_j - t_{j-1}}{\sum_{j \in B_j} \phi_j} b_e \tag{2}$$

WFQ was analyzed by Parekh and Gallager [59, 60] under the condition that sources are regulated by token buckets. A bucket with (bucket capacity) parameter

β_i and (token generation rate) ρ_i upper-bounds the traffic in each interval t by $\rho_i t + \beta_i$. This is called the Linear Bounded Arrival Process (LBAP) and was initially proposed by Cruz [14]. A particular case of PGPS [59] where $\phi_i = \rho_i$ is called RPPS (Rate Proportional Processor Sharing).

They also show that, by using the leaky bucket on the input traffic, the following deterministic delay bound can be guaranteed for a connection i routed on a path P_i

$$D_i < \frac{\beta_i + (|P_i| - 1) max_k L_i^k}{r_i} + \sum_{e \in P_i} \frac{max_{i,k} L_i^k}{b_e} \qquad (3)$$

The main disadvantage of this scheduling algorithm is that it is expensive to implement since it needs to implement a priority queue, which has an $O(\log n)$ bottleneck. However, it has the advantage of providing isolation between flows, and strict end-to-end delay guarantees when used in conjunction with leaky bucket traffic shaping.

Besides the deterministic guarantee surveyed above, this scheduling discipline can also be used to provide statistical guarantees. For example, Zhang *et al.* [82] give a bound on the cell loss probability if the scheduling mechanism is used together with a traffic model based on the stochastic envelope process. It should be also noted that RPPS servers allow statistical multiplexing [11].

Several variants or improvements of PGPS have been proposed in the literature. WF^2Q, proposed in [8], is an enhancement of PGPS that reduces the discrepancy between the fluid-based and packet-based model, while being work-conserving and providing the same bounds as PGPS. However, it is as expensive to implement as PGPS. Golestani [29], as well as several other researchers (see [23] for more details), have proposed a modification of PGPS, called Self-Clocked Fair Queueing (SCFQ), or Virtual Spacing (VS). The difference with PGPS lies in the value of the virtual time $V(t_j)$. Instead of being calculated as above, it is set to the timestamp of the packet currently being serviced. Hence, their algorithm is simpler to implement than PGPS. Moreover, it retains the bounded end-to-end delay of PGPS [29]. In [32], Goyal *et al.* proposed *Start-Time Fair Queueing* (SFQ), an algorithm with the same implementation complexity as SCFQ for the case where the server provides a variable output bandwidth instead of a constant one.

2.2.3 Virtual Clock

This scheduling algorithm was proposed by Zhang in [81]. It is work-conserving, and each session receives a ***guaranteed average rate*** ρ_i. It can be implemented with a priority queue and the packets are served in increasing order of timestamps (similarly to PGPS), but the timestamp is computed here as:

$$f_i^k = f_i^{k-1} + \frac{1}{\rho_i} \tag{4}$$

Furthermore, the traffic should be shaped to respect a User-Behavior Envelope (UBE). See [81] for more details. Compared to PGPS, although a session is guaranteed an average rate independent of the other sessions, the instantaneous rate may be dependent on other sessions. For example, a session that is able to transmit a burst (at a rate higher than its guaranteed average rate), due to the inactivity of the other sessions will be penalized later if other sessions increase their rates. This "punishment" feature is mentioned in [59]. In this respect, Virtual Clock does not provide as much isolation of flows as PGPS.

Later work [79] was able to obtain a delay guarantee, even without source traffic shaping. A non-work-conserving variant of Virtual Clock, called Idling Virtual Clock, was proposed in [36], to provide end-to-end delay bounds and better fairness for best-effort traffic. Other improvements of Virtual Clock are mentioned in [73].

2.2.4 Stop-and-Go

This is a non-work-conserving scheduling algorithm proposed by Golestani in [28]. Golestani proved bounded end-to-end delays and zero packet loss. In this scheme, each session is associated with a class g (whose type has to be specified in packet headers), and its frame size T_g. When the traffic is shaped and its peak rate is limited to r_i over an interval T_g, the traffic is said to be (r_i, T_g)-smooth and the delay experienced by a session on a path $|P_i|$ is bounded by:

$$|P_i|T_g \leq D_i \leq 2|P_i|.T_g \tag{5}$$

The algorithm is easy to implement, but there is an overhead. Indeed the class type of a session needs to be included in each packet header. In terms of guarantees, the maximum delay bound is larger than with PGPS and resources are not fully utilized due to the non-work-conserving nature of the algorithm. Huang and Tsao [35] studied a variant of Stop-and-Go, *Continuous Framing*, where the ingoing and outgoing frames are synchronized. The delay bound is smaller than with Stop-and-Go and the time frames can be of any length.

2.2.5 Other Scheduling Algorithms

For the sake of completeness, we list here other scheduling algorithms proposed in the literature. Some work-conserving scheduling algorithms, called Rate Controlled Service algorithms (RCS) use traffic shapers at each network node. For example, in *Static Priority* (SP), sessions are assigned to a given set and all sessions within one set receive the same delay bound. *Earliest Deadline First* provides an individual delay bound D_i to each session i. A comparison of GPS-based and RCS-based scheduling algorithms is done in [26]. Other examples of work-conserving scheduling algorithms include: *CORR*, an extension of Weighted Round Robin proposed in [67], which provides end-to-end delay bounds by using composite traffic shapers; *Head-of-Line EDD*, described in [75]; *Delay EDD* [18]; Work Conserving Random [75]; *Deficit-Round-Robin* [70]; and *Preemptive Cut Through* [10].

Among other non-work-conserving algorithms, *Jitter-EDD*, an extension of Delay-EDD proposed by Verma *et al.* [74], provides isolation and bounds on the end-to-end delay and the delay jitter. *TCRM*, proposed in [48], provides deterministic end-to-end delay bounds when used with a leaky bucket. *Leave-in-Time* (LiT) was proposed in [19] to provide bounds on end-to-end delay, delay jitter, buffer space, and probability of distribution. It uses traffic enforcement based on the minimum rate and the maximum length of packets. Other examples of non-work conserving algorithms include *Hierarchical Round Robin* (HRR) [41], and *Smallest Response Time* (SRT) [43].

2.2.6 Comparison of scheduling algorithms

There have been numerous studies comparing different scheduling algorithms [75, 80]. In brief, the conclusions are that there are tradeoffs between strong guarantees, complexity of implementation and full utilization of resources. In general, work-conserving algorithms provide better packet end-to-end delay and better resource utilization than non-work-conserving ones. WFQ is a good compromise and provides a bound on the delay located between Stop-and-Go lower and upper bounds [2]. SCFQ would be simpler to implement, but it would result in worse bounds on the delay [31].

2.3 Routing Algorithms

This section surveys unicast routing algorithms. We first review routing algorithms for telephone networks. Then we survey algorithms for best effort traffic developed for traditional networks, and based on minimum-hop or shortest-cost path calculation according to one metric. In the third part, we review QoS routing algorithms, where the goal is to find a path satisfying one or more constraints.

2.3.1 Routing in Telephone Networks

Originally these networks were designed as *hierarchical networks*, but the technology has been moving towards *non-hierarchical networks*. Non-hierarchical networks for telephone networks are usually fully connected and are based on *alternate routing*. When a call needs to be set up, the direct link is chosen if it has enough capacity to route the call. If not, alternate routes of length two hops or more are tried following some routing scheme, until the returned path is AC-permissible, that is, it is accepted by the Admission Control algorithm. Note that in today's telephone networks, most of the traffic is carried on one or two-link paths [3]. In this section, we review algorithms for the choice of an alternate 2-hop path.

There are different variants in the ways the alternate routes are tried. Non-hierarchical routing algorithms are either *fixed* or *dynamic*. In fixed routing (FAR) [55], alternate routes are tried in a fixed pre-determined sequence. Otherwise, when the sequence of alternate routes varies with time, the routing algorithm is dynamic. A survey of dynamic routing algorithms can be found in [3, 37, 27]. For dynamic routing the sequence of alternate routes can be determined either in a *pre-planned* or *real-time* manner. For pre-planned routing algorithms (also called time-dependent routing algorithms), the alternate routes are chosen depending on the time/hour of the day, given an *a-priori* knowledge of the traffic matrix. The DNHR (Dynamic Non-Hierarchical Routing) method by AT&T is an example of a time-dependent pre-planned routing scheme. On the other hand, real-time routing algorithms find the sequence of alternate routes depending on the current state of the network. Real-time routing networks can be divided into state-dependent (which depend on the current state of the network such as link utilizations), or event-dependent routing methods [3]. State-dependent routing algorithms are either centralized, distributed, or isolated, depending on the information available to the algorithm. DCR (Dynamically Controlled Routing) of Northern Telecom is a centralized state-dependent routing strategy, which assumes a fully connected network topology. Another example of a centralized routing scheme is TSMR (Trunk Status Map Routing), an extension of DNHR. In distributed state-dependent schemes, the update of the routing can be done periodically, with a period in the order of minutes or hours, such as in WIN (Worldwide International Network), STAR (System to Test Adaptive Routing), developed by France Telecom, or DR-5, developed by Bellcore. The update can be also call by call, such as in RTNR (Real Time Network Routing), the AT&T long distance network. Event-dependent routing methods, such as DAR (Dynamic Alternative Routing) developed for the British Telecom network (by Kelly), uses trunk reservation and sticky routing. Roughly speaking, the sticky routing mechanism works as follows: when a direct link not available for routing, the same alternate path is tried until it becomes too congested (trunk reservation level reached). Then another alternate path is picked up at random. DAR assumes a fully connected network. A comparison of DAR with FAR (fixed alternate routing) can be found in [55]. Other examples of time-dependent methods include LRR (Learning with Random Routing), STT (Success-to-the-top), and STR (state -and time dependent routing).

Most of the state-dependent routing algorithms are based on a least-loaded path such as LBA (Least-Busy Alternate Routing). In [54], theoretical results are given for ALBA (Aggregate Least-Busy Alternate Routing), when link aggregate states are considered.

2.3.2 Routing in Traditional Data Networks

In traditional data networks, most routing algorithms find a shortest-cost (least-cost) path based on Dijkstra's or Bellman and Ford's algorithms. A more complete survey of routing algorithms used in traditional networks can be found in [16].

The Bellman-Ford algorithm computes shortest-cost paths from a source to all destinations. On a graph of m edges and n vertices, it runs in $O(mn)$. This algorithm is implemented in a distributed manner in the Routing Implementation Protocol (RIP).

Dijkstra's algorithm computes a shortest-cost path between a source and a destination. Implemented with a binary heap, it runs in $O((m+n)\log n)$. This algorithm is used in the Open Shortest-Path-First protocol (OSPF), which has been proposed to address the shortcomings of RIP. This implementation of Dijkstra's algorithm is a centralized link-state algorithm, where the cost is a function of the link capacity [16].

If several paths have exactly the same cost – this is likely to occur when the cost is based on the number of hops – a simple modification of the relaxation procedures of the above two algorithms could be used to select the least cost path according to a second criterion. One could also implement *arbitrary* or *random* tie-breaks to choose amongst the paths with same costs [24].

Another shortest-cost path algorithm is the Floyd-Warshall algorithm, which computes the shortest-cost paths between all node pairs. It can be used when all paths need to be computed at once, for example when the routing table is updated at regular intervals. Ford and Fulkerson [20] have proposed a distributed implementation for the shortest-path problem.

2.3.3 QoS Routing

The problem of QoS routing consists of finding a path that satisfies one or more constraints (or QoS requirements). In the general case, the constraint can be *additive*, *multiplicative* or *concave* [76]. When an arbitrary set of constraints has to be satisfied, the problem is NP-hard [22, 76]. In addition, reducing all the constraints to one constraint, i.e., expressing all the QoS requirements as a single requirement, may not be possible [76, 61]. A survey of QoS routing can be found in [50]. In this section, we study the issues of the computational complexity of the routing algorithm, the effect of the path selection on the call acceptance probability and the fairness of the network.

One simple case is where there is only one constraint, and in particular a bandwidth constraint. This model is used for telephone networks, but can also be used in an ATM network, when the QoS requirements can be reduced to a bandwidth constraint. For example, when the QoS requirement is a statistical delay bound, the effective bandwidth can be calculated given the effective bandwidth approximation of [33, 17]. When the problem is reduced to a bandwidth requirement problem, a path can be found by pruning first the graph from links that do not have enough bandwidth and then running one of the shortest-path algorithms presented in the previous section, where the cost of the path is set in order to optimize the network throughput. Matta and Shankar in [53] have studied the case where a request has a statistical delay bound, and they have shown that a cost function expressed as the sum of the link utilizations outperforms other routing functions in terms of transient behavior and request acceptance probability. However, in their case, the choice of the path is limited to minimum-hop and minimum-hop + 1 paths, and the choice of the path is made based on *probabilistic routing* (a path is assigned a weight and a path is selected probabilistically using this weight). One of the problems of a cost based on the utilizations is that it may lead to oscillations [53]. A study of greedy routing schemes for permanent virtual circuits was done by Gawlick, Kalmanek and Ramakrishnan in [24]. They have shown that a cost function exponential in the link utilizations outperforms other routing algorithms such as min-hop and max-min routing.

After having described the case of one constraint, we consider now the case of two constraints, and first the case where we have a bandwidth constraint and a delay constraint. A solution consists of pruning from the graph the links whose capacity are too small and then running a shortest-cost path algorithm with respect to the delay. Wang and Crowcroft propose another solution based on a variant of Dijkstra's algorithm [76].

Consider the problem where we are looking for the shortest cost path that has enough bandwidth, and that satisfies the delay constraints. If the delay constraints are expressed in terms of number of hops, this can be solved by pruning the graph and then using a modified version of the Bellman-Ford algorithm proposed by Guérin *et al.* for QoS routing in OSPF [34]. We give here the pseudo-code for this algorithm that computes shortest-cost paths of at most h hops for all nodes reachable from the source s. The cost of each edge (u, v) is denoted by $w(u, v)$. The algorithm uses an estimate $s(u, i)$ of the shortest cost of a path of at most i hops from the source to u.

```
1  for each node $u$
2      for $i \leftarrow 0$ to $h$
3          $s(u,i) \leftarrow INFINITY$
```

```
        $p(u,i) \leftarrow NIL$
for $i \leftarrow 0$ to $h$
    $s(u,s) \leftarrow 0$
for $i \leftarrow 1$ to $h$
    for each node $u$
        $s(u,i) \leftarrow s(u,i-1)$
        $p(u,i) \leftarrow p(u,i-1)$
    for each edge $(u,v)$
        if $s(v,i) > s(u,i) + w(u,v)$
            $s(v,i) \leftarrow s(u,i) + w(u,v)$
            $p(v,i) \leftarrow u$
```

However, if the delay constraint is not expressed in terms of number of hops, but in terms of propagation or queueing delay (depending on the scheduling algorithm used), then the problem of finding a shortest-cost path satisfying the delay constraint becomes NP-hard [22]. Furthermore, finding a path satisfying two arbitrary constraints or more is NP-hard [76]. In this case heuristics are necessary. Such strategies were proposed for delay constrained unicast routing in [69], and multicast routing in [77, 38, 68]. Rampal and Reeves [65] have studied the heuristic called Min Max Cost with Delay Bound (MMCDB) algorithm, which minimizes the maximum link cost while meeting the delay constraint. However, they showed that, when the delay constraints are not too difficult to achieve, this heuristic can be outperformed by a shortest cost path algorithm, where the cost is set as a function of the scheduler constraints.

For the case where the QoS constraints are dependent on a particular scheduling algorithm, the QoS routing problem may be solved in polynomial time. For example, Pornavalai *et al.* [64] derive a practical algorithm for the QoS routing problem when the WFQ scheduling algorithm is used, and when several QoS constraints (number of hops, delay, jitter, loss, bottleneck bandwidth) are considered.

2.4 Admission Control Criteria

In this section, we are interested in the problem of deciding if a path returned by the routing algorithm can be accepted, in order to maximize the profit of the network.

2.4.1 Phone Networks

The problem of admission control has been studied extensively in the area of telephone networks. Usually alternate routing algorithms in phone networks use a non-greedy admission control criterion to decide if a call can be accepted. This strategy, referred to as *trunk reservation*, works as follows. An alternate path is accepted only if some condition is met on the link utilizations in order to give priority to calls routed on one-hop paths. This strategy has been shown to outperform greedy strategies in theory [54] and has been implemented in many telephone networks.

2.4.2 Known Traffic Matrix on General Topologies

This section presents admission control algorithms proposed for general topology networks (such as the Internet), that require the traffic matrix to be known in advance, or to be estimated by online measurements. We will describe the algorithms by Ott and Krishnan [58] and Sibal and DeSimone [71].

State-Dependent Routing

Ott and Krishnan [58] proposed a state-dependent routing scheme based on Markov chain theory. The idea is to approximate the probability that accepting a circuit on a given link will cause a circuit to be rejected in the future due to capacity violation. A path is accepted only if the sum of the probabilities for each link of the path is smaller than one. The admission control condition of a request on a path P at time τ is:

$$\sum_{e \in P} \frac{B(b_e, \lambda_e)}{B(n_e(\tau), \lambda_e)} \leq 1 \tag{6}$$

where $n_e(\tau)$ is the current occupancy of the link (edge) e and λ_e is the offered load on e. The function $B(n, \lambda)$ represents the probability of blocking on a link of capacity n with a load of λ erlangs. This function is given by the following formula (also called the *B-erlang* formula):

$$B(n,\lambda) = \frac{\frac{\lambda^n}{n!}}{\sum_{j=0}^{n} \frac{\lambda^j}{j!}} \tag{7}$$

The offered load λ_e may be calculated from the traffic matrix (if the knowledge of traffic matrix is assumed), or may be calculated from online measurements. This latter variant of the algorithm is referred to as SDR-ADAPT.

Controlled Alternate Routing

Sibal and DeSimone [71] present a variant of alternate routing that uses state protection (trunk reservation), also called Controlled Alternate Routing. The algorithm works as follows. Each source-destination pair is associated with a primary path (of min-hop length) and a set of alternate paths. Suppose a request arrives at the network. If the primary path has enough capacity, the request is accepted on this path. However, if the primary path is full, alternate paths are tried in order of increasing length until one is found that satisfies the following admission control criterion: An alternate path P is AC-permissible if and only if:

$$\forall e \in P, b_e - n_e(\tau) \geq r_e \tag{8}$$

which means each link of the path needs to have at least r_e available bandwidth. Each link has a capacity of b_e and the parameter r_e is calculated according to the following formula:

$$\frac{B(b_e, \lambda_e)}{B(b_e - r_e, \lambda_e)} \leq 1/H \tag{9}$$

where H is the maximum allowable number of hops in any path and λ_e is the primary load on e (in Erlang).

Sibal and DeSimone studied the performance of their algorithm through simulations on a fully-connected quadrangle, and on a sparse topology of 12 nodes. It was shown to outperform the algorithm by Ott and Krishnan over all loads, and Single Path routing under moderate loads. Their simulations are done with networks where all links have the same capacity.

2.4.3 Competitive Analysis Approach

In this approach, the network topology is assumed to be general (not fully-connected as for phone networks or VP-based ATM networks). In contrast with the algorithms presented in the previous section, no knowledge of the traffic matrix is required. This approach is based on *competitive* analysis [9], used previously for online problems such as machine scheduling and page fault recovery [72]. The idea of competitive analysis is to give performance bounds on the worst-case behavior of a particular algorithm. Theoretical bounds for such algorithms are given with respect to the *competitive ratio*, which measures the profit lost by the algorithm compared with an off-line algorithm that would have knowledge of the entire input sequence. The use of the competitive ratio for the problem of routing and admission control algorithm for general topology networks was first proposed in [4]. Several models were considered, such as the throughput-maximization model and the congestion minimization model. A survey of the competitive strategy in these settings can be found in [63] and applications to high-performance computing are described in [7]. In this section, we focus on the throughput-maximization model, which corresponds to the problem of admission control for virtual circuits. A competitive non-greedy admission control algorithm was proposed for this problem by Awerbuch, Azar, and Plotkin in [5]. The algorithm accepts a circuit request only if the cost of the path is below a threshold, where the cost of the path is a function exponential in the link utilizations. In the case of Permanent Virtual Circuits (PVC), this strategy leads to a competitive ratio of $O(\log L)$, where L is the maximum length of any path that can be requested. However, the proof of competitiveness is based on the constraint that a circuit bandwidth never exceeds a fraction $O(1/\log L)$ of the link capacity. For the case where the holding times are finite (Switched Virtual Circuit Case) and known upon the request arrival, the competitive ratio is $O(\log LT)$, where T is the ratio of the maximum over the minimum holding times. In this case, the bandwidth requested should not exceed a fraction $O(1/\log LT)$ of the link capacity. For the case of unknown holding times, as shown in [21], there exists for any online algorithm, an input sequence such that the ratio of the online algorithm over the optimum is unbounded. However, competitive ratios were found if assumptions are made on the distribution of the holding times, or if limited rerouting of circuits is allowed [6].

Due to the weak guarantees on performance of the above theoretical algorithms, a practical algorithm called *EXP*, was designed in [25]. EXP does not have a provable competitive ratio, since it is a simplified version of the algorithm in [5], where the admission control condition becomes independent of the holding times. It is assumed that all virtual circuits require the same bandwidth, and in their work, all the links of the network have the same capacity. The goal of the algorithm is to optimize the acceptance probability. The algorithm works as follows. Let $u_e(\tau)$ denote the fraction of the utilized capacity of link e at time τ , that is:

$$u_e(\tau) = \frac{n_e(\tau)}{b_e} \tag{10}$$

The admission control criterion consists of accepting the request if and only if:

$$\sum_{e \in P} \mu^{u_e(\tau)} \leq \rho \tag{11}$$

This equation is somewhat similar to the admission criterion of the algorithm by Ott and Krishnan, but the cost of EXP does not depend on the online measurements of arrival rates. A method for setting the parameters μ and ρ of EXP is proposed in [25]. First they choose to set ρ equal to μ, so that a single link path that has enough bandwidth always satisfies the admission control condition. The parameter μ is expressed in function of the *critical utilization* u^*, which is defined as follows. The critical utilization is such that a two-link path with the same utilization u^* on each of its links should be rejected. The parameter μ can thus be derived from the admission control condition to be:

$$\mu = 2^{1/(1-u^*)} \tag{12}$$

In order to set u^*, they use the following heuristic. They first estimate the maximum arrival rate λ^* that is expected on a single link. The value of λ^* can be calculated from the equation below, if we assume a target maximum blocking probability of 0.02 on a single link of capacity b (all links are assumed to have the same capacity).

$$B(b, \lambda^*) = 0.02 \tag{13}$$

The equation below may be solved by the iteration method given in [39]. Once $\lambda*$ is known, they approximate the probability that accepting a call on a two-hop path with same utilization of each link will cause a future call to be rejected. This probability is similar to the calculation given by Ott and Krishnan for paths of arbitrary length, and assuming that link departure processes are independent. If each link has the same capacity b, it is given by:

$$2\frac{B(b,\lambda^*)}{B(b-u^*b,\lambda^*)} \tag{14}$$

The critical utilization is the utilization for which the above expression is smaller than 1.

