Admission Control and Routing: Theory and Practice

by

Rainer Gawlick

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1992)
B.A., Physics
University of California - Berkeley
(1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Signature of Author ______________________________
Department of Electrical Engineering and Computer Science
June 14, 1995

Certified by ______________________________
Nancy A. Lynch
Professor of Computer Science
Thesis Supervisor

Certified by ______________________________
David D. Clark
Senior Research Scientist
Thesis Supervisor

Certified by ______________________________
Baruch Awerbuch
Professor of Computer Science, Johns Hopkins University
Thesis Supervisor

Accepted by ______________________________
Frederic R. Morgenthaler
Chair, Departmental Committee on Graduate Students

Admission Control and Routing: Theory and Practice

by

Rainer Gawlick

Submitted to the Department of Electrical Engineering and Computer Science

on June 14, 1995, in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Emerging high speed Broadband Integrated Services Digital Networks (B-ISDN) will carry traffic for services such as video-on-demand and video teleconferencing, which require resource reservation along the path on which the traffic is sent. As a result, such networks will need effective admission control algorithms. The simplest approach is to use *greedy* admission control; in other words, accept every resource request that can be physically accommodated. However, *non-greedy* admission control can lead to better network performance in many situations.

This thesis develops several non-greedy algorithms that out-perform greedy admission control algorithms. Some of the algorithms are evaluated using simulations while others are evaluated using the theory of competitive analysis. The thesis considers both unicast communication (connections with two end-points) and multicast communication (connection with more than two end-points).

The results of the thesis have already had a direct influence on the algorithms used in several commercial networks.

Thesis Supervisor: Nancy A. Lynch

Title: Professor of Computer Science

Thesis Supervisor: David D. Clark

Title: Senior Research Scientist

Thesis Supervisor: Baruch Awerbuch

Title: Professor of Computer Science, Johns Hopkins University

Keywords: ATM, Virtual Circuit Networks, BISDN, Multimedia Networks, Interconnection Networks for Massively Parallel Computers, Multicast Communication, Broadcast Communication, Routing, Admission Control, Competitive Analysis, Competitive Ratio, Lower Bounds, Simulation, On-Line Algorithms, Randomized Algorithms

Acknowledgements

My greatest thanks go to my thesis supervisors. Baruch Awerbuch taught me much about algorithms and competitive analysis. David Clark provided me with many insights into the practical issues that arise in large scale networks. He helped me identify the ways in which the theoretical analysis in the thesis informs the design of practical algorithms. Nancy Lynch taught me much about research and careful reasoning during my years of graduate school.

Most parts of this thesis were co-authored. Baruch Awerbuch, Yuval Rabani, and Tom Leighton contributed to Chapters 3 and 4. Tom Leighton also provided comments that significantly improved the presentation in Section 3.5. Serge Plotkin, Ram Ramakrishnan, and Anil Kamath contributed to Chapters 5 and 6. Finally, Baruch Awerbuch and Yossi Azar contributed to Chapter 7. I thank all my co-authors for the many fruitful hours of discussions.

The work in this thesis was done at the M.I.T. Laboratory for Computer Science and at AT&T Bell Laboratories. I would like to thank all of those who helped make M.I.T. and AT&T such productive and fun places. At M.I.T. I would like to especially acknowledge Roberto Segala, with whom I shared many classes, a paper, and countless discussions. At AT&T special thanks go to Debasis Mitra for his personal and professional support. Ram Ramakrishnan, Chuck Kalmenek, and Bala Rajagopalan also contributed generously with their time and insights.

I also owe a great deal to my parents, Mechthild and Dieter, whose devotion, support, and emphasis on education laid the foundation for my pursuit of a graduate degree. Finally, my deepest thanks go to my wife, Christine, for many years of happiness and for her constant support as I was writing this thesis.

Table of Contents

CHAPTER 1

Introduction

1.1 Circuit Networks

The problem of virtual circuit admission control and routing can arise whenever there is a request to send a large amount of data from one node in a network to another node. The *admission control* aspect of the problem is to decide whether or not the network can or should accommodate the request, and the *routing* aspect of the problem is to decide how to route the data if the request is to be accommodated. The data is routed by establishing a path, called a *virtual circuit*, through the network that connects the two nodes that wish to communicate. The data packets are then sent along the established virtual circuit. We will focus on networks where virtual circuits are characterized by a bandwidth requirement and the bandwidth required for the virtual circuit is reserved explicitly. This thesis provides several virtual circuit admission control and routing algorithms. Furthermore, it provides several lower bounds on the performance of any virtual circuit admission control and routing algorithm. The admission control and routing problem arises in many contexts. We survey a few of these.

Future Broadband Integrated Services Digital Networks (B-ISDN) will carry a wide spectrum of new consumer services, such as video-on-demand, video teleconferencing [Sha94], etc. A key characteristic of these services is that they require quality-of-service (QoS) guarantees. Assuring QoS requires reservation of resources. As a result, B-ISDN will likely allocate resources in terms of virtual circuits. Examples of broadband networking technology that uses a virtual circuit based approach are ATM (Asynchronous Transfer Mode) [deP91, Bou92] and PARIS/PlaNET [CG88, CGG91].

The Internet also seems to be moving in the direction of using virtual circuits. Traditionally, nodes communicating on the Internet may use a different path for each data packet. As a result, there are no performance guarantees and no resource reservations. However, new routing architectures currently being developed for the Internet, e.g., NIMROD [CCS94], include facilities for virtual circuit routing. Furthermore, protocols are being developed for resource reservation and for signalling admission control decisions, e.g., RSVP [ZDE+93].

Parallel supercomputers are another important area where admission control and routing problems arise. Applications on parallel supercomputers often need fast access to potentially large amounts of data that is stored remotely. Hence, there needs to be a communications network embedded in the supercomputer that is capable of supporting such requests for data. In some supercomputers this data is routed using some form of virtual circuit routing. (In the past, many supercomputers have used packet routing where each