The routing algorithm of EXP is the minimum-hop path among the ones satisfying the admission control criterion. However, an alternate routing algorithm, called EXP-MC was found to have nearly identical performance. The routing algorithm in EXP-MC is simply a shortest cost algorithm minimizing: $\sum_{e\in P}\mu^{u_e(\tau)}$.

A new online algorithm was proposed by Kamath *et al.* in [42]. The performance of their algorithm is given as a function of the optimum expected *rejection ratio* R^*. It achieves an expected rejection ratio of $R^*+\epsilon$, where $\epsilon = O(\sqrt{r\log n})$, and r is the maximum fraction of an edge bandwidth that can be requested by a single circuit. However, the practical performance of their algorithm was not studied. Their algorithm and analysis are based on the assumption that the holding times and circuit inter-arrival times are exponentially distributed. Observe that R^* can be estimated by a solution to a max-sum multicommodity flow problem [51, 44]. The algorithm works as follows: The routing function consists of finding the shortest path using the cost $a^{\Lambda_e(\tau)}/b_e$. The admission control criterion is more complicated than EXP. Its pseudocode is reproduced below (from their algorithm we have replaced the expressions *cost_reject* and *cost_accept* by their exact values):

```
1  if $\rho a^{\Lambda_{rej}(\tau)/b_{rej}} \le \sum_{e\in P} a^{\Lambda_e(\tau)}/b_e$
2     Reject the circuit.
3     Add the circuit to $f(rej)$.
4     Compute a random termination time.
5  else
6     Update $f$ as if the circuit is routed along the shortest_path
7     If (routing will cause capacity violation)
8        Reject the circuit.
9        Compute a random termination time.
10    Else
11       Route the circuit.
```

2.4.4 Comparison of Admission Control Algorithms

The competitive analysis approach has the advantage of providing theoretical bounds on the rejection rates. However, the guarantees that can be proved may be much weaker than the performance achievable in practice. A first performance evaluation of EXP using extensive simulations in [25] shows that EXP outperforms the algorithm of Sibal and DeSimone and a Greedy strategy based on minimum-hop routing. The performance of EXP was also studied in [1]. They show that one disadvantage with EXP is the lack of a theoretical method for selecting its parameters ρ and μ, which, if not set correctly, can significantly affect the performance of EXP depending on the network topology and offered load. The same experiments suggest that EXP is less effective than the State-Dependent Routing (SDR) algorithm by Ott and Krishnan. However, SDR has the disadvantage of requiring an advance knowledge of the traffic matrix, or an online measurement of offered loads. Moreover, the analysis of SDR is based on the assumption that circuit departures on edges are independent processes, which is obviously an approximation, more likely to be valid on highly connected networks. In fact, SDR was shown in [71] to be outperformed by the Controlled Alternate Routing (CAR) of Sibal and DeSimone on a sparse topology. Since EXP has been shown to outperform CAR on this topology [25], EXP actually outperforms SDR in this case! Although EXP and SDR have been experimentally compared, the online algorithm KPP has not been studied yet. KPP, in contrast with EXP, makes some assumptions about the distribution of call arrivals and holding times. In Chapter 3, we will show that KPP does not outperform EXP. Furthermore, the algorithms developed for the competitive strategy depend on parameters (such as μ) whose best values can be found only by empirical simulations.

Regarding the fairness of the admission control, Sibal and DeSimone noted that their alternate routing algorithm is more fair than a Single-Path routing algorithm. The trade-off between low blocking and poor fairness was observed in particular by Rampal and Reeves in [65]. They show that an algorithm that would accept only calls between adjacent nodes (which obviously is unfair) could outperform all their other routing algorithms in terms of average blocking at high loads.

In the next chapter, we discuss extensively the fairness and performance of admission control algorithms, including GREEDY, EXP, KPP, and some new variations that we propose.

Chapter 3

Experimental Results

In this thesis, we conduct an experimental study of routing and admission control algorithms. In particular, we are interested in the non-greedy algorithms (such as EXP and KPP) proposed in the context of competitive worst-case analysis for general topology networks. As stated earlier, the advantage of such algorithms is to have provable performance bounds in the worst-case, and to be independent of any advance knowledge of the traffic matrix. Simulation results are necessary however in order to know the practical performance of these algorithms. Experiments on the EXP algorithm were already conducted by several researchers. In particular, in recent experimental findings [1], the online EXP algorithm was shown to be outperformed by SDR on most of the networks considered. We note however that most of the networks simulated were highly connected ones, on which SDR is likely to perform well [71]. On a sparse topology, EXP was found to outperform SDR (from the results in [71] and [25]). Since the performance of EXP compared to SDR appears to be dependent on the network topology used, we believe further experiments need to be conducted on EXP, in order to be able to predict its behavior on a given topology.

Our goal is to study the following problems:

- So far, competitive online algorithms such as EXP have been compared in terms of average blocking probability. In this performance measure, EXP has been shown to outperform a greedy minimum-hop algorithm [25]. However, fairness can also be an important issue. It was used as a criterion of comparison for other algorithms by Rampal and Reeves [65] and Sibal and DeSimone [71], where the fairness is measured for instance by the variation in blocking probabilities. EXP uses a non-greedy approach with an inherent unfairness, since at high loads, requests for long paths are more likely to be rejected than requests for short paths. However, one can wonder if the online nature of EXP and its lack of statistical information about the input makes it too unfair. If so, is there a way to improve its fairness? Is there a way to derive from EXP an online algorithm that would be fairer than EXP while achieving the same network throughput?

- What is the practical performance of the online algorithm proposed by Kamath, Palmon and Plotkin [42]? They have proved that their algorithm has a rejection ratio of $R^*+\epsilon$, but no experimental study has been done yet. Does this algorithm outperform EXP in practice?

- What is the effect of the routing algorithm in the EXP admission control algorithm? Is the routing algorithm dependent on the admission control criterion? Is it possible to derive an algorithm that would achieve a better practical performance than EXP by combining another routing algorithm with the EXP admission control criterion?

- What is the performance of competitive algorithms such as EXP when QoS constraints are introduced into the model?

Given the goals above, we introduced the framework shown in Figure 1, as a high-level diagram of the pseudo-code of the algorithms we are going to simulate. When a request arrives, the *routing algorithm* finds a path in the network. We assume a call-by-call model in which the routing algorithm has a complete knowledge of the network topology and current link utilizations. Given the path returned by the routing algorithms, the scheduling algorithm checks if the path satisfies the QoS requirements (locally or end-to-end). If it does, the admission control criterion is finally checked. Observe that the EXP algorithm defined in [25] fits the framework, where the scheduling condition always returns a positive answer. In our work, we consider the routing and admission control criterion as different entities not necessarily independent of each other. The scheduling algorithm was introduced to study the more general settings where paths can have delay constraints.

The routing and admission control algorithms we used are described in the next section. The model considered, and the terminology used, are described below. Each virtual circuit request has two parameters:

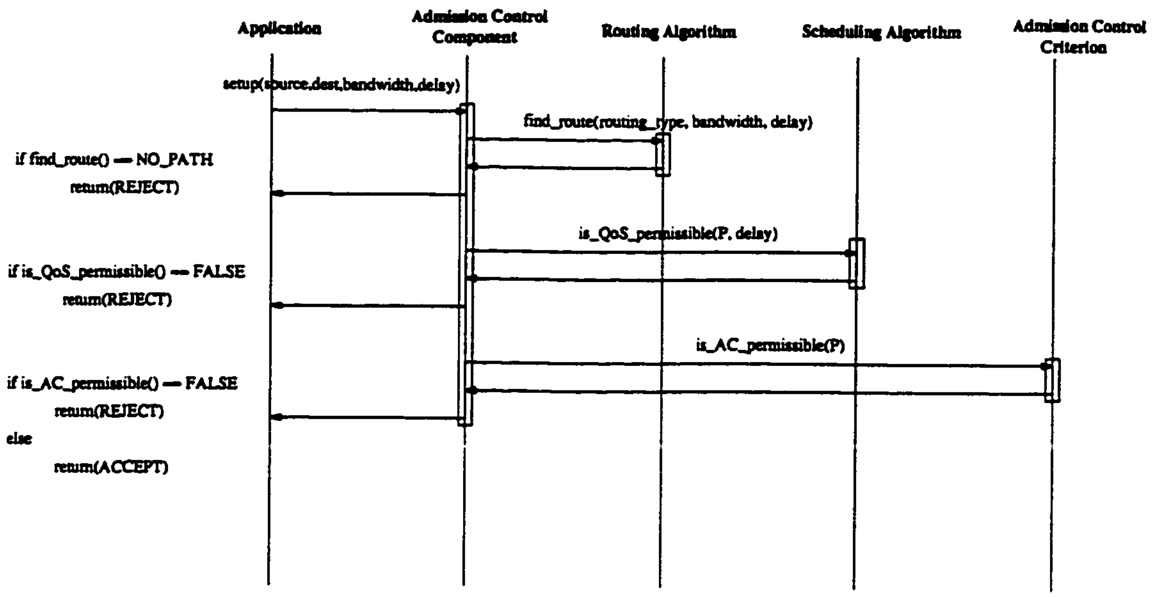

Application
Admission Control Component
Routing Algorithm
Scheduling Algorithm
Admission Control Criterion
setup(source,dest,bandwidth,delay)
find_route(routing_type, bandwidth, delay)
if find_route() == NO_PATH
return(REJECT)
is_QoS_permissible(P, delay)
if is_QoS_permissible() == FALSE
return(REJECT)
is_AC_permissible(P)
if is_AC_permissible() == FALSE
return(REJECT)
else
return(ACCEPT)

Figure 1: Admission control event trace

- a requested bandwidth r_i.
- a maximum number of hops allowed d_i.

We note that this model can be applied to CBR connections of constant bandwidth r_i, with end-to-end delay constraints. It can also be applied to VBR connections for which the bandwidth that needs to be reserved to satisfy QoS requirements is set *a-priori* to the sustainable rate r_i, as it is in RPPS servers (see Section 2.2.2). In this case, the end-to-end delay bound is still a function of the number of hops [60].

In this work, the following assumptions are made:

- The duration of each circuit is unknown to the admission control function.
- If a request can not be accepted by the network, the request is lost (no queueing of requests at the network interface).
- No re-routing of virtual circuits is allowed.

Broadly the experiments are organized as follows: In Section 3.2.1 we analyze the behavior of a greedy admission control algorithm based on minimum-hop routing, and the EXP routing and admission algorithm. These two combinations of routing function and admission control criterion were compared in [25]. The case with no delay constraint and medium size jobs is studied. We vary the reservation level and the traffic matrix. In Section 3.2.2, we evaluate alternative combinations of routing and admission control algorithms, once again with no delay constraint and medium size jobs. Also we run simulations of the KPP algorithm. In Section 3.2.3, we simulate the algorithms above with different delay constraints. Finally, in Section 3.2.4, we look at small and large jobs with 4-hop delay constraints.

3.1 Organization of Experiments

In the experiments we performed, we varied the parameters listed below. Each parameter will be described in more detail in the rest of this section.

- Choice of routing algorithm
- Choice of admission control algorithm
- Critical utilization for the EXP-type algorithm
- Choice of scheduling criteria
- Combination of routing algorithm with admission control and scheduling criterion
- Type of load
- Type of delay constraint
- Type of profit measure
- Type of network topology
- Traffic matrix

3.1.1 Routing Algorithms

Our experiments used a total of eleven routing algorithms. These can be roughly classified as favoring minimum hop paths or favoring a shortest cost path, where the cost of a link is some function of the utilization, or favoring a combination of the two. More precisely, the routing algorithms can be classified into four categories, listed below together with their implementation:

- pick up a min-cost path, using Dijkstra's algorithm.
- pick up the min-cost path among the minimum-hop paths. This was implemented by modifying the relaxation procedure of Dijkstra's algorithm.
- pick up the min-cost path among the minimum-hop and minimum-hop+1 paths. We implemented it using a simple modification of Dijkstra's algorithm we propose in Appendix B.
- pick up the min-cost path among the paths whose number of hops is no greater than a predefined threshold. We used the modified version of the Bellman-Ford algorithm reproduced in Section 2.3.3.

Each algorithm, except the *EXP-CONG* algorithm, operates by first pruning the graph from the edges that do not have enough capacity and search for a path in the remaining subgraph. The algorithms corresponding to a simple shortest-cost path calculation were implemented using Dijkstra's algorithm. All the instances of routing algorithms we considered are listed below:

1. MIN-HOP: Pick an arbitrary minimum hop path.
2. UTIL: Pick a min-cost path, where the cost of a link is $1/(1 - u_e(\tau))$ where $u_e(\tau)$ is the fraction of the bandwidth already allocated on the link at time τ. This algorithm is taken from [65].
3. EXP-UTIL: Pick a min-cost path where cost of a link is $\mu^{u_e(\tau)}$. This corresponds to the routing algorithm of the EXP algorithm proposed in [25], where μ is a parameter dependent on the reservation level.
4. UTIL-MIN-HOP: Pick a min-cost path (where the cost is defined as in (2) above) among the minimum hop paths.
5. UTIL-MIN-HOP-1 : Pick a min-cost path (where the cost is defined as in (2) above) among the minimum hop and minimum hop +1 paths.
6. EXP-UTIL-MIN-HOP : Pick a min-cost path (where the cost is defined as in (3) above) among the minimum hop paths.

7. EXP-UTIL-MIN-HOP-1: Pick a min-cost path (where the cost is defined as in (3) above) among the minimum hop and minimum hop +1 paths.

8. POWER-UTIL with power k: Pick a min-cost path where the cost of a link is $u_e(\tau)^k$

9. EXP-UTIL-UNDER-MAX-HOPS: Pick the min-cost path (where the cost is defined as in (3) above) among the ones that have a length smaller or equal than the maximum allowable number of hops d_i.

10. UTIL-UNDER-MAX-HOPS: Pick the min-cost path (where the cost is defined as in (2) above) among the ones that have a length smaller or equal than the maximum allowable number of hops d_i.

11. EXP-CONG: Pick the min-cost path, where the cost is defined as $\Lambda_e(\tau)$. This is the routing algorithm of the KPP algorithm presented in Chapter 2. This algorithm does not prune the graph from the edges that do not have enough capacity.

3.1.2 Scheduling Criteria

In our experiments, for simplicity, we express the delay constraints as a maximum number of hops allowable. If the WFQ scheduling algorithm of an RPPS server is used, the maximum number of hops H_i could be calculated given the source characterization (r_i, β_i) and the maximum end-to-end delay requirement D_i, from the equation:

$$H_i = 1 + \frac{r_i(D_i - \sum_{e \in P_i} \frac{max_{i,k} L_i^k}{b_e}) - \beta_i}{max_k L_i^k} \tag{15}$$

In the equation we assume that the VBR connections have an average rate ρ_i and have a maximum end-to-end delay requirement of D_i, and that the propagation delay is negligible, compared to the queueing delay. Given the same end-to-end delay constraint, the number of hops could vary based on the scheduling strategy.

In this framework, if the delay constraint is 2 hops, then only jobs that need to traverse 2 hops or less can be accepted. Given the end-to-end delay requirement and the bandwidth already assigned, the scheduling criterion thus becomes:

$$\text{Scheduling condition:}\quad |P| \leq H_i$$

where H_i is as defined previously.

3.1.3 Admission Control Criteria

We started with the greedy admission control criterion, and with the non-greedy criterion from [25] called *EXP*. In the first, we consider the sub-graph of the network consisting of all links that have the requisite bandwidth, and find a path according to the routing algorithm. If such a path exists, and the delay requirement would be met along this path, then the call is accepted, and the necessary bandwidth allocated along the path. In the second algorithm, we do essentially the same, except the path must also satisfy what we will call the EXP condition:

$$\text{EXP condition:}\quad \textstyle\sum_{e \in P} \mu^{u_e(\tau)} \leq \mu$$

On analysis of the EXP condition, it was found that it is designed to meet two requirements: all single hop paths will always be accepted, except if there is not enough bandwidth, and secondly, a link has a certain critical utilization above which it is reserved for single-link paths. With these principles in mind, we designed a more general class of admission control algorithms, which we call POWER. It depends on two parameters: the exponent k to which the utilization is raised, and the reservation parameter μ. The admission control condition that needs to be satisfied to accept a request is:

$$\text{POWER condition:}\quad \textstyle\sum_{e \in P} u_e(\tau)^k \leq \mu$$

The parameter μ is a function of the critical utilization. As in [25], μ is chosen such that a two-link path request should be rejected if both links have an utilization greater or equal to the critical utilization, that is:

$$\mu = 2u^{*k} \tag{16}$$

The last admission control algorithm simulated in the KPP algorithm presented in Chapter 2. The criterion to accept a path is shown below. See entire pseudocode in Chapter 2 for the calculation of $\Lambda_e(\tau)$.

$$\text{KPP condition:}\qquad \textstyle\sum_{e \in P} a^{\Lambda_e(\tau)}/b_e \leq \rho a^{\Lambda_{rej}(\tau)/b_{rej}}$$
$$\text{and (routing will not cause capacity violation)}$$

3.1.4 Combination of Routing Algorithm with Admission Control and Scheduling Criterion

In [25], the EXP admission control algorithm was tried with the routing algorithm returning the minimum-hop paths among the ones satisfying the EXP condition, and with the EXP-UTIL routing (this particular combination was called EXP-MC). We differ from the work of [25] in considering a number of routing algorithms as compatible with EXP as well as with GREEDY. In principle any combination of routing algorithm and admission control criterion can be tried. We note however that for the EXP admission control criterion, some routing algorithms may *underfunction*,that is algorithms may return a path that does not satisfy the admission control criterion when such a path exists. For example, the MIN-HOP algorithm may return a 2-hop path that will be rejected by EXP even if a less utilized path of length 3 hops would have satisfied the EXP condition. In the same way, some algorithms may underfunction with respect to the scheduling criterion, that is, they may not return a path that satisfies the delay constraint when such a path exists. For example, let us consider a request with a delay constraint of 3 hops, and let us suppose that a path of length 2 hops exists. The routing algorithm UTIL may return a path of length 4 hops if the sum of its edge costs is smaller than for the 2-hops path. Other algorithms that underfunction with respect to the delay constraint include EXP-UTIL, UTIL-MIN-HOP-1, EXP-UTIL-MIN-HOP-1. However, the other algorithms, namely MIN-HOP, UTIL-MIN-HOP, EXP-UTIL-MIN-HOP, and EXP-UTIL-UNDER-MAX-HOPS, do not under-function when delay constraints are considered.

Observe that a routing algorithm may underfunction:

- with respect of the scheduling criterion only.
- with respect to the admission control criterion only.
- with respect to both scheduling and admission control criteria.

3.1.5 Type of Load

We considered four kinds of loads, depending on the value of the bandwidth requested by virtual circuits as specified below:

1. All circuits require a *small* amount of bandwidth.

2. All circuits require a *medium* amount of bandwidth.
3. All circuits require a *large* amount of bandwidth.
4. Heterogeneous jobs: each circuit is equally likely to require a medium or small amount of bandwidth.

The actual values used for small, medium, and large jobs depend on the topology considered and are specified in Table 1 in terms of the number of circuits that can be routed on a link.

3.1.6 Type of Delay Constraint

We study three cases:

1. There is no delay constraint. (The maximum number of hops H_i is considered infinite.)
2. The delay constraint is the same for all requests. The values of delay constraint considered were 2, 3, 4 and 5 hops.
3. Each request is equally likely to have a delay constraint of 2 or 4 hops (mix of delay constraints)

3.1.7 Different Profit Measures

The profit measure generally used by researchers is the percentage of accepted calls. However, we could also consider the profit acquired by accepting a call to be proportional to the number of hops traversed by the call. Thus the profit measure would be the total number of hops traversed by accepted calls.

3.1.8 Topology and Traffic Matrix

We ran our experiments on two different topologies used as benchmarks in the literature. The first is a 12-node topology of diameter 5 based on the NSFNet T3 Backbone [71] (see Figure 2). The second is a 25-node topology of diameter 4 based on an existing commercial network [25] (see Figure 3). The traffic matrices used are given in Appendix A. As stated in Chapter 1, the traffic matrices reflect the variety of

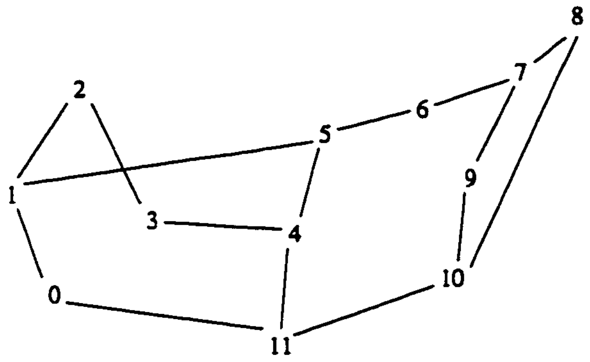


Figure 2: NSFNet topology

load patterns on the currently existing networks [71]. Each value of the traffic matrix represents the proportion of arrivals between each node pair. Each arrival process is Poisson, and the mean arrival rate is calculated as follows: If $m(i,j)$ is the value of the matrix for the pair (i,j), and A is the aggregate arrival rate desired, the mean arrival rate is equal to:

$$m(i,j) * A / \sum_{i,j} m(i,j) \tag{17}$$

In all our experiments, the offered load is shown in Erlang (mean number of expected arrivals in the mean expected holding time). Other ways of defining the x-axis that have been used in the literature are:

- Associate a matrix with the nominal load 10 (as done in [71]) and obtain other loads by linearly scaling the matrix.
- Define the load in number of connections per time unit (as done in [25]).

The default parameters used are shown in Table 1. Note that the holding times are exponentially distributed. The mean holding time on the ComNet topology is the projected mean time for a video call. On the NSFNet topology, the mean holding times and arrival rates were chosen arbitrarily to correspond to the traffic matrix given in [71]. The results of our experiments would be the same for any holding time, provided the arrival rate is adjusted accordingly. The size of medium size jobs reflects a medium picture quality video call requiring 1 Mb/s. However, all the link capacity is not available for this traffic.

Parameters	NSFNet topology	ComNet topology
Nb of jobs per link: small jobs case	400	200
Nb of jobs per link: medium jobs case	100	140
Nb of jobs per link: large jobs case	25	35
Connection type	Unidirectional	Bidirectional
Mean holding time (secs)	10	1800
Simulation time (HoldingTimes)	100	100
Length of time before gathering statistics (HT)	10	10
Link capacity	155 Mb/s	155 Mb/s

Table 1: Default parameters

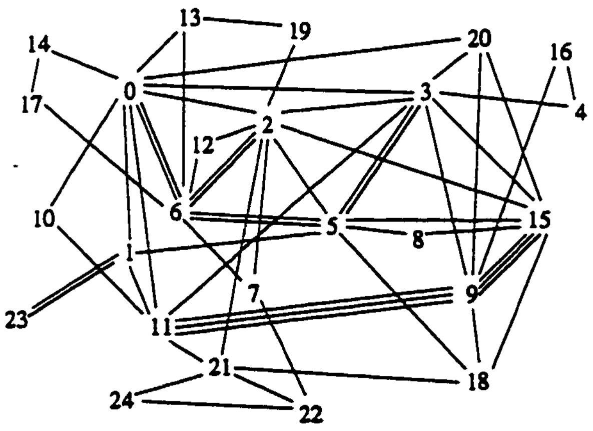


Figure 3: Existing commercial network topology

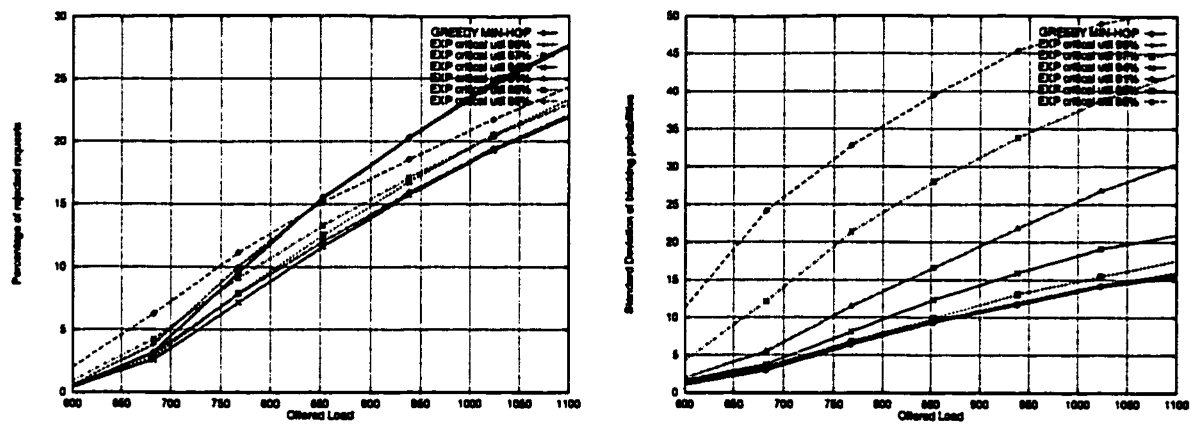


Figure 4: Varying the critical utilization on NSFNet for EXP: mean blocking probability (left) and standard deviation of blocking probabilities (right)

3.2 Results of Experiments

3.2.1 Analysis of GREEDY and EXP Strategies

In this section, we discuss the behavior of GREEDY MIN-HOP, and EXP EXP-UTIL algorithms, which were compared in [25][1]. In the rest of this section we will simply refer to these as GREEDY and EXP.

Effect of the Critical Utilization on EXP

In Figures 4 and 5, we vary the critical utilization on the two configurations. Although the EXP algorithm improves the mean blocking probability over GREEDY for a wide range of critical utilizations (as observed in [25]), the standard deviation of blocking probabilities is always worse with EXP.

Effect of Profit Measure

To aid our analysis of the difference between GREEDY and EXP, we varied the profit measures. In particular, on the NSFNet topology, rather than counting the percentage or number of accepted calls, we counted in Figure 6 the total number of hops traversed by accepted calls. The results show that even with this profit measure, EXP performs better than GREEDY. However, the increase in number of hops traveled is due to

[1]Even though it is possible to combine GREEDY and EXP with other routing algorithms, in this section we stick with GREEDY MIN-HOP and EXP EXP-UTIL as the purpose is to examine the behavior of these algorithms more closely.

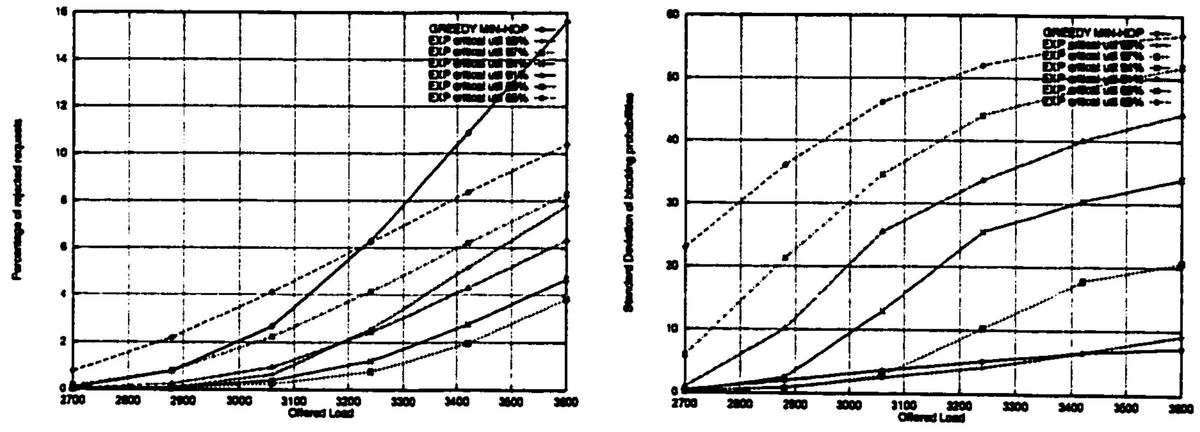


Figure 5: Varying the critical utilization on ComNet for EXP: mean blocking probability (left) and standard deviation of blocking probabilities (right)

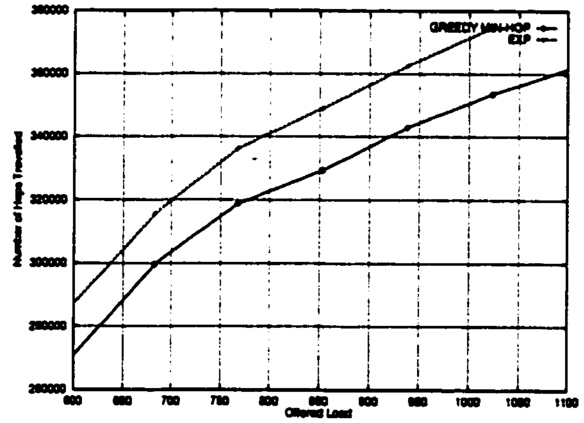


Figure 6: GREEDY vs EXP: num hops vs load

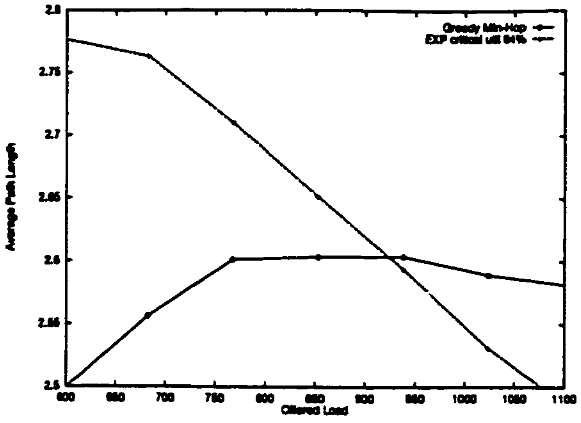


Figure 7: GREEDY vs EXP: average path length vs load

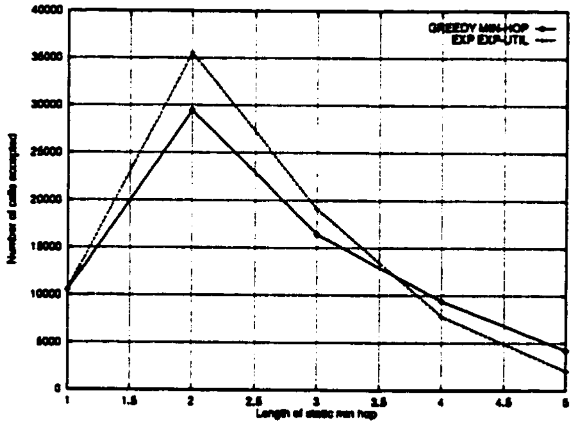


Figure 8: GREEDY vs EXP: calls vs length at load 850

the selection of longer paths at low loads, and to a lower rejection probability at high loads. Indeed Figure 7 shows that with EXP the average path length is longer than for GREEDY at low loads and larger at high loads. At high loads, EXP favors calls that don't have to travel very far, as shown in Figure 8, where the number of accepted calls is plotted with respect to the minimum distance between two nodes (we also call this number the length of the static min-hop path). The results show that GREEDY accepts more calls that have to traverse more than 2 hops. Hence although EXP outperforms GREEDY with different profit measures, EXP is more unfair than GREEDY.

Detailed Observations and Analysis of EXP Behavior

In this section, we explored further the unfairness of the EXP behavior noted above. We choose a particular simulation run on the NSFNet topology and look at other network statistics (proportion of rejected requests on a node pair basis, utilization of the links), to aid our comparison of EXP and GREEDY. We simulated the two algorithms for the case without delay constraint and for a load of 1100 Erlang on the NSFNet topology. The proportions of rejected requests on a pair by pair basis are shown in tables 2 through 6, where the pair of nodes at the same distance are grouped in the same table. The distribution of the percentage of rejected requests is plotted together in Figure 9 for the NSFNet topology and in Figure 10 for the ComNet topology. In these figures, for each integer value i on the x-axis, we plot the number of node pairs that have a percentage of rejected requests in the interval $[i, i+1]$. The mean link rates for each bidirectional link are shown in Table 7. Several observations can be drawn from these results:

- With EXP and GREEDY only three links are congested (by congested we mean that they reach their maximum utilization, and can block call arrivals): $5 \rightarrow 6$, $6 \rightarrow 7$, and $10 \rightarrow 11$. Note that these links are congested in both directions.
- The utilization dynamics of the congested links, plotted in Figures 11, 12 and 13, show that the utilization of links $5 \rightarrow 6$ and $6 \rightarrow 7$ with EXP are lower than with GREEDY, but the utilization of link $10 \rightarrow 11$ is similar for the two admission control schemes.
- EXP reduces greatly the rejection rates of calls going to or coming from node 6. Other calls have either a rejection rate slightly greater than with GREEDY (for example pair $(11, 10)$) or slightly lower (for example pair $(9, 2)$).

The fact that EXP favors calls having node 6 for origin or destination at the expense of rejecting more calls that have to travel through both links is consistent with the reservation-based nature of EXP. In this case EXP not only reserves bandwidth for single-hop paths, but also reserves bandwidth for virtual circuits that have to go through only one congested link. The lower mean utilization with EXP of the congested links $5 \rightarrow 6$ and $6 \rightarrow 7$, shows that EXP rejects circuits even if there is enough capacity. However one can wonder why link $10 \rightarrow 11$ has the same mean

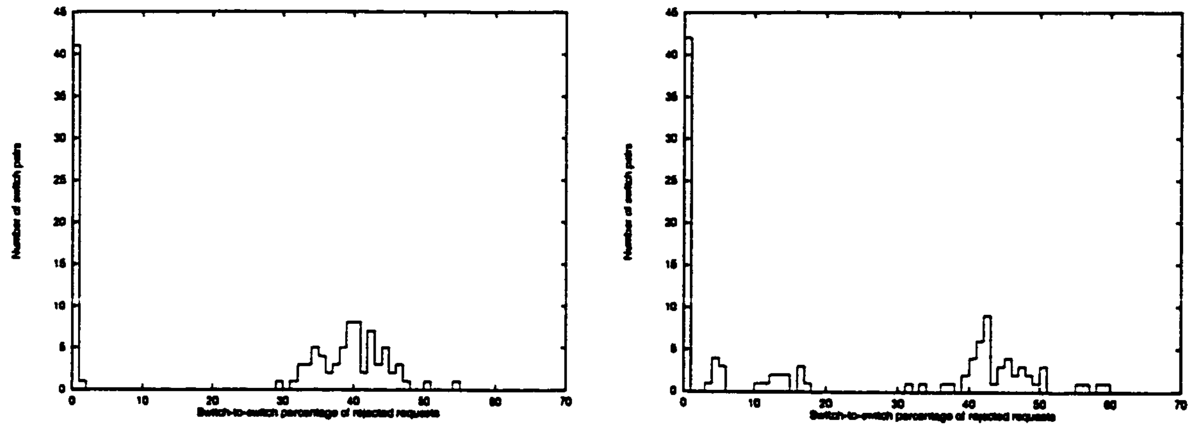


Figure 9: Distribution of the percentage of rejected requests for node pairs on the NSFNet topology for GREEDY (left) and EXP (right)

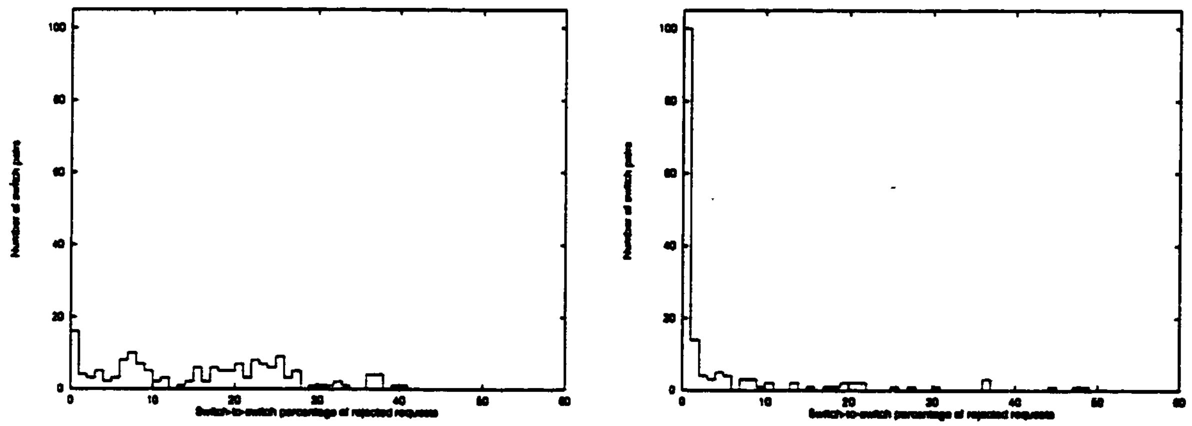


Figure 10: Distribution of the percentage of rejected requests for node pairs on the ComNet topology for GREEDY (left) and EXP (right)

utilization with GREEDY and EXP; in particular why EXP does not favor calls using only the link 10 → 11, as it favors calls using only one of the congested links adjacent to node 6. The reason for this is intuitively clear: most calls routed on link 10 → 11 are not likely to be using another congested link, and thus unlikely to be rejected by the EXP condition. Indeed, consider a path using link 10 → 11, and one of the two other congested links. Such a path would need to have a length of at least 4 hops given the topology. For example (10,6) can be routed on 10 → 11 → 4 → 5 → 6. But this path is longer than the path connecting (10,6) through 10 → 9 → 7 → 6. Since both paths are using two congested links, the shortest one will be returned by the EXP-UTIL routing algorithm. This argument could be used for other pairs as well to show that when a request has no choice but to be routed on two congested links, the path containing links 5 → 6 and 6 → 7 is is preferred to the path using link 10 → 11.

Another intriguing question is why are the utilization rates of pairs which are not coming from or going to node 6 not consistently greater with EXP than with GREEDY? For example (8,2) has a rejection rate of 61.04 (greater than with GREEDY) and (9,2) gets 39.49 (smaller than with GREEDY). By looking at the trace of the simulation run, we can see that when one of these calls is routed through links 5 → 6 and 6 → 7, EXP is more likely to reject (8,2) than (9,2) because the link 8 → 7 is more utilized than link 9 → 7. The traffic matrix confirms that there is more traffic originating from node 8 than from node 9. We note that the behavior of EXP is too aggressive in this case. Since link 8 → 7 is utilized at 87% but never congested, rejecting more (8,2) pairs to favor (9,2) pairs does not help to reduce the global percentage of rejected requests.

Node Pairs	GREEDY	EXP
(0,1)	0.00	0.00
(0,11)	0.00	0.00
(1,0)	0.00	0.00
(1,2)	0.00	0.00
(1,5)	0.00	0.00
(2,1)	0.00	0.00
(2,3)	0.00	0.00
(3,2)	0.00	0.00
(3,4)	0.00	0.00
(4,3)	0.00	0.00
(4,5)	0.00	0.00
(4,11)	0.00	0.00
(5,1)	0.00	0.00
(5,4)	0.00	0.00
(5,6)	0.00	0.00

Node Pairs	GREEDY	EXP
(6,5)	0.00	0.00
(6,7)	33.70	5.29
(7,6)	33.04	4.68
(7,8)	0.00	0.00
(7,9)	0.00	0.00
(8,7)	1.09	0.00
(8,10)	0.46	0.00
(9,7)	0.00	0.00
(9,10)	0.00	0.00
(10,8)	0.00	0.00
(10,9)	0.00	0.00
(10,11)	43.44	47.51
(11,0)	0.00	0.00
(11,4)	0.00	0.00
(11,10)	35.18	41.90

Table 2: Percentage of rejected requests for calls at 1 hop distance

Node Pairs	GREEDY	EXP
(0,2)	0.00	0.00
(0,4)	0.00	0.00
(0,10)	46.64	50.84
(1,3)	0.00	0.00
(1,4)	0.00	0.00
(1,6)	32.39	12.39
(1,11)	0.00	0.00
(2,0)	0.00	0.00
(2,4)	0.00	0.00
(3,1)	0.00	0.00
(3,11)	0.00	0.00
(4,0)	0.00	0.00
(4,1)	0.00	0.00
(4,2)	0.00	0.00
(4,6)	34.09	13.08
(4,10)	40.50	45.53
(6,1)	38.10	15.07

Node Pairs	GREEDY	EXP
(6,4)	38.31	17.00
(6,8)	34.15	5.71
(6,9)	31.73	4.90
(7,10)	0.00	0.00
(8,6)	34.07	5.33
(8,9)	0.75	0.00
(8,11)	43.83	50.20
(9,6)	36.42	3.60
(9,8)	0.00	0.00
(9,11)	43.02	42.73
(10,0)	41.78	41.31
(10,4)	42.31	40.91
(10,7)	0.00	0.00
(11,1)	0.00	0.00
(11,3)	0.00	0.00
(11,8)	40.59	45.42
(11,9)	39.60	43.85

Table 3: Percentage of rejected requests for calls at 2 hop distance

Node Pairs	GREEDY	EXP
(0,3)	0.00	0.00
(0,6)	36.56	13.56
(0,8)	40.77	46.47
(0,9)	34.82	41.00
(1,7)	42.86	47.06
(1,10)	40.26	43.12
(2,6)	33.24	11.54
(2,11)	0.00	0.00
(3,0)	0.00	0.00
(3,6)	34.14	10.89
(3,10)	38.94	45.19
(4,7)	40.92	43.90
(4,8)	39.14	47.60
(4,9)	40.22	42.40
(6,0)	38.74	17.91
(6,2)	37.62	17.70

Node Pairs	GREEDY	EXP
(6,3)	40.40	14.56
(6,10)	32.78	5.56
(6,11)	39.97	17.00
(7,1)	39.83	38.10
(7,4)	46.59	46.02
(7,11)	41.74	43.48
(8,0)	46.28	51.34
(8,4)	45.42	56.91
(9,0)	42.83	43.84
(9,4)	44.67	43.69
(10,1)	45.42	39.87
(10,3)	42.57	40.16
(10,6)	35.14	4.13
(11,2)	0.00	0.00
(11,6)	32.37	13.18
(11,7)	38.78	38.78

Table 4: Percentage of rejected requests for calls at 3 hop distance

Node Pairs	GREEDY	EXP
(0,7)	44.04	44.04
(1,8)	39.62	46.26
(1,9)	37.28	40.92
(2,7)	37.62	40.59
(2,10)	29.09	35.45
(3,7)	35.90	37.61
(3,8)	39.44	48.83
(3,9)	39.06	41.90
(7,0)	54.00	54.00
(7,2)	40.62	41.41
(7,3)	35.92	33.98
(8,1)	44.76	57.02
(8,3)	42.60	59.28
(9,1)	42.09	41.04
(9,3)	47.53	45.20
(10,2)	50.00	46.23

Table 5: Percentage of rejected requests for calls at 4 hop distance

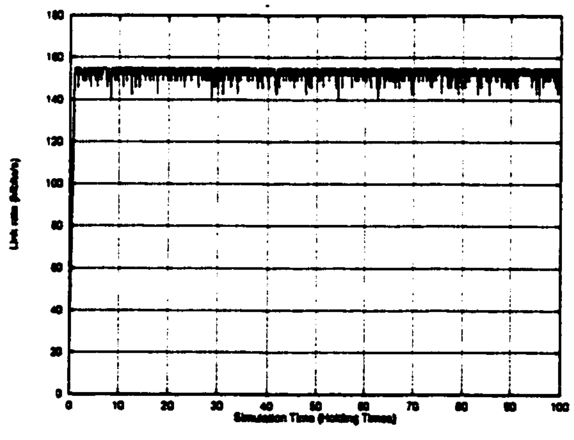
Node Pairs	GREEDY	EXP
(2,8)	39.80	48.11
(2,9)	42.78	46.46
(8,2)	44.16	61.04
(9,2)	44.57	39.49

Table 6: Percentage of rejected requests for calls at 5 hop distance

links	GREEDY	EXP
0 → 1	72.90	105.22
1 → 0	77.12	107.28
0 → 11	87.29	96.69
11 → 0	83.77	96.18
1 → 2	45.32	87.13
2 → 1	46.32	89.02
1 → 5	94.30	90.60
5 → 1	99.46	93.61
2 → 3	37.16	79.01
3 → 2	40.00	80.53
3 → 4	81.10	89.08
4 → 3	85.69	92.01
4 → 5	120.40	94.76
5 → 4	115.45	92.93
4 → 11	141.82	112.30

links	GREEDY	EXP
11 → 4	142.48	108.75
5 → 6	151.77	147.82
6 → 5	152.12	148.90
6 → 7	151.44	144.02
7 → 6	151.76	142.65
7 → 8	143.57	112.03
8 → 7	145.14	121.64
7 → 9	97.22	102.38
9 → 7	101.08	96.868
8 → 10	134.05	121.83
10 → 8	111.26	112.22
9 → 10	62.46	100.71
10 → 9	78.40	104.56
10 → 11	152.89	152.45
11 → 10	152.44	152.25

Table 7: Utilization of Links



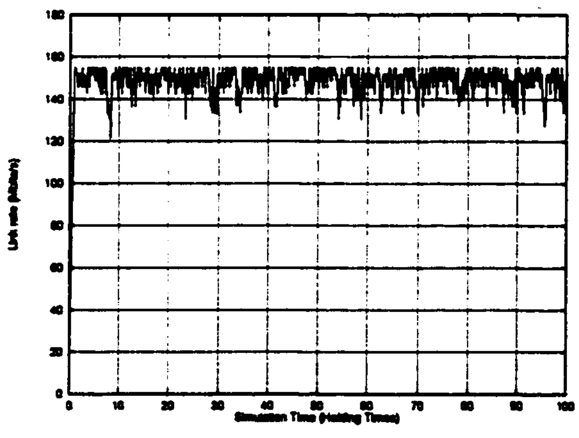


Figure 11: Utilization of bottleneck link 5 to 6 for GREEDY (left) and EXP (right)

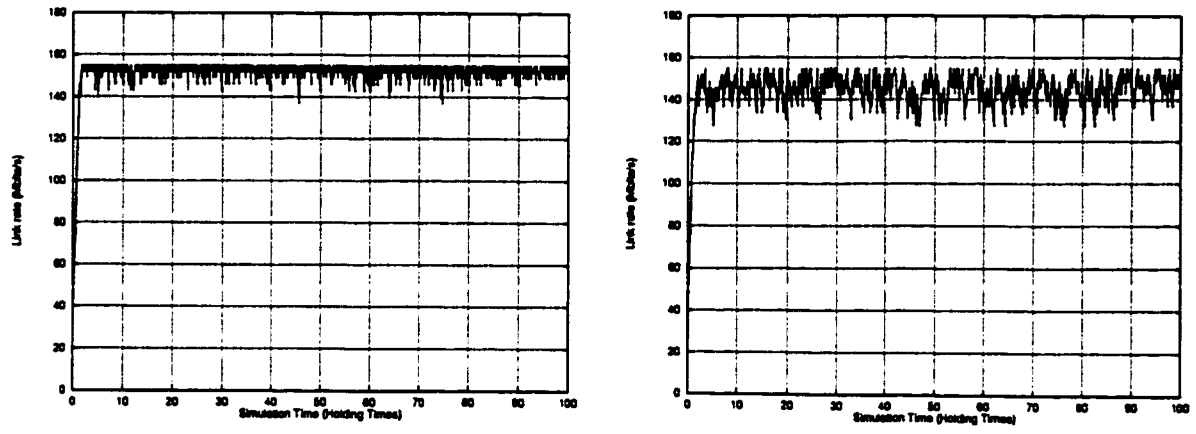


Figure 12: Utilization of bottleneck link 6 to 7 for GREEDY (left) and EXP (right)

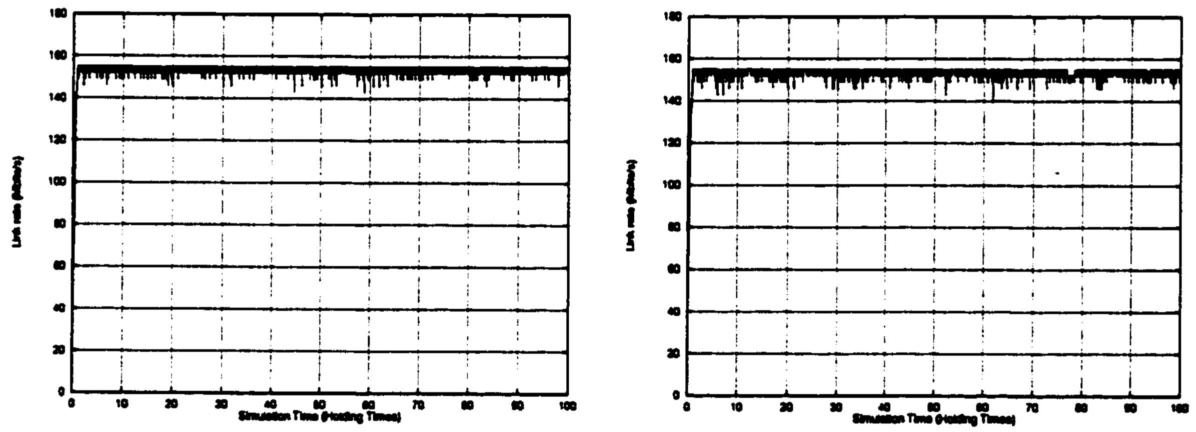


Figure 13: Utilization of bottleneck link 10 to 11 for GREEDY (left) and EXP (right)

Effect of Traffic Matrix on the NSFNet Topology

In this subsection, we explore the effect of the traffic matrix on the performance of GREEDY and EXP, using the NSFNet topology. The parameter μ for EXP was set first to the best value found in the previous section, corresponding to a critical utilization level of 94%.

First, we simulated EXP and GREEDY by using a random traffic matrix (Figure 14) or by skewing the matrix such that 80% of the traffic consists of calls at a x hops distance, where x varies between 1 and 5 (figures 15, 16, and 17). The results show that the performance advantage of EXP over GREEDY is the greatest when 80% of the traffic is 2 hops, and the performance of EXP worsens as the traffic matrix becomes less matched. When most of the traffic is 4 or 5 hops, GREEDY and EXP have nearly the same performance at high loads. It appears that at low loads with 80% of calls being 5-hop calls, GREEDY slightly outperforms EXP. However, this difference at low loads can be reduced by choosing the parameter μ differently. Indeed, Figure 18 shows that EXP achieves the same performance as GREEDY if the critical utilization is set to 99%. At high loads, GREEDY and EXP are nearly identical even if the critical utilization is set to 94%. This can be explained from Figure 18, which shows that EXP is less sensitive to the critical utilization at high loads.

Moreover, a set of experiments was run by keeping in the traffic matrix only pairs at a distance of 4 and 5 hops (see Figure 19). Here we observe that GREEDY outperforms slightly EXP not only at low loads, but over all the loads simulated, due to an incorrect setting of the critical utilization. Thus, the absence of short calls in the traffic matrix makes EXP even more sensitive to μ.

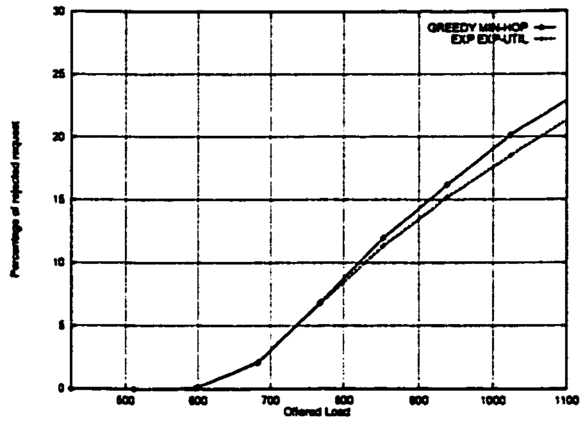
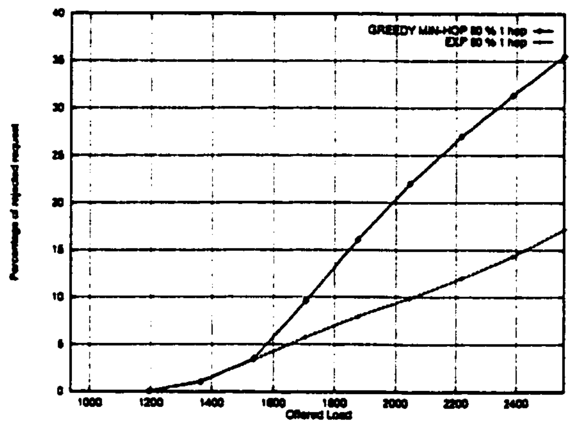


Figure 14: Effect of traffic matrix on the NSFNet topology: random matrix



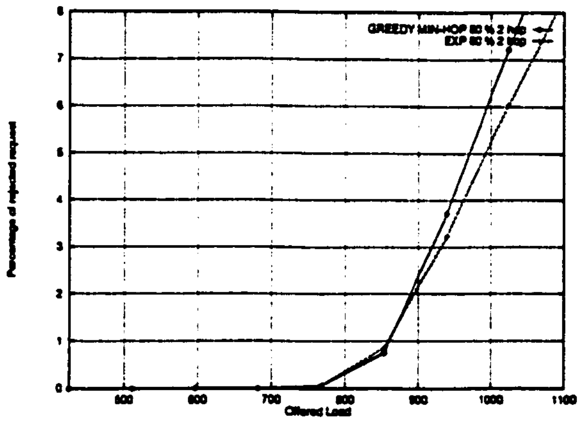
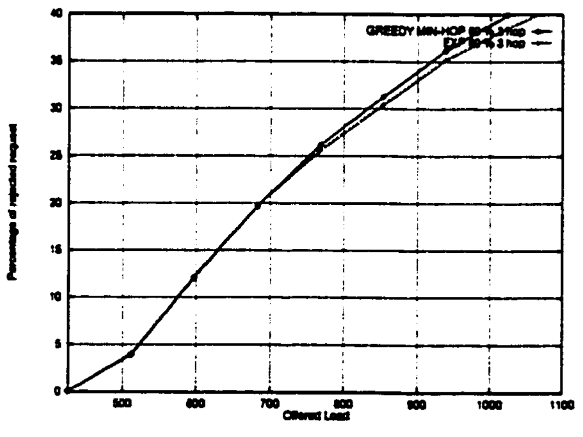


Figure 15: Effect of traffic matrix on the NSFNet topology: 80% of all calls are 1-hop calls (left) and 2-hop calls (right)



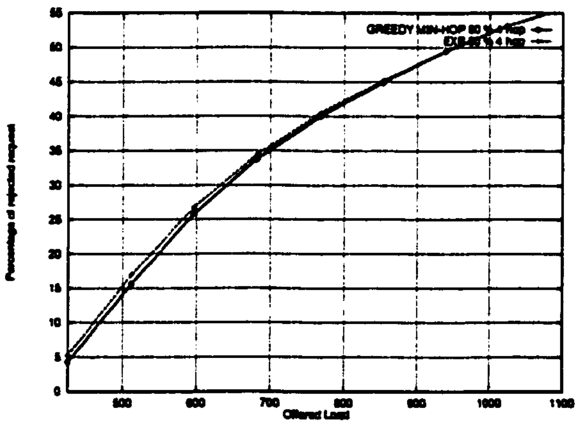


Figure 16: Effect of traffic matrix on the NSFNet topology: 80% of all calls are 3-hop calls (left) and 4-hop calls (right)

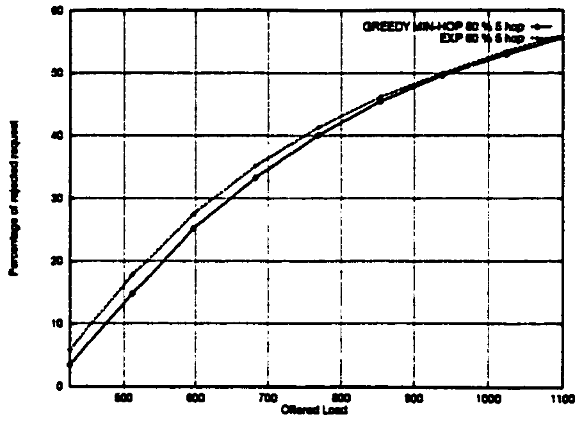


Figure 17: Effect of traffic matrix on the NSFNet topology: 80% of all calls are 5-hop calls

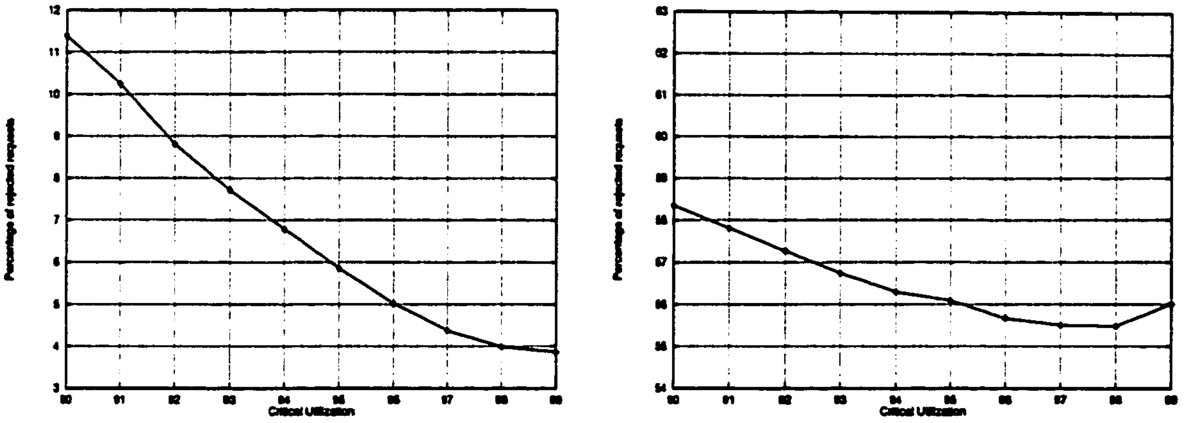


Figure 18: Effect of traffic matrix on the behavior of EXP on the NSFNet topology with 80% of all calls being 5-hop calls: varying the critical utilization at load 400 Erlangs (left) and at load 1100 Erlangs (right)

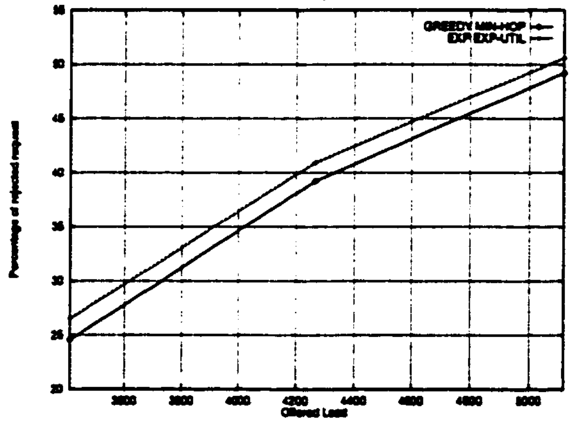


Figure 19: GREEDY vs EXP: 4_5_hops traffic

Admission control criterion	Routing algorithm	Critical utilization (in %)
GREEDY	EXP-UTIL	95.0
GREEDY	EXP-UTIL-MIN-HOP	95.0
GREEDY	EXP-UTIL-MIN-HOP-1	95.0
EXP	MIN-HOP	99.0
EXP	UTIL	97.0
EXP	EXP-UTIL	97.0
EXP	UTIL-MIN-HOP	99.0
EXP	UTIL-MIN-HOP-1	97.0
EXP	EXP-UTIL-MIN-HOP	98.0
EXP	EXP-UTIL-MIN-HOP-1	97.0

Table 8: Critical utilizations (in %) for the ComNet topology

3.2.2 Evaluation of Other Admission Control Algorithms

Varying the Routing Algorithm with EXP and GREEDY

In this section, we study the effect of the routing algorithm on the percentage of rejected requests.

The results are shown in Figure 20 for the small topology and in Figure 21 for the big topology. The simulations are run with medium size jobs and no delay constraint. For EXP EXP-UTIL, the best μ value was found in the previous section to correspond to critical utilizations of 94% and 97% on the NSFNet and ComNet topologies respectively. But these values may not remain the best ones when the routing algorithms are varied. For the algorithms depending on the parameter μ, the value of this parameter was chosen by running simulations (not included here) at the highest load considered with different values of μ , namely in $1, 2, ..., 6$, and selecting the one given the lowest blocking probability. On the NSFNet, the best values were found to be the one corresponding to a critical utilization of 94% regardless of the routing algorithm. But for the ComNet topology, different values, given in Table 8, were chosen.

From the simulation results we can make the following observations:

- GREEDY routing algorithms perform the same on the small topology. However, on the big topology, the routing algorithm has an effect and GREEDY EXP-UTIL performs the best. Further experiments (not shown here) show that with

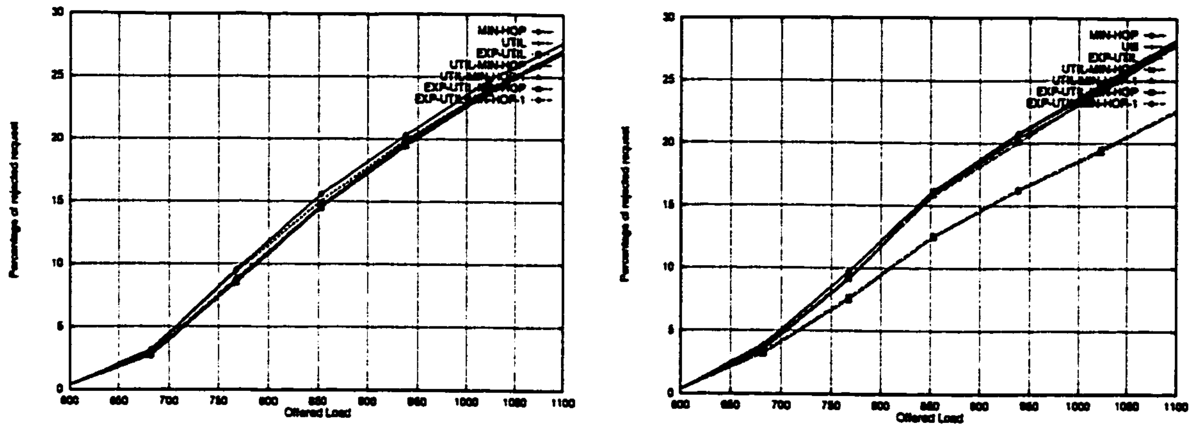


Figure 20: Effect of the routing algorithms on the NSFNet topology for GREEDY (left) and EXP (right)

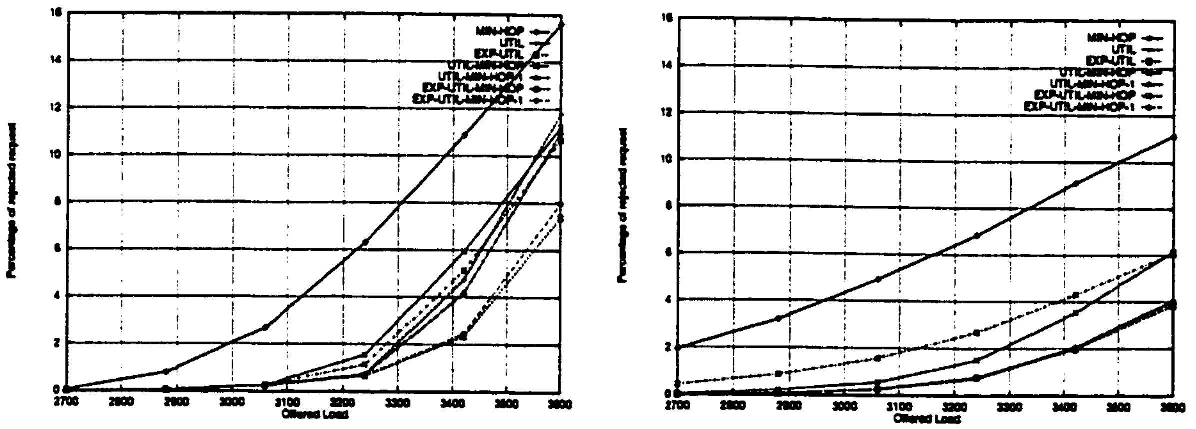


Figure 21: Effect of routing algorithms on the Commercial Network Topology for GREEDY (left) and EXP (right)

EXP-UTIL the average path length is greater than with MIN-HOP. Observe that GREEDY EXP-UTIL was shown in simulations by Gawlick *et al.* [24] to increase the amount of bandwidth routed in the case of permanent virtual circuits.

- EXP EXP-UTIL outperforms GREEDY EXP-UTIL on both topologies.
- When combined with a routing algorithm (for example MIN-HOP) that underfunctions with respect to the admission control condition, EXP is outperformed by GREEDY.

Critical Utilization	μ
90	1024
92	2212
92	5793
93	19972
94	104032
95	1.05e6
96	3.36e7
97	1.08e10
98	1.13e15
99	1.27e30

Table 9: μ as a function of the critical utilization

Varying the Admission Control Criterion

In this section, we are interested in the performance of non-greedy admission control criteria. In particular, we are looking at ways to reduce the unfairness of non-greedy approaches.

First, we run some algorithms of the class POWER(k) with different critical utilizations at a given offered load and compare it to GREEDY EXP-UTIL and EXP EXP-UTIL. Here the routing algorithm used is POWER-UTIL. The results on the NSFNet topology at load 1100 are shown on Figure 22. The results on the big topology at load 3600 are shown on Figure 23. On the NSFNet topology, POWER(k) does not outperform EXP neither in terms of mean blocking probability, nor in terms of standard deviation of blocking probabilities. A simulation of POWER(10) over all loads is shown in Figure 24. However, on the big topology, GREEDY EXP-UTIL and EXP EXP-UTIL are outperformed in terms of both measures by POWER(10) and POWER(15). In Figure 25, we compare EXP-MC and POWER(15), each taken with the critical utilization at which it performs the best: that is 97% for EXP-MC and 99% for POWER(15). The results show that POWER(15) outperforms EXP for all loads. The distribution of blocking probabilities for each node pair is shown in Figure 26.

We have observed that by using another cost function, EXP can be outperformed in terms of variance of the blocking probabilities. Now we go back to EXP and study the case where $\rho \neq \mu$. Results are shown on Figure 27 for the ComNet topology. We

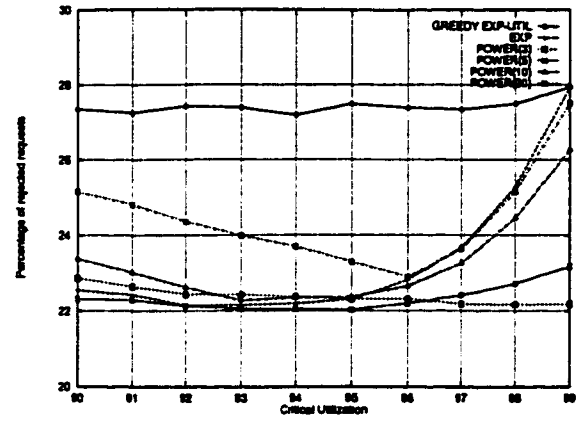
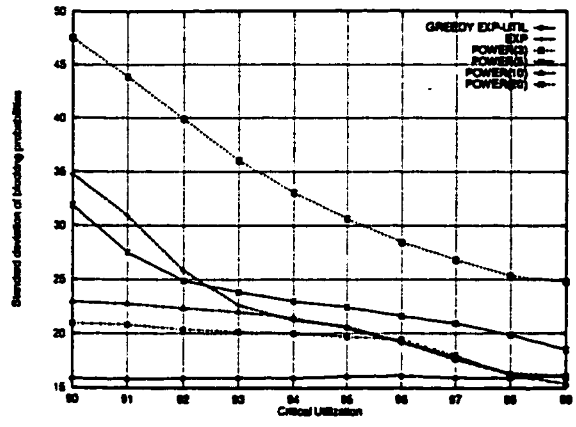



Figure 22: Varying the critical utilization for POWER(k) on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

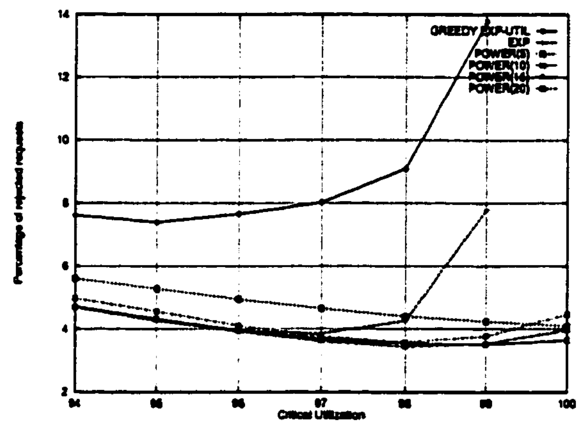
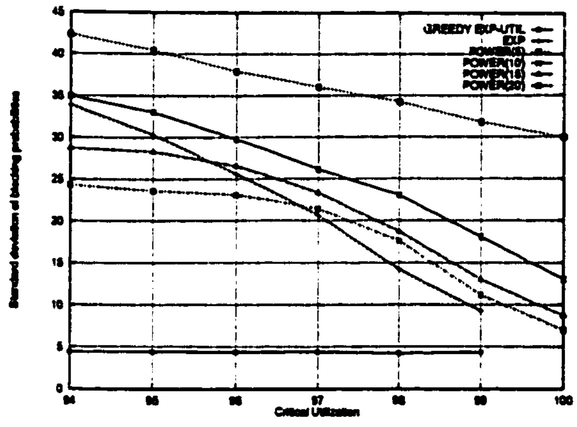



Figure 23: Varying the critical utilization for POWER(k) on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

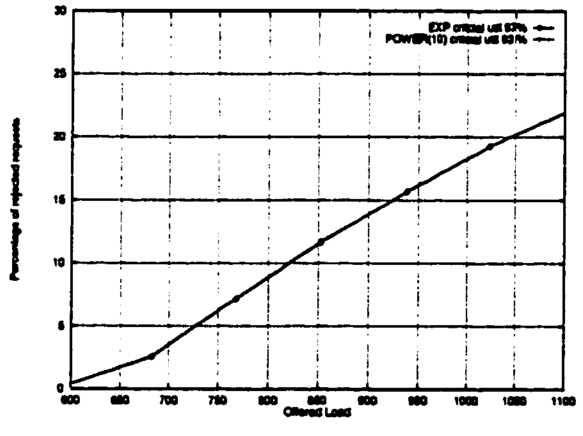
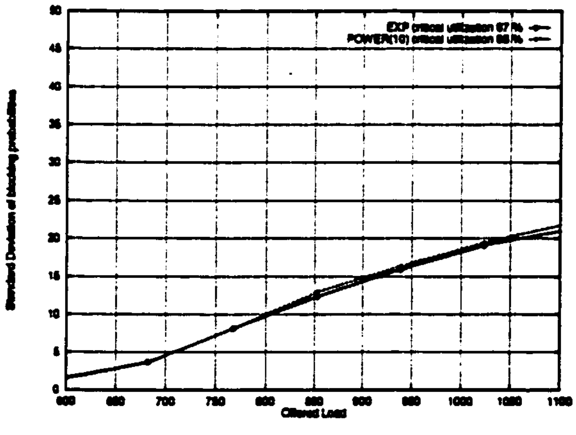



Figure 24: Comparison of POWER(10)(with a critical utilization of 93%) with EXP (critical utilization of 94%) on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

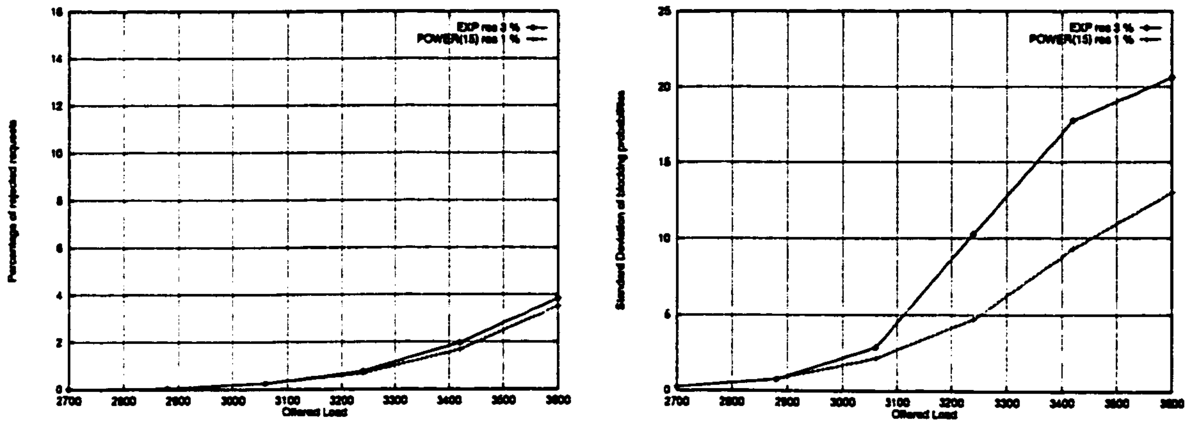


Figure 25: Comparison of POWER(15)(with a critical utilization of 99%) with EXP (critical utilization of 97%) on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

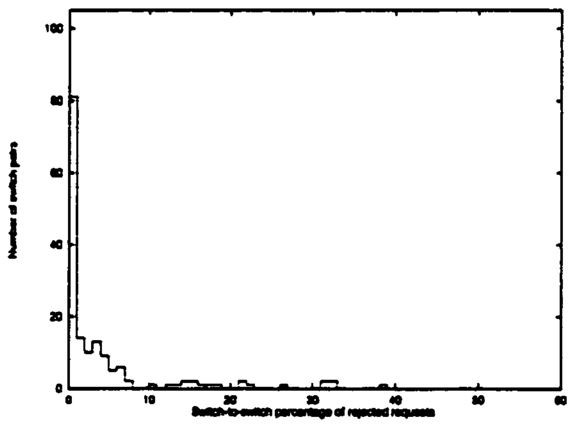


Figure 26: Distribution of the percentage of rejected requests for each switch pair for POWER(15) at load 3600 Erlang on the ComNet topology

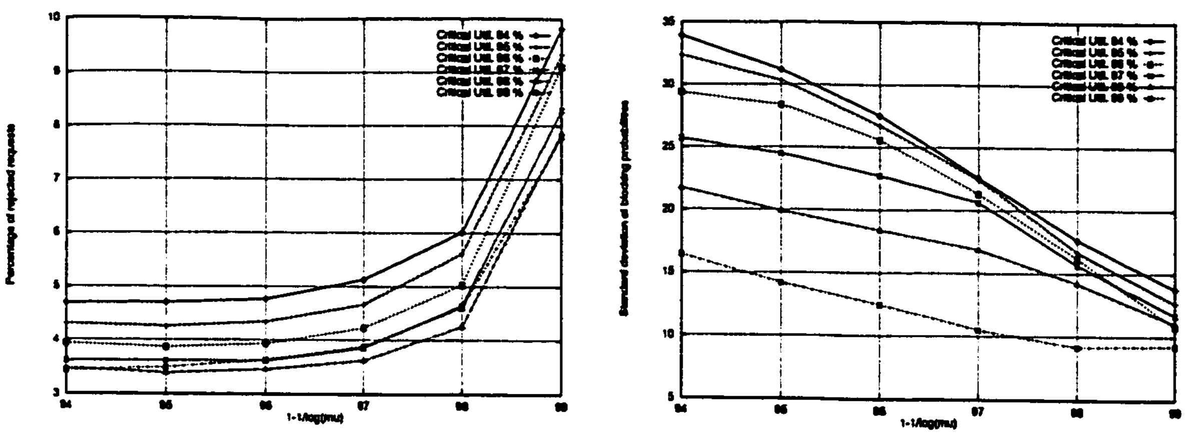


Figure 27: Varying μ for different ρ (or critical utilization) values for EXP on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

make μ take the values, shown in Table 9, that were used in the previous study. On the x-axis, instead of plotting the value of μ, we plotted $1 - 1/\log\mu$, which corresponds to the critical utilization as defined in the previous study. The parameter ρ is then calculated depending on the critical utilization and the parameter μ.

$$\rho = 2\mu^{u^*} \tag{18}$$

The results show that having μ different from ρ improves the performance of EXP. On the big topology, if we take μ such that $1 - 1/\log\mu = 97\%$, that is $\mu = 1.08e10$, and ρ such that the critical utilization is 99%, that is $\rho \simeq 1.72e10$, we note an interesting result. The standard deviation is much lower than for the regular EXP, while the percentage of rejected requests is nearly the same as before. Note that this corresponds to raising the threshold in the EXP admission control criterion. The results over all loads for these values are shown in Figure 29 and the distribution of the percentage of rejected requests at load 3600 Erlang is shown in Figure 30. The standard deviation can be also diminished on the NSFNet topology, as shown in Figure 28.

Testing the KPP Algorithm

The first problem that arises in order to simulate the KPP algorithm is how to choose the parameters a and $u(rej)$ since they depend on the parameters ϵ and R^*. We choose to take arbitrary values for a and $u(rej)$ to study the robustness of KPP.

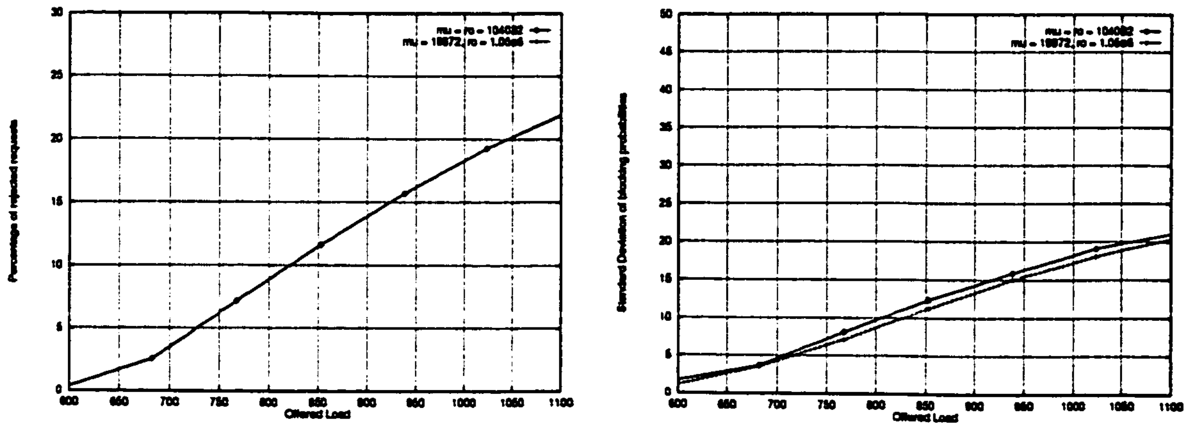


Figure 28: Comparison of EXP 94% with EXP using $(\mu, \rho) = (19972, 1.05e6)$ on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

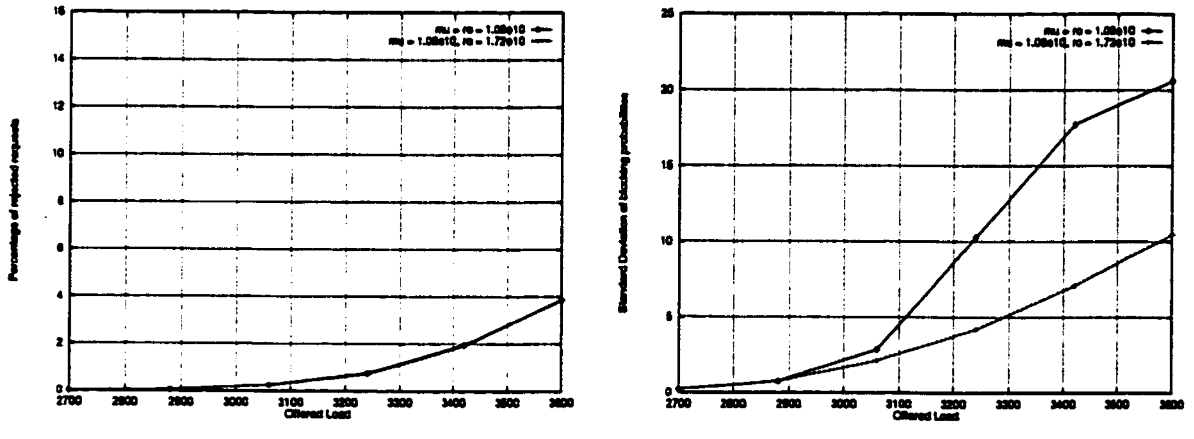


Figure 29: Comparison of EXP 97% with EXP using $(\mu, \rho) = (1.08e10, 1.72e10)$ on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

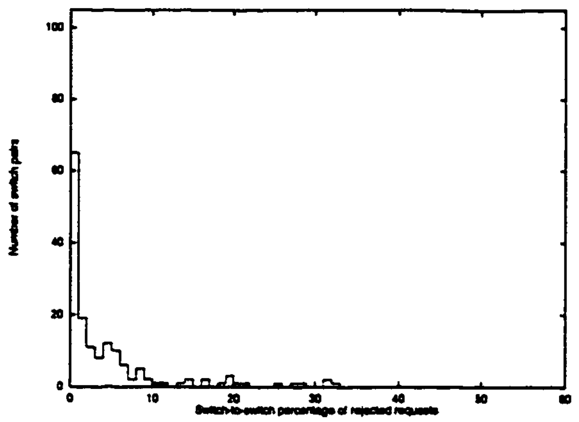


Figure 30: Distribution of the percentage of rejected requests for each switch pair for EXP using $(\mu, \rho) = (1.08e10, 1.72e10)$ at load 3600 Erlang on the ComNet topology

We first simulated KPP at low loads and observed that KPP is very sensitive to the value of $u(rej)$. Experiments show that at low loads, if $u(rej)$ is set too high, all the requests may be rejected. The results at high loads are shown on Figure 31 for the NSFNet topology and on Figure 32 for the ComNet topology. We plotted on the same figures the results for GREEDY EXP-UTIL and EXP taken from the previous section. We observe that KPP does not perform better than EXP in terms of mean blocking on both topologies (however, on the ComNet topology the mean blocking is nearly the same as for EXP). Secondly, the standard deviation of blocking probabilities for KPP is smaller than for EXP for a given mean blocking. Moreover, the deviation is the smallest when $u(rej)$ is set to 1, that is, when a path is almost never assigned to the edge rej. We observe also that KPP is less sensitive than EXP to the base of the exponential cost (called a for KPP and μ for EXP). On the NSFNet topology, when a (or μ in the case of EXP) becomes very large, the mean blocking for EXP converges towards the one of GREEDY EXP-UTIL, whereas the mean blocking of KPP stays below the one of GREEDY EXP-UTIL.

On analysis of the KPP algorithm, it is clear that a path returned by the routing algorithm can be rejected for one of two reasons:

- the condition on the cost of the path is not satisfied.
- the path would cause a capacity violation on one of its links.

Let us consider the variant of KPP where the first condition is set to be always

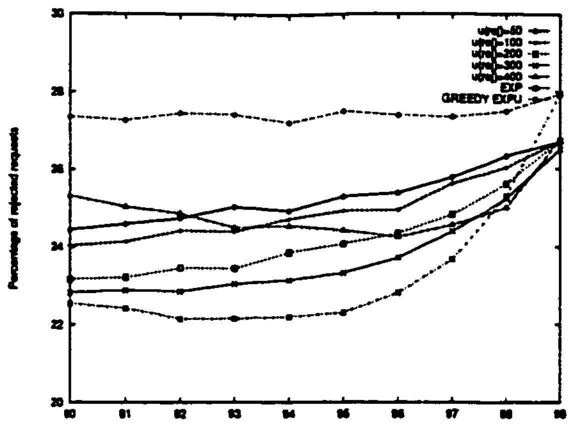
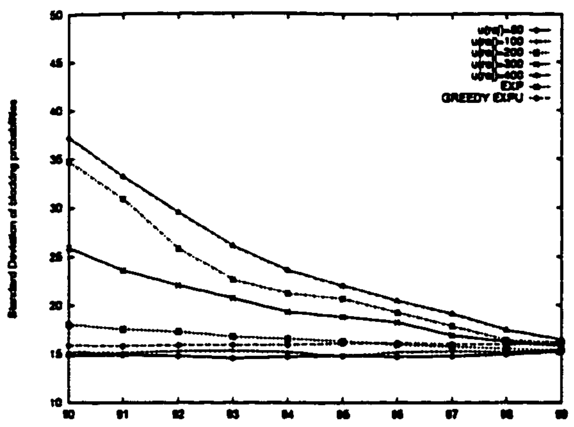



Figure 31: Varying the parameter a for the KPP algorithm on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

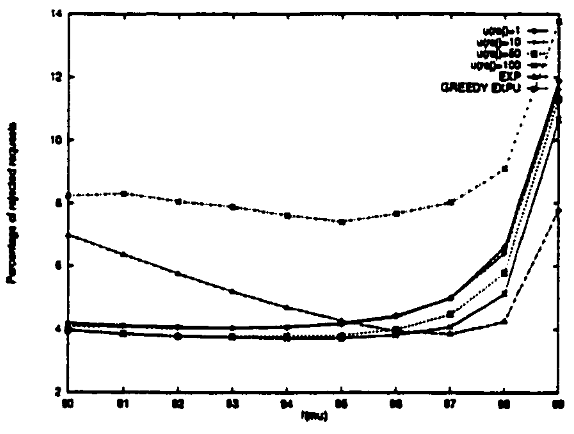
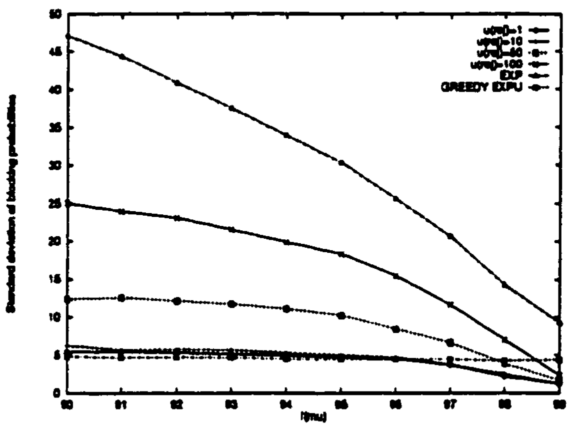



Figure 32: Varying the parameter a for the KPP algorithm on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

satisfied, and let us denote it as KPP^*. This algorithm can be seen as a Single-Path routing algorithm, picking the shortest path, where the cost is exponential in the congestion of edges. It is interesting to note that, from the experiments, this non-greedy algorithm outperforms GREEDY EXP-UTIL in terms of mean blocking, while remaining as fair. The mean blocking probability and the standard deviation of blocking probabilities over all loads are shown in Figure 33 on the NSFNet topology and . in Figure 34 for the ComNet topology. Hence KPP^* could be considered as a fair online algorithm easy to implement (since independent of the parameter $u(rej)$), holding the promise to be nearly as effective as EXP on particular topologies. However, we were not able to explain why KPP^* has a mean blocking so high compared to EXP on the NSFNet topology.

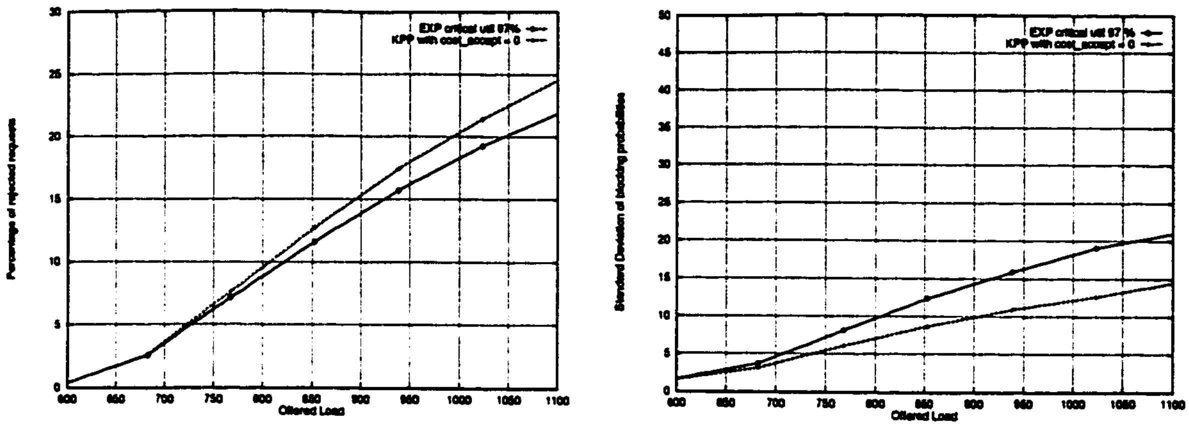


Figure 33: Comparison of EXP (94%) with KPP^* on the NSFNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

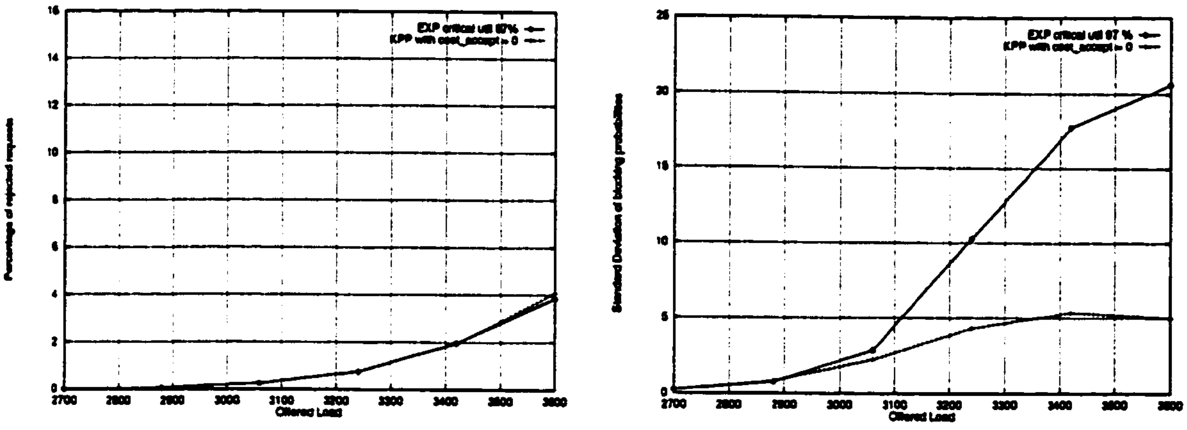


Figure 34: Comparison of EXP (97%) with KPP^* on the ComNet topology: mean blocking (left) and standard deviation of blocking probabilities (right)

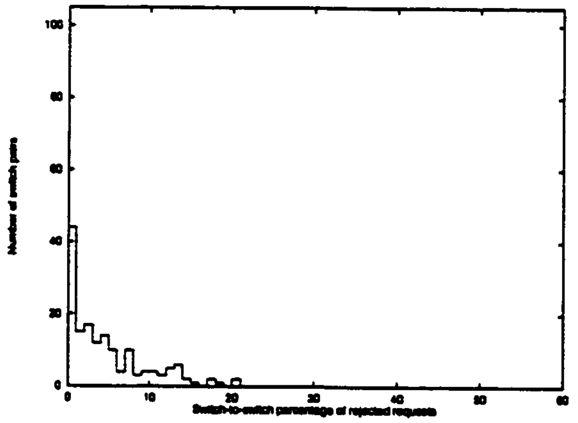


Figure 35: Distribution of the percentage of rejected requests for each switch pair for KPP^* at load 3600 Erlang on the ComNet topology

Algorithm	Mean % of rejected requests	Standard Deviation	Max
GREEDY EXP-UTIL	7.39	4.42	25.25
EXP	3.85	20.62	48.70
POWER(k)	3.53	13.01	38.15
EXP(μ, ρ)	3.89	10.48	32.95
KPP^*	4.08	5.00	20.47

Table 10: Comparison of admission control algorithms on the ComNet topology at load 3600 Erlangs

In this section, we have discussed three different ways to reduce the unfairness of EXP on the ComNet topology, that is, by using the algorithms POWER(k), EXP with $\mu \neq \rho$, or KPP. A comparison of these different algorithms is shown in Table 10. While POWER(k) achieves the lowest percentage of rejected requests, the KPP algorithm is the best in terms of standard deviation of blocking probability and maximum blocking probability.

AC criterion	Routing algorithm	2h	3h	4h	5h	Mix
GREEDY	EXP-UTIL-MIN-HOP	94.0	94.0	94.0	94.0	94.0
GREEDY	EXP-UTIL-UNDER-MAX-HOPS	94.0	94.0	94.0	94.0	94.0
EXP	MIN-HOP	99.0	97.0	94.0	94.0	96.0
EXP	UTIL-MIN-HOP	97.0	96.0	94.0	94.0	95.0
EXP	EXP-UTIL-MIN-HOP	97.0	96.0	94.0	94.0	95.0
EXP	EXP-UTIL-UNDER-MAX-HOPS	97.0	94.0	94.0	94.0	95.0
EXP	UTIL-UNDER-MAX-HOPS	97.0	96.0	94.0	94.0	94.0

Table 11: Critical utilizations for the NSFNet topology, given for each delay constraint value. "2h" refers to the case of two-hop delay constraint

3.2.3 Delay-Constrained Case

The main goal of this section is to study the robustness of EXP in the case where requests are delay-constrained. We evaluate the effect of the routing algorithms on GREEDY and EXP and the performance advantage of EXP over GREEDY in this case. We consider only the routing algorithms that pick a path among the ones satisfying the delay constraint (if such a path exists), namely MIN-HOP, UTIL-MIN-HOP, EXP-UTIL-MIN-HOP, EXP-UTIL-UNDER-MAX-HOPS, and UTIL-UNDER-MAX-HOPS. The critical utilizations that were found to work the best for each routing algorithm depending on μ are shown in Table 11 for the NSFNet topology, and in Table 12 for the ComNet topology. We observe that the value of the critical utilization is sensitive to the delay constraint, especially on the NSFNet topology. The more stringent the delay constraint, the higher the critical utilization needs to be set.

A major conclusion of this work is that delay constraints act as some sort of reservation scheme. Thus, while GREEDY performs worse than EXP without taking into account end-to-end delay, GREEDY augmented with a concern for end-to-end delay functions as a reservation-based scheme. When delay constraints are stringent, GREEDY and EXP have virtually the same acceptance rate. We note also that either for GREEDY or EXP, the routing algorithm that performs the best is always EXP-UTIL-UNDER-MAX-HOPS.

When the delay requirements are stringent (2 hops, as in Figure 36 for NSFNet or 41 for ComNet), the rejection rate is very high, but this is because the delay

AC criterion	Routing algorithm	2h	3h	4h	5h	Mix
GREEDY	EXP-UTIL-MIN-HOP	94.0	94.0	94.0	94.0	94.0
GREEDY	EXP-UTIL-UNDER-MAX-HOPS	94.0	95.0	95.0	95.0	94.0
EXP	MIN-HOP	99.5	99.5	99.0	99.0	99.5
EXP	UTIL-MIN-HOP	98.0	99.0	99.0	99.0	99.5
EXP	EXP-UTIL-MIN-HOP	98.0	99.0	98.0	98.0	99.5
EXP	EXP-UTIL-UNDER-MAX-HOPS	99.0	97.0	97.0	97.0	97.0
EXP	UTIL-UNDER-MAX-HOPS	99.0	97.0	97.0	97.0	98.0

Table 12: Critical utilizations for the ComNet topology

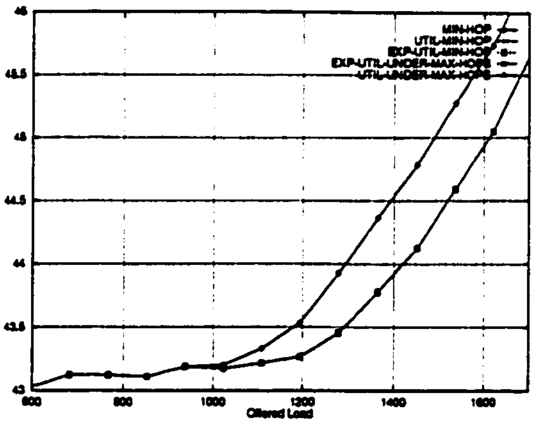


Figure 36: Effect of 2-hop delay constraints on NSFNet for GREEDY (EXP has identical performance)

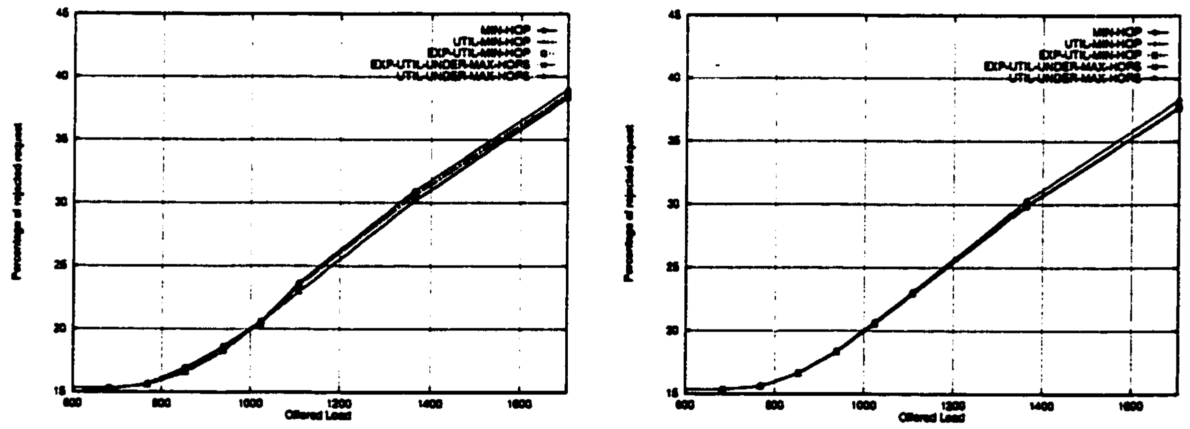


Figure 37: Effect of 3-hop delay constraints on NSFNet for GREEDY (left) and EXP (right)

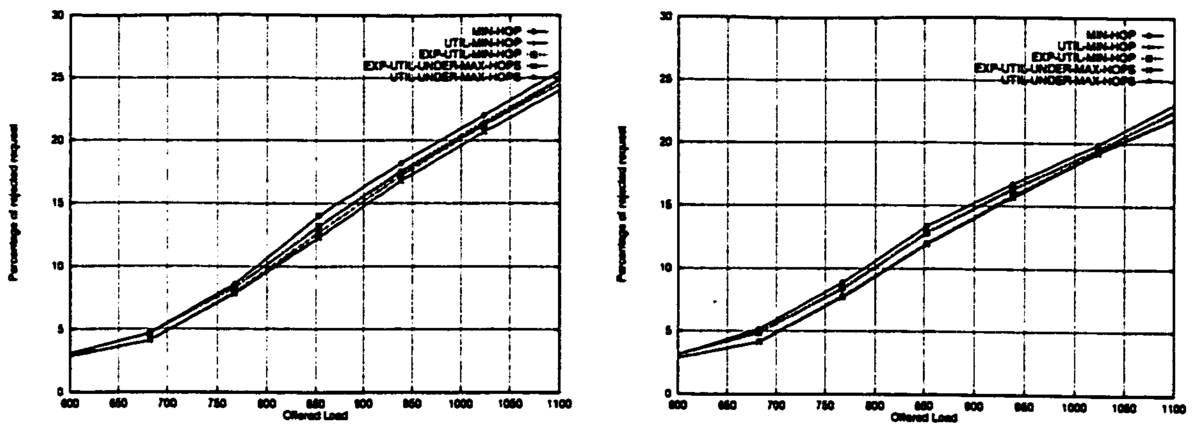


Figure 38: Effect of 4-hop delay constraints on NSFNet for GREEDY (left) and EXP (right)

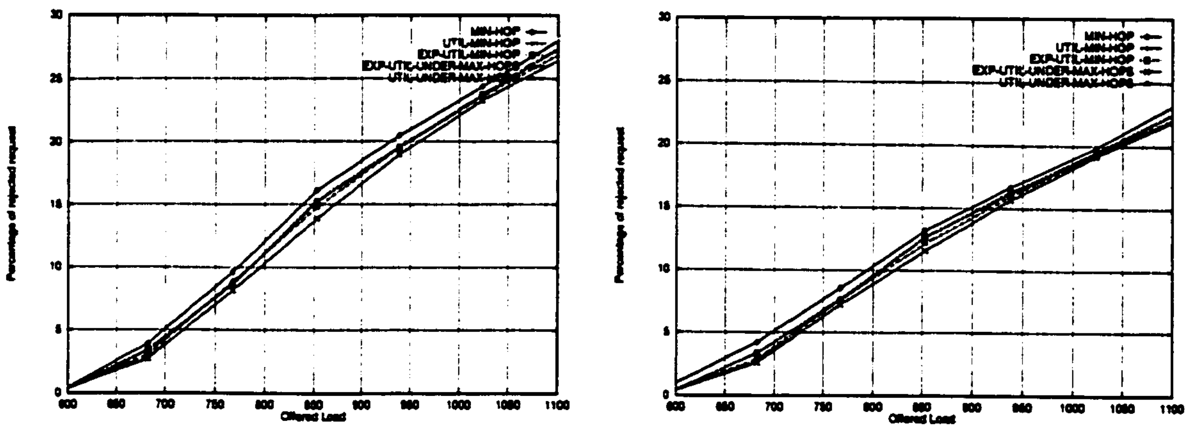


Figure 39: Effect of 5-hop delay constraints on NSFNet for GREEDY (left) and EXP (right)

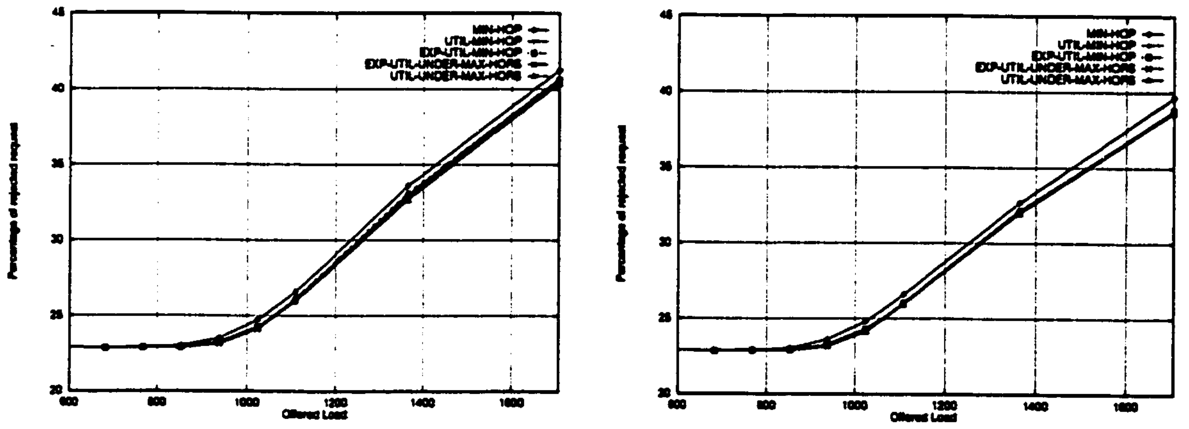


Figure 40: Effect of mix delay constraints on NSFNet for GREEDY (left) and EXP (right)

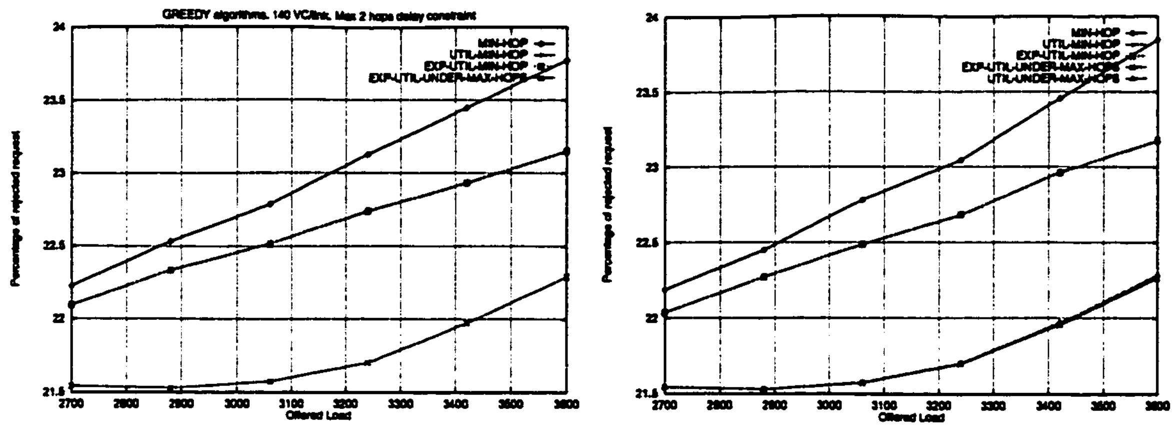


Figure 41: Effect of 2-hop delay constraints on ComNet for GREEDY (left) and EXP (right)

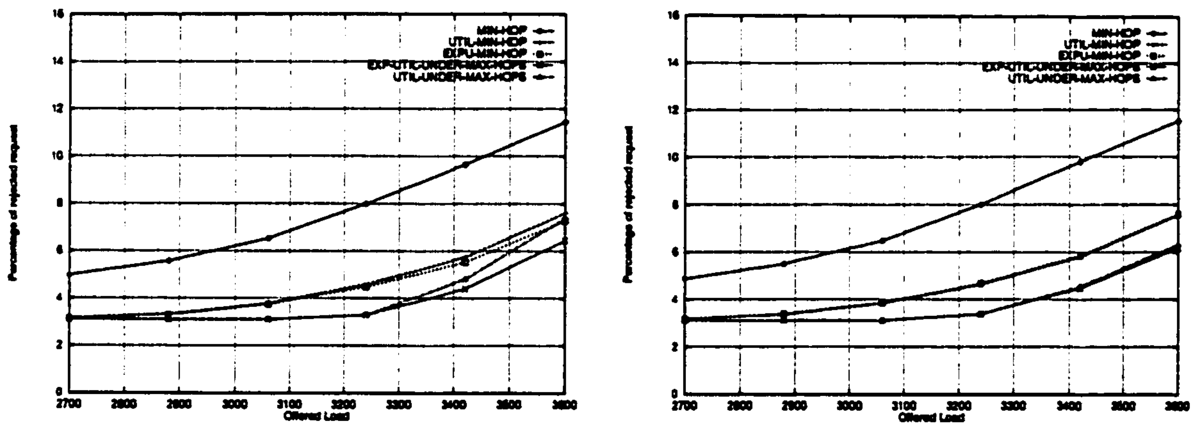


Figure 42: Effect of 3-hop delay constraints on ComNet for GREEDY (left) and EXP (right)

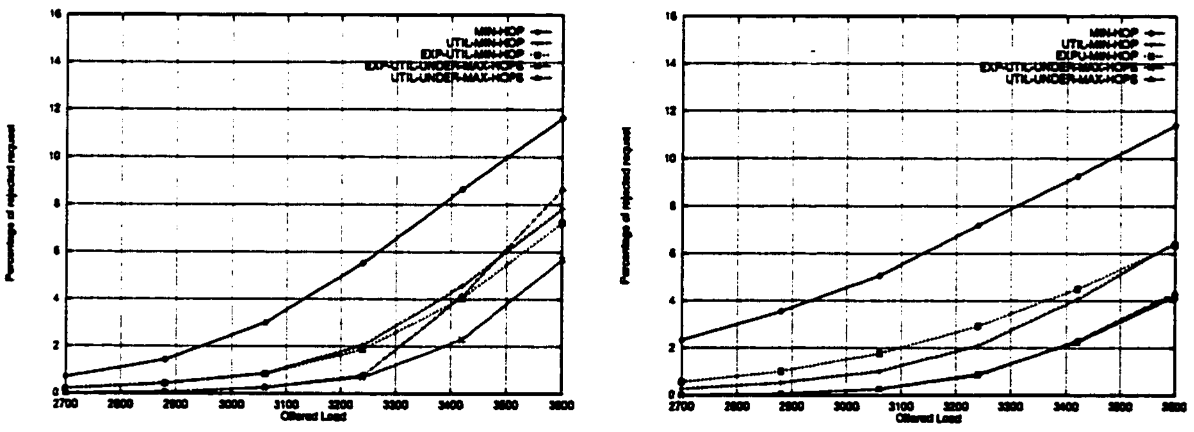
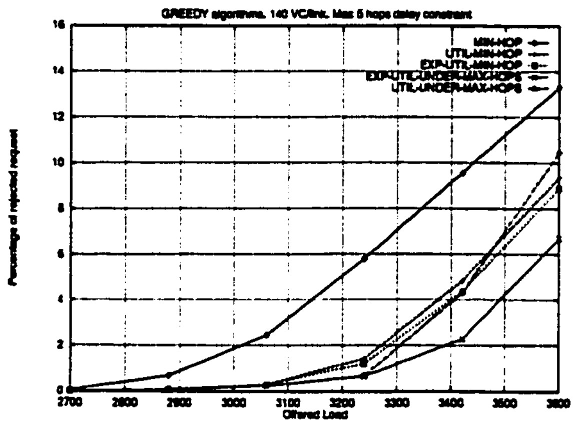


Figure 43: Effect of 4-hop delay constraints on ComNet for GREEDY (left) and EXP (right)



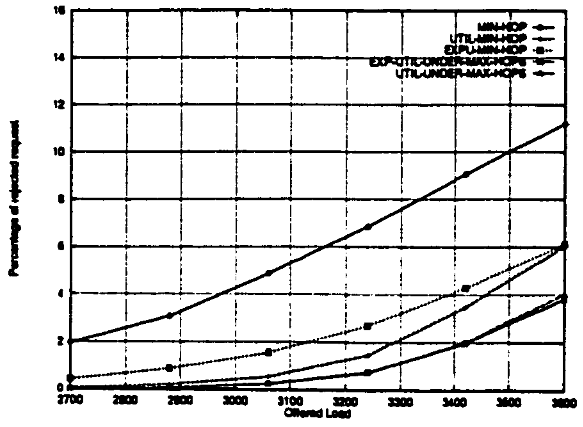
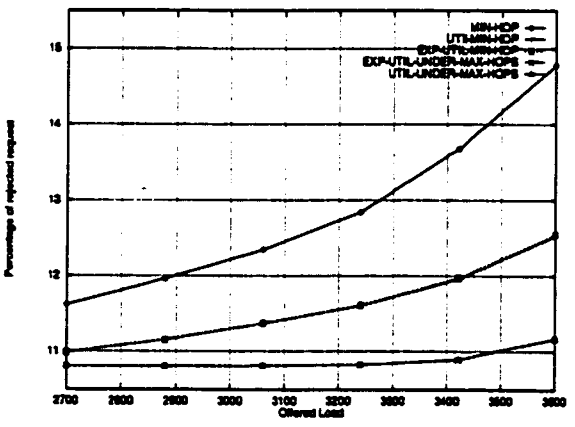


Figure 44: Effect of 5-hop delay constraints on ComNet for GREEDY (left) and EXP (right)



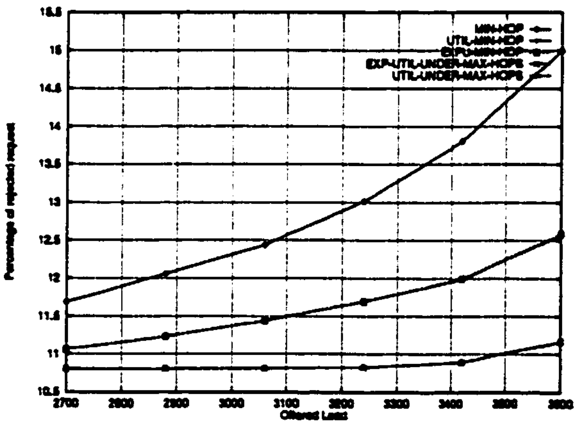


Figure 45: Effect of mix delay constraints on ComNet for GREEDY (left) and EXP (right)

requirements of most jobs cannot be met. The expected proportion of calls for which the delay constraint cannot be met can be calculated from the traffic matrix. For the NSFNet topology, these values are 43.15% for a constraint of 2 hops, 15.13% for 3 hops, 2.70% for 4 hops. The maximum distance between any two nodes is 5 for NSFNet. The values correspond to the proportion of rejected requests at low loads as shown in the results for the NSFNet topology.

For 2-hop and 3-hop delay constraints, GREEDY and EXP perform nearly identically on both topologies (See Figures 36 and 37 for the NSFNet topology and Figures 41 and 42 for the ComNet topology). However, the difference between GREEDY and EXP becomes bigger as the delay constraint increases to 4 hops (Figures 38 and 43) and 5 hops (Figures 39 and 44). We observe that, for 4-hop delay constraints, the GREEDY algorithm performs slightly better than in the case with no delay constraint, thus confirming the fact that delay constraints act as a kind of reservation scheme.

When we use a mix of delay constraints, as shown in Figures 40 and 45, the results are similar to the 2-hop delay constraint case, that is, GREEDY admission control does almost as well as EXP on both topologies when half of the requests have stringent delay requirements.

3.2.4 Effect of Job Size

In this section, we further study the robustness of EXP and compare its performance with GREEDY for different bandwidth requirements. The requests are still delay constrained, but their bandwidth requirements are either smaller than the values taken previously, larger, or heterogeneous. Both GREEDY and EXP are studied in combination with the same routing algorithms considered in Section 3.2.3.

The parameters chosen for the μ-based algorithms are shown for both topologies in Tables 13 and 14.

For big jobs, GREEDY and EXP are nearly identical if considered in the same range of rejection rates used in the medium job case (Figures 47 and 50). However, at higher loads, for example on the NSFNet topology (see Figure 46), EXP outperforms GREEDY.

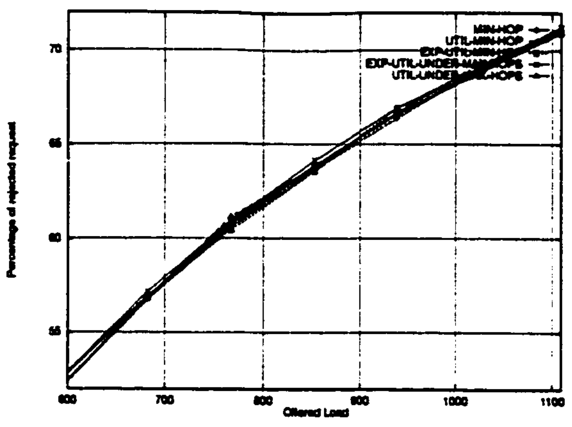
When jobs are smaller, the performance advantage of EXP over GREEDY increases on the ComNet topology (Figure 51). We would expect the conclusion to be

AC criterion	Routing algorithm	Het	Big	Small
GREEDY	EXP-UTIL-MIN-HOP	94.0	94.0	94.0
GREEDY	EXP-UTIL-UNDER-MAX-HOPS	94.0	94.0	94.0
EXP	MIN-HOP	99.0	92.0	98.0
EXP	UTIL-MIN-HOP	99.0	91.0	98.0
EXP	EXP-UTIL-MIN-HOP	98.0	91.0	97.0
EXP	EXP-UTIL-UNDER-MAX-HOPS	97.0	91.0	98.0
EXP	UTIL-UNDER-MAX-HOPS	99.0	91.0	98.0

Table 13: Critical utilizations (in %) for small, large, and heterogeneous jobs on the NSFNet topology

AC criterion	Routing algorithm	Big	Small
GREEDY	EXP-UTIL-MIN-HOP	94	94.0
GREEDY	EXP-UTIL-UNDER-MAX-HOPS	92	96.0
EXP	MIN-HOP	98	99.5
EXP	UTIL-MIN-HOP	96	99.0
EXP	EXP-UTIL-MIN-HOP	97	98.0
EXP	EXP-UTIL-UNDER-MAX-HOPS	94	97.0
EXP	UTIL-UNDER-MAX-HOPS	94	97.0

Table 14: Critical utilizations (in %) for small and large jobs on the ComNet topology



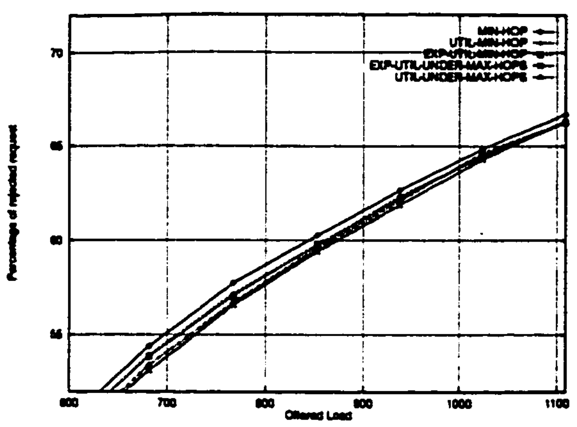
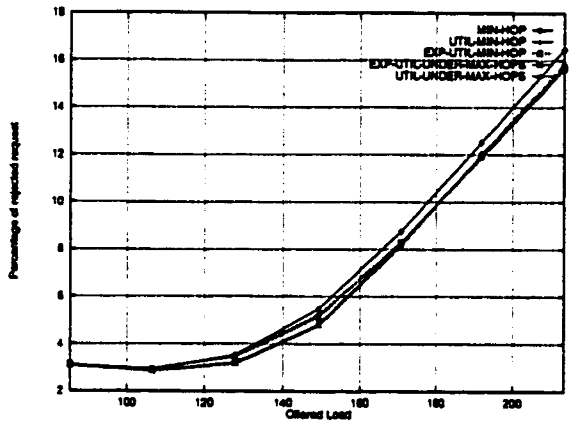


Figure 46: Effect of big jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right)



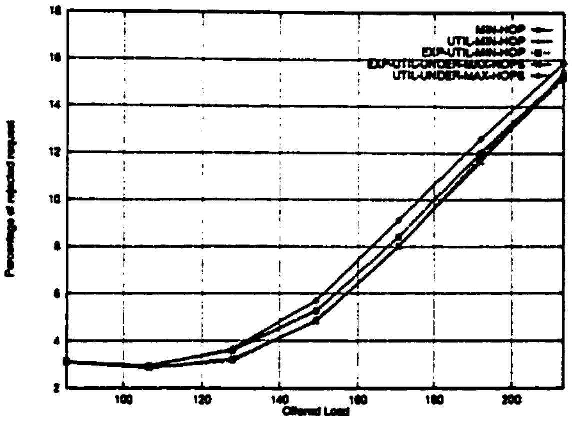
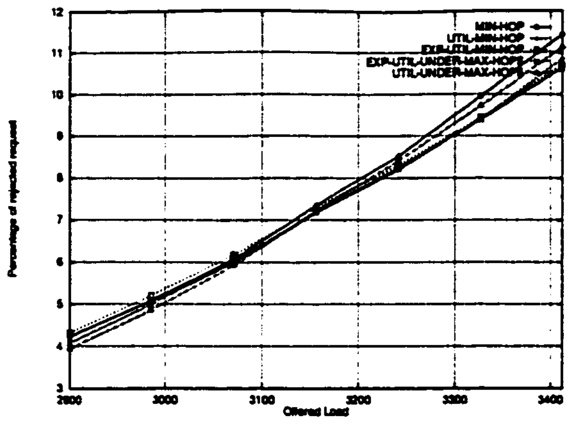


Figure 47: Effect of big jobs on NSFNet at low loads with 4-hop delay constraint for GREEDY (left) and EXP (right)

the same on the NSFNet topology. However, GREEDY and EXP are found to have identical performance in this case. We were not able to explain this behavior. Moreover, for the case of heterogeneous jobs on the same topology (Figure 49), GREEDY and EXP have the same performance.



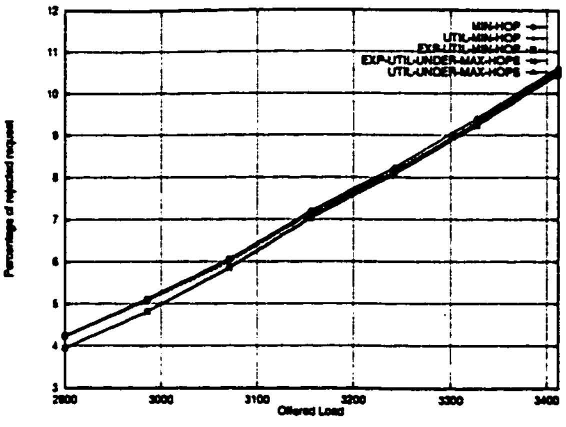
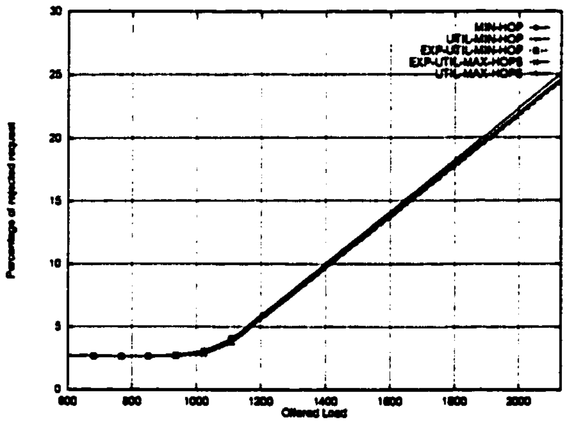


Figure 48: Effect of small jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right)



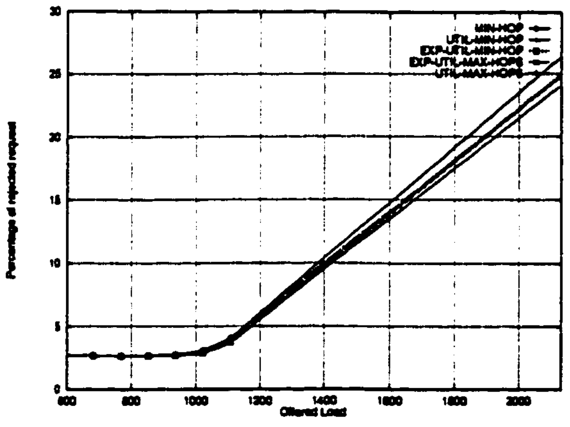
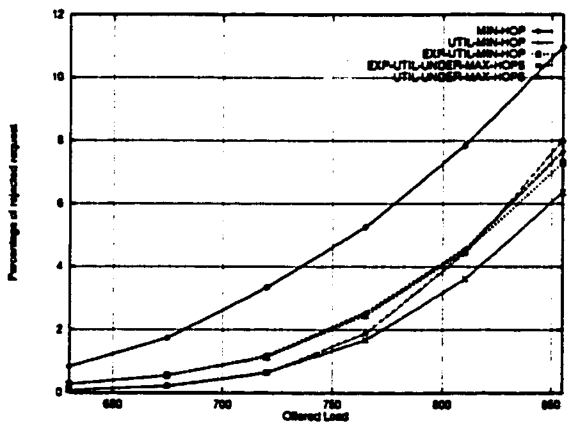


Figure 49: Effect of heterogeneous jobs on NSFNet with 4-hop delay constraint for GREEDY (left) and EXP (right)



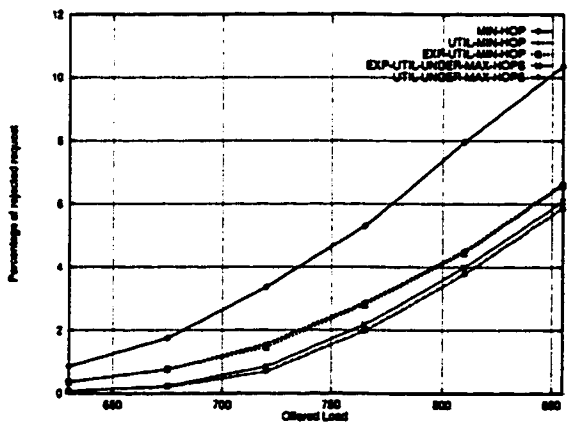
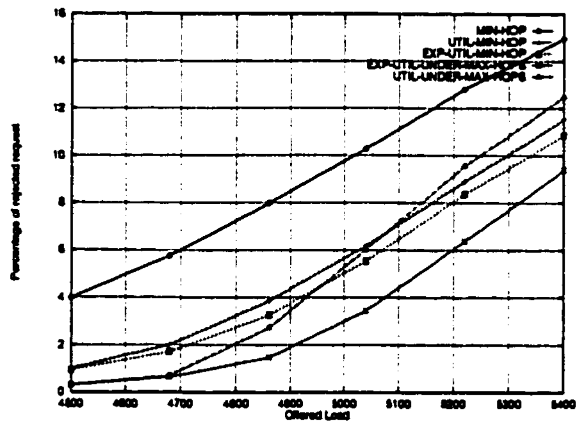


Figure 50: Effect of big jobs on ComNet with 4-hop delay constraint for GREEDY (left) and EXP (right)



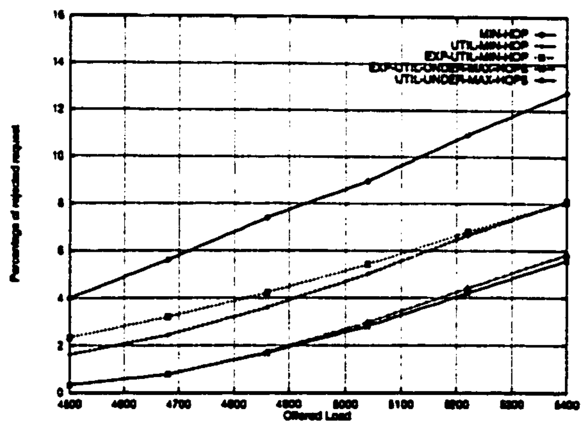


Figure 51: Effect of small jobs on ComNet with 4-hop delay constraint for GREEDY (left) and EXP (right)

Chapter 4

Simulator Implementation

The simulation results presented in the previous chapter were done using an upgraded version of the NIST ATM Network Simulator. This chapter gives an overview of the changes made to the simulator in terms of functionalities and architecture.

4.1 The NIST ATM Network Simulator

4.1.1 Functionalities

The NIST ATM Network simulator [30] is a software package available in the public domain and used primarily for the simulation of ABR flow control mechanisms and performance evaluation of medium access control for Cable-TV (HFC) networks. A survey of the functionalities of the simulator can be found in [30]. In itself, the simulator is not adequate for the simulation of routing and admission control algorithms. Indeed, in the simulator, the routing is done in a static manner. The routes need to be created by the user before the beginning of the simulation and no connection can be added once the simulation has begun.

4.1.2 Architecture

The architecture diagram of the simulator is presented in figure 52.

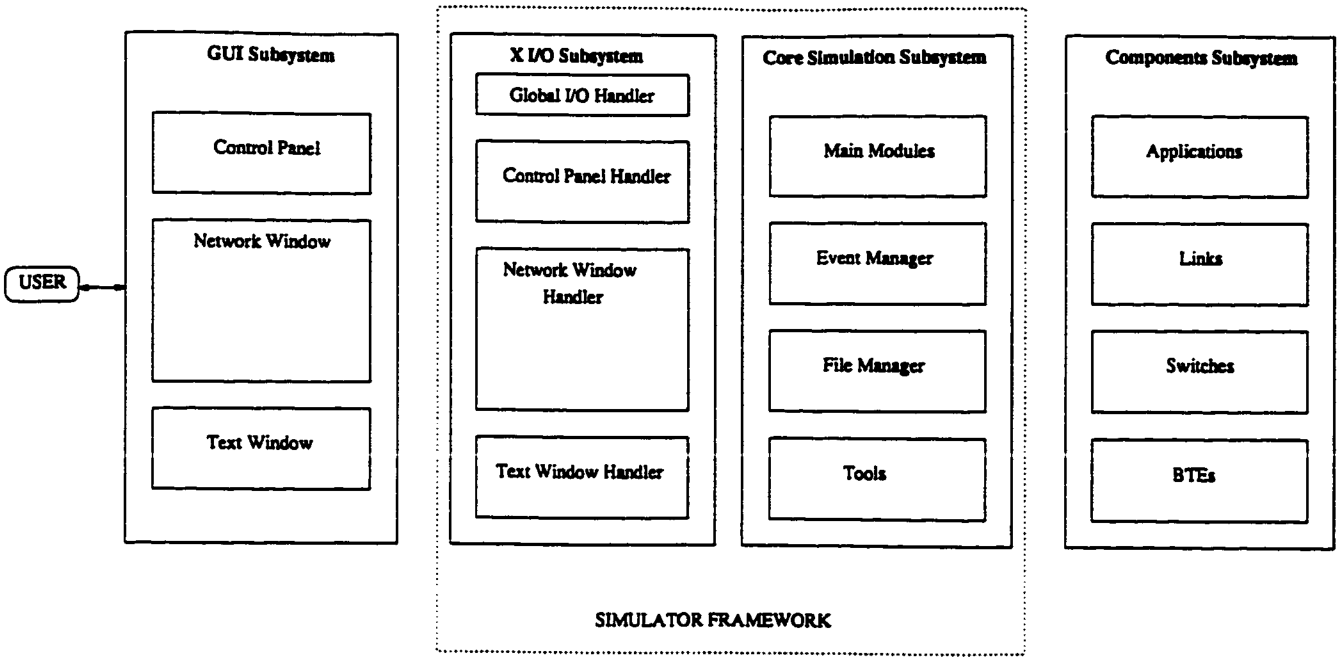


Figure 52: Architecture of the simulator

4.2 Changes Made to the Simulator

In this section we describe the changes made to the simulator. First we give the specifications of the changes made to the interface, then we look at the design and implementation of the internal system.

4.2.1 New Functionalities

A new class of component, the *GENERATOR* class, has been created to simulate a connection arrival process between two nodes. An instance is the *CBRGENERATOR*, which generates applications of the type *CBRCONNECTION* between two nodes with a Poisson arrival process. This component needs to be connected to the BTE source. The parameters of the CBRGENERATOR are as follows:

- *Name*: Name of component
- *Destination Name*: Name of the receiving application component (of type CBR-CONNECTION).
- *Bit Rate (MBits/s)*: Bandwidth requirement of connections generated.

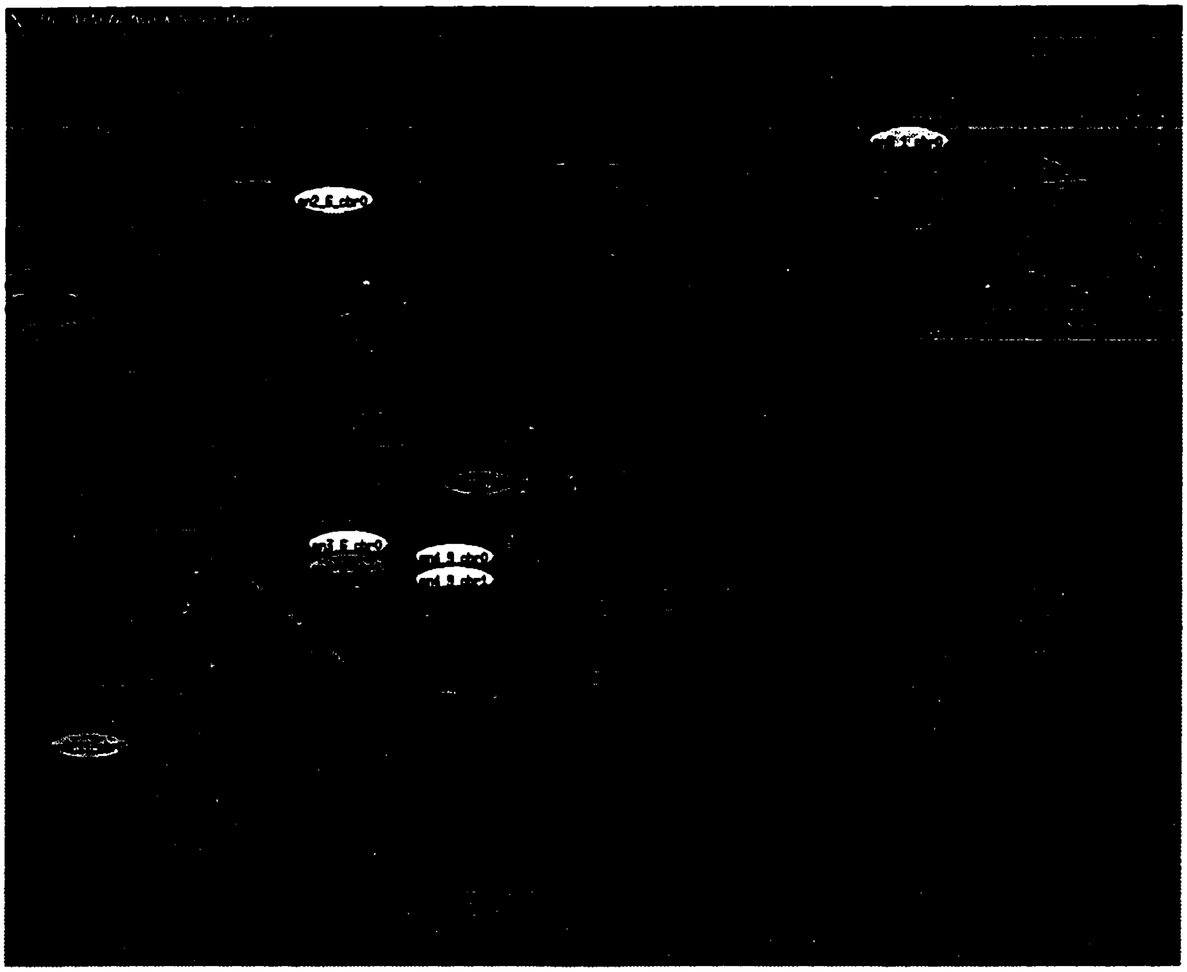


Figure 53: Simulator screen

- *Mean Arrival Rate (# of connections/s)*: Mean arrival rate of connections generated.
- *Mean Holding Time (usecs)*: Mean of the exponential distribution of holding times.
- *Traffic Type (Homogeneous:0, Het.: 1).*
- *Bit Rate 2 (Mbits/s)*

The output parameters of the CBRGENERATOR component are:

- *Number of arrivals*: cumulative number of arrivals.
- *Number of accepted arrivals*

Moreover, a special component, the *Connection Admission Control* component was also added. There is only one instance of this component during the simulation. It is used to gather statistics about the admission control algorithm, and appears on the bottom right corner of the simulator screen, as shown in Figure 53. Its input parameters are:

- *CAC type*
- *Routing Type*
- *Number of requests accepted*
- *Number of requests rejected*
- *Average Path length*

The command line has been updated to: **sim [-x] [-s seed] [-w warmuptime] [-c] [-f] [-a criterion] [-r routing] [worldfile [stoptime]]**

The new switches of the command line are described in Table 15.

4.2.2 Changes Made to the Architecture/Design Document

We have added the following modules to the core simulation subsystem of the simulator:

Command line switch	Definition
-c	the simulation is done at the call level only
-f	requests have the same start and release times
-w	used to specify the warmup time
-a	used to specify the admission control criterion
-r	used to specify the routing algorithm

Table 15: Switches of command line

- *GRAPH*
- *SHORTEST_PATH*
- *MHEAP*
- *ROUTING*
- *SCHEDULING*
- *RES_MANAGER*

The interfaces and dependencies of these modules are shown in Figure 54.

We have created an abstraction *GRAPH*, on which the routing algorithm is run, to make the implementation of the routing algorithm independent of the actual simulator data structures. The graph corresponding to the network topology is created when the first connection request is made. The *SHORTEST_PATH* module implements the path calculation. A new version of the heap has been implemented in a new module called *MHEAP*, which is a heap using two keys. It is used by the SHORTEST_PATH module in order to find shortest_paths based on two criteria. The *ROUTING* module provides an interface between the CAC module and the SHORTEST_PATH module. It sets the correct costs (based on number of hops or a function of the link utilization) and calls the appropriate shortest_path calculation algorithm. The *res_manager* manages the bandwidth reservations made on a link.

The original *HEAP* module, used by the event manager, was modified, to make sure that events scheduled at the same time will be fired in a deterministic order, namely a FIFO order. This was necessary to have exactly the same arrival pattern of requests for the same seed.

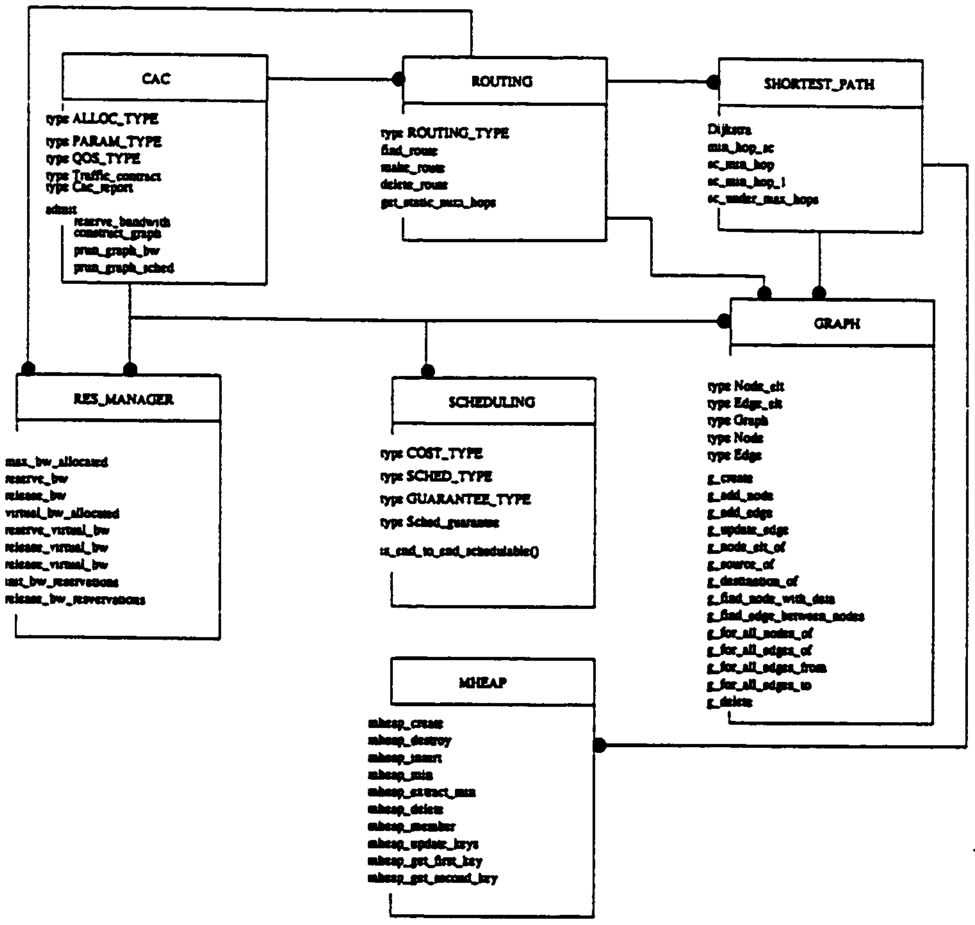


Figure 54: Interfaces

4.2.3 Tests and Validation

The modules Graph and Shortest_path were tested independently. The Generator and Reservation components were validated by running the Single Path routing algorithm on the NSFNet topology whose simulation results are given by Sibal and DeSimone [71].

Chapter 5

Conclusions and Future Work

In this thesis, we have studied the performance of routing and admission control algorithms in general topology networks. We make detailed observations about the behavior of EXP and the greedy algorithm in different settings. We propose a couple of new ideas to improve the unfairness of EXP and present the first experimental results on the newly proposed KPP algorithm.

Our results confirm the conclusions of previous studies that EXP has a lower rejection rate than the greedy algorithm especially at high loads. Our experiments show that this performance advantage is maintained in both topologies, over a variety of routing algorithms and reservation levels. Furthermore, even for a different profit measure, namely, the total number of hops traveled by accepted calls, EXP outperforms the greedy algorithm.

However, the difference in performance is diminished to negligible levels by varying some other parameters. For instance, the routing algorithm used appears to have an effect on the performance of both Greedy and EXP. While for greedy admission control, the routing algorithm using a cost exponential in the utilization works the best, combining EXP with routing algorithms such as Min-hop impacts its performance so much that it is outperformed by Greedy. Secondly, the traffic matrix, or the match of the traffic matrix to the topology also affects the relative performance of EXP and Greedy. For example, when the traffic matrix is not well matched and most of the traffic is between nodes at a distance close to the network diameter, the performance of the two algorithms is identical. Finally, when there are additional QoS constraints, such as stringent bounds on the end-to-end-delay, they seem to act as a kind of a reservation, thus equalizing the performance of Greedy and EXP.

We also examined the fairness of EXP and Greedy. It was found that on a variety of yardsticks, EXP was significantly more unfair than Greedy. In particular, the yardsticks used were the standard deviation of blocking probabilities, the mean blocking probability on a per-hop basis, and the variation in blocking probabilities for calls between nodes at the same distance. We proposed a new class of routing functions and admission control criteria called POWER(k) which, for carefully chosen values of k and the critical utilization level, substantially reduces this unfairness on a dense commercial network topology, while maintaining the same level of network throughput. We also showed that the same effect can be achieved by varying slightly the admission control criterion of EXP, in particular, by setting μ different from ρ.

In our experiments, the KPP algorithm was not found to outperform EXP in terms of mean blocking probability, which was found nearly identical to that of EXP on the commercial network topology. However, KPP is fairer than EXP, when we measure the fairness by the standard deviation of blocking probabilities. Another advantage of the KPP algorithm is that it is less sensitive than EXP to the parameter μ.

From our experiments it seems clear that a variety of factors (topology, traffic matrix, bandwidth requirements) make a difference to the performance of the admission control algorithm. Our conclusions regarding the relative performance of Greedy and EXP are somewhat limited by the number of experiments. In order to draw complete conclusions about the best algorithm to use for a given topology, a large number of experiments on different network topologies and traffic matrices would be required. It would be worthwhile to study the effects of these factors on KPP and the new admission control algorithms proposed, which have not been studied as extensively as EXP in this thesis.

It would be also interesting to study how these algorithms scale to more complex QoS routing models. For example, the call request could have a CTD requirement, but the bandwidth to reserve along the path could not be known *a-priori*, and could be dependent on the CTD and the number of hops of the path chosen (as it is the case for PGPS servers). Would a non-greedy strategy outperform a greedy strategy in this case? Finally, our algorithms are all centralized; making the routing and admission control algorithms distributed adds layers of complexity to the problem. Since, in practice, these algorithms will not in general, have centralized control, the distributed version of the problem deserves further study.

Appendix A

Traffic Matrices

A.1 Matrix for NSFNet Topology

The matrix of the NSFNet topology is shown in the following table:

$$\begin{pmatrix}
0 & 4 & 2 & 3 & 6 & 0 & 7 & 1 & 9 & 5 & 2 & 3 \\
4 & 0 & 3 & 6 & 13 & 0 & 15 & 2 & 19 & 9 & 4 & 6 \\
2 & 3 & 0 & 2 & 5 & 0 & 6 & 1 & 7 & 3 & 1 & 2 \\
3 & 6 & 2 & 0 & 8 & 0 & 10 & 1 & 12 & 6 & 2 & 4 \\
7 & 14 & 5 & 9 & 0 & 0 & 24 & 3 & 31 & 15 & 6 & 9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 17 & 6 & 11 & 25 & 0 & 0 & 4 & 37 & 18 & 7 & 11 \\
1 & 2 & 1 & 1 & 3 & 0 & 3 & 0 & 4 & 2 & 1 & 1 \\
11 & 22 & 9 & 15 & 33 & 0 & 39 & 5 & 0 & 24 & 9 & 15 \\
5 & 9 & 4 & 6 & 14 & 0 & 16 & 2 & 21 & 0 & 4 & 6 \\
2 & 4 & 1 & 2 & 5 & 0 & 6 & 1 & 8 & 4 & 0 & 2 \\
3 & 6 & 2 & 4 & 8 & 0 & 10 & 1 & 12 & 6 & 2 & 0
\end{pmatrix}$$

Table 16: Traffic matrix for the NSFNet topology

A.2 Matrix for ComNet Topology

$$\begin{pmatrix}
0 & 720 & 512 & 0 & 0 & 48 & 64 & 0 & 0 & 496 & 448 & 272 & 104 & 128 & 32 & 112 & 0 & 112 & 384 & 0 & 1072 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 32 & 128 & 128 & 64 & 88 & 624 & 432 & 48 & 64 & 848 & 64 & 160 & 0 & 64 & 160 & 32 & 416 & 48 & 0 & 0 & 32 \\
0 & 0 & 0 & 960 & 64 & 1136 & 632 & 768 & 0 & 416 & 256 & 112 & 520 & 128 & 256 & 240 & 88 & 48 & 128 & 0 & 640 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 64 & 544 & 128 & 0 & 0 & 864 & 0 & 56 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 128 & 0 \\
0 & 0 & 0 & 0 & 0 & 64 & 96 & 0 & 32 & 96 & 0 & 32 & 88 & 0 & 0 & 0 & 0 & 0 & 96 & 0 & 64 & 88 & 0 & 256 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 224 & 64 & 0 & 1808 & 96 & 104 & 184 & 0 & 0 & 0 & 56 & 80 & 128 & 32 & 336 & 32 & 0 & 256 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 192 & 32 & 512 & 0 & 584 & 456 & 144 & 16 & 0 & 0 & 608 & 64 & 32 & 392 & 64 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 304 & 0 & 88 & 104 & 720 & 64 & 336 & 512 & 96 & 256 & 0 & 592 & 32 & 0 & 0 & 176 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 400 & 0 & 128 & 232 & 0 & 0 & 192 & 0 & 0 & 128 & 16 & 208 & 0 & 0 & 0 & 32 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 848 & 0 & 96 & 192 & 48 & 256 & 80 & 992 & 0 & 1368 & 32 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 496 & 88 & 0 & 0 & 0 & 0 & 48 & 64 & 0 & 336 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 104 & 0 & 56 & 160 & 0 & 512 & 320 & 0 & 192 & 0 & 0 & 56 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56 & 144 & 0 & 16 & 0 & 104 & 152 & 56 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 112 & 672 & 224 & 0 & 576 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 160 & 0 & 0 & 256 & 0 & 0 & 256 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 48 & 208 & 80 & 32 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 256 & 0 & 0 & 0 & 64 & 56 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 320 & 112 & 800 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 32 & 384 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 80 & 48 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Table 17: Traffic matrix for the commercial network topology

Appendix B

Algorithm Implementation

B.1 MIN-HOP-1 Algorithm

In the pseudocode, we adopt the following notation

- V: set of nodes
- $w(u, v)$: cost of edge (u, v)
- *source*: source node
- *dest*: destination node
- $h[u]$: estimate of minimum number of hops from the source to node u.
- $h[u]$: estimate of cost of sc_min_hop path from source to u.
- $s'[u]$: estimate of cost of sc_min_hop+1 path from source to u.
- $p[u]$: estimate of predecessor of u in the sc_min_hop path from source to u.
- $p'[u]$: estimate of predecessor of u in the sc_min_hop path+1 from source to u.

```
for each node $x$ of $V$
    $h[x] \leftarrow s[x] \leftarrow s'[x] \leftarrow INFINITY$
    $p[x] \leftarrow p'[x] \leftarrow NIL$
$h[source] \leftarrow s[source] \leftarrow s'[source]$
$S \leftarrow V$
while $S \neq NIL$
        $u \leftarrow$ node with min $s$ among min $h$
        $S \leftarrow S - u$
        if $u \neq source$
          for each edge $(k, u)$ such that $k$ is in $S$
              if $h[k] = h[u]$
                 if $s'[u] > s[k] + w(k, u)$
                    $s'[u] \leftarrow s[k] + w(k, u)$
                    $p'[u] \leftarrow k$
        for each edge $(u, v)$ such that $v$ is in $S$
            if $h[v] = h[u]$ and $s'[v] > s[u] + w(u, v)$
              $p'[v] \leftarrow u$
              $s'[v] \leftarrow s[u] + w(u, v)$
            if $h[v] = h[u] + 1$ and $s'[v] > s'[u] + w(u, v)$
              $p'[v] \leftarrow u$
              $s'[v] \leftarrow s'[u] + w(u, v)$
            if $h[v] > h[u] + 1$
              if $p'[u] \neq NIL$
                 $p'[v] \leftarrow u$
                 $s'[v] \leftarrow s'[u] + w(u, v)$
            if $h[v] > h[u] + 1$ or
              $h[v] = h[u] + 1$ and $s[v] > s[u] + w(u, v)$
              /* case sc_min_hop path to v goes through u */
              $h[v] \leftarrow h[u] + 1$
              $s[v] \leftarrow s[u] + w(u, v)$
              $p[v] \leftarrow u$
```

Bibliography

[1] W. Aiello, M. Andrews, S. Bhatt, K. R. Krishnan, and L. Zhang. A performance comparison of competitive on-line routing and state-dependent routing. In *Proceedings of IEEE Globecom'97*, pages 1813–1819, Nov. 1997.

[2] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne. Real-time communication in packet-switched networks. *Proceedings of the IEEE*, 82(1):122–139, Jan 1994.

[3] G. R. Ash. *Dynamic Routing in Telecommunications Networks.* McGraw-Hill, 1997.

[4] J. Aspnes, Y. Azar, A. Fiat, and S. P. O. Waarrts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, volume 1, pages 623–631, 1993.

[5] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proceedings of the 34th IEEE Annual Symposium on the Foundations of Computer Science*, pages 32–40, Nov. 1993.

[6] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown durations. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms*, volume 1, pages 321–327, 1994.

[7] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proceedings of the 35th IEEE Annual Symposium on the Foundations of Computer Science*, Nov. 1994.

[8] J. C. R. Bennett and H. Zhang. *WF^2Q: Worst-case fair weighted fair queueing.* In *IEEE INFOCOM'96*, volume 1, pages 120–128, 1996.

[9] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[10] Çağlan M. Arras, D. S. Reeves, and R. C. Luo. Low latency, high acceptance real-time communication for wide area networks. Technical Report TR-092-010, North Carolina State University, Railegh, Dec. 1992.

[11] S. Chong and S.-Q. Li. (σ, ρ)-characterization based connection control for guaranteed services in high-speed networks. In *IEEE INFOCOM*, volume 1, pages 835–844, 1995.

[12] S. Chong, S.-Q. Li, and J. Ghosh. Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM. *IEEE Journal on Selected Areas in Communications*, 13(1):12–23, Jan. 1995.

[13] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, 1992.

[14] R. L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, Jan. 1991.

[15] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM'89*, pages 1–12, September 1989.

[16] M. Dickie. *Routing in Today's Internetworks: The Routing Protocols of IP, DECnet, NetWare, and AppleTalk.* Van Nostrand Reinhold, New York, 1994.

[17] A. I. Elwalid and D. Mitra. Effective bandwidth of general markovian traffic sources and admission control of high-speed networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, June 1993.

[18] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 9(3):368–379, April 1990.

[19] N. R. Figueira and J. Pasquale. Leave-in-time: A new service discipline for real-time communications in a packet-switching network. In *Proceedings of ACM SIGCOMM'95*, pages 207–218, 1995.

[20] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[21] J. A. Garay and I. S. Gopal. Call preemption in communication networks. In *IEEE INFOCOM*, volume 44, pages 1043–1050, 1992.

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

[23] M. W. Garrett. A service architecture for ATM: From applications to scheduling. *IEEE Network*, 10(3):6–14, May/June 1996.

[24] R. Gawlick, C. Kalmanek, and K. G. Ramakrishnan. On-line routing for permanent virtual circuits. *Computer Communications*, 19:235–244, 1996.

[25] R. Gawlick, A. Kamath, S. Plotkin, and K. G. Ramakrishnan. Routing and admission control in general topology networks. Technical Report STAN-CS-TR-95-1548, Stanford University, Stanford, CA, 1995.

[26] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[27] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks.* Addison-Wesley, 1990.

[28] S. J. Golestani. A framing stategy for congestion management. *IEEE Journal on Selected Areas in Communications*, 9(7):1064–1077, Sept. 1991.

[29] S. J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *IEEE INFOCOM*, volume 2, pages 636–646, 1994.

[30] N. Golmie, A. Koenig, and D. Su. The NIST ATM Network Simulator, Operation and Programming, Version 1.0. Internal Report 5703, NIST, Aug. 1995.

[31] P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digtal Audio and Video*, pages 287–298, Durham, New Hampshire, April 1995.

[32] P. Goyal, H. M. Vin, and H. Cheng. Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks. In *Proceedings of ACM SIGCOMM'96*, pages 157–168, 1996.

[33] R. Guérin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968–981, Sept. 1991.

[34] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS routing mechanisms and ospf extensions. *Internet Draft*, January 1998.

[35] J.-H. Huang and P.-C. Tsao. Continuous framing mechanism for congestion control in broadband networks. *Computer Communications*, 18(10):718–724, Oct. 1995.

[36] A. Hung and G. Kesidis. Bandwidth scheduling for wide-area ATM networks using virtual finishing times. *IEEE/ACM Transactions on Networking*, 4(1):49–54, February 1996.

[37] B. R. Hurley, C. J. R. Seidl, and W. F. Sewell. A survey of dynamic routing methods for circuit-switched traffic. *IEEE Communications Magazine*, 25(9):13–21, 1987.

[38] A. Iwata, R. Izmailov, D.-S. Lee, B. Sengupta, G. Ramamurthy, and H. Suzuki. ATM routing algorithms with multiple QOS requirements for multimedia internetworking. *IEICE Transactions on Communications*, E79-B(8):999–1007, Aug. 1996.

[39] D. L. Jagerman. Methods in traffic calculations. *AT&T Bell Laboratories Technical Journal*, 63(7):1283–1310, Sept 1984.

[40] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *Proceedings of ACM SIGCOMM'95*, pages 2–13, 1995.

[41] C. R. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proceedings of IEEE Globecom'90*, pages 12–20, Dec. 1990.

[42] A. Kamath, O. Palmon, and S. Plotkin. Routing and admission control in general topology networks with poisson arrivals. *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1996.

[43] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *11th International Conference on Distributed Computing Systems (DCS)*, pages 300–307, Arlington, TX, 1991.

[44] P. Klein, S. Plotkin, C. Stein, and Éva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, 1994.

[45] E. Knightly and H. Zhang. Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models. *IEEE INFOCOM*, pages 1137–1145, 1995.

[46] E. W. Knightly. H-BIND: A new approach to providing statistical performance guarantees to VBR traffic. Technical Report TR-95-034, University of California at Berkeley, 1995.

[47] D. Kofman and M. Gagnaire. *Réseaux haut débit: réseaux ATM, réseaux locaux, réseaux tout-optiques*. InterEditions, 1996.

[48] S.-K. Kweon and K. G. Shin. Traffic-controlled rate-monotonic priority scheduling of ATM cells. In *IEEE INFOCOM*, volume 2, pages 655–662, 1996.

[49] J.-Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[50] W. C. Lee, M. G. Hluchyj, and P. A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network*, 9(4):46–55, July/August 1995.

[51] T. Leighton, F. Makedon, S. Plotkin, C. Stein, Éva Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50:228–243, 1995.

[52] I. Matta and M. Krunz. Packing and least-loaded based routing in multi-rate loss networks. Technical Report NU-CCS-96-12, Northeastern University, Boston, 1996.

[53] I. Matta and A. U. Shankar. Dynamic routing of real-time virtual circuits. Technical Report NU-CCS-96-07, Northeastern University, 1996.

[54] D. Mitra, R. J. Gibbens, and D. H. Huang. State-dependent routing on symmetric loss networks with trunk reservations–I. *IEEE Transactions on Communications*, 41(2), 1993.

[55] D. Mitra and J. B. Seery. Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks. *IEEE Journal on Selected Areas in Communications*, 39(1):102–116, Jan. 1991.

[56] W. M. Moh, M.-J. Chen, N.-M. Chu, and C.-D. Liao. Traffic prediction and dynamic bandwidth allocation over ATM: a neural network approach. *Computer Communications*, 18(8):563–571, Aug 1995.

[57] R. O. Onvural. *Asynchronous Transfer Mode Networks : Performance Issues.* Artech House, Boston, 1995.

[58] T. Ott and K. Krishnan. State dependent routing and the use of separable routing schemes. In *Proceedings of the 11th International Teletraffic Congress (Kyoto, Japan)*, volume 1, 1985.

[59] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[60] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[61] C. Partridge. *Gigabit Networking.* Addison-Wesley, 1994.

[62] A. Pitsillides, J. Lambert, and D. Tipper. Dynamic bandwidth allocation in broadband-ISDN using a multilevel optimal control approach. In *IEEE INFO-COM'95*, volume 3, pages 1086–1094, 1995.

[63] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1128–1136, Aug. 1995. Invited Paper.

[64] C. Pornavalai, G. Chakraborty, and N. Shiratori. QoS routing algorithms for pre-computed paths. *IEEE INFOCOM*, pages 248–251, 1997.

[65] S. Rampal and D. S. Reeves. Routing and admission control algorithms for multimedia traffic. *Computer Communications*, 18(10):755–768, Oct 1995.

[66] E. P. Rathgeb. Modeling and performance comparison of policing mechanisms for ATM networks. *IEEE Journal on Selected Areas in Communications*, 9(3):325–334, Apr. 1991.

[67] D. Saha, S. Mukherjee, and S. K. Tripathi. Carry-over round robin: A simple cell scheduling mechanism for ATM networks. In *IEEE INFOCOM'96*, volume 1, pages 630–637, 1996.

[68] H. F. Salama, D. S. Reeves, and Y. Viniotis. An efficient delay-constrained minimum spanning tree heuristic. In *Proceedings of the 5th International Conference on Computer Communications and Networking*, October 1996.

[69] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. *IEEE INFOCOM*, 1997.

[70] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. In *Proceedings of ACM SIGCOMM'95*, pages 231–242, 1995.

[71] S. Sibal and A. DeSimone. Controlling alternate routing in general-mesh packet flow networks. In *Proceedings of ACM SIGCOMM'94*, pages 168–179, 1994.

[72] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, Feb. 1985.

[73] S. Suri, G. Varghese, and G. P. Chandranmenon. Leap forward virtual clock: An $O(loglogN)$ fair queuing with guaranteed delays and throughput fairness. Technical Report WUCS-96-10, Washington University, St. Louis, 1996.

[74] D. C. Verma, H. Zhang, and D. Ferrari. Delay jitter control for real-time communication in a packet switching network. In *Proceedings IEEE TriCom'91*, pages 35–42, April 1991.

[75] M. Vishnu and J. W. Mark. HOL-EDD: A flexible service scheduling scheme for ATM networks. In *IEEE INFOCOM*, volume 2, pages 647–654, 1996.

[76] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, Sept. 1996.

[77] R. Widyono. The design and evaluation of routing algorithms for real-time channels. Technical Report TR-94-024, University of California at Berkeley and International Computer Science Institute, June 1994.

[78] D. E. Wrege, E. W. Knightly, H. Zhang, and J. Liebeherr. Deterministic delay bounds for VBR video in packet-switching networks: Fundamentals limits and practical trade-offs. *IEEE/ACM Transactions on Networking*, 4(3):352–362, June 1996.

[79] G. G. Xie and S. S. Lam. Delay guarantee of virtual clock server. *IEEE/ACM Transactions on Networking*, 3(6):683–639, Dec 1995.

[80] H. Zhang and E. W. Knightly. RCSP and stop-and-go: A comparison of two non-work-conserving disciplines for supporting multimedia applications. *Multimedia Systems Journal*, 1998.

[81] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, 1990.

[82] Z.-L. Zhang, Z. Liu, J. Kurose, and D. Towsley. Call admission control schemes under the generalized processor sharing scheduling. Technical Report 2711, INRIA, November 1995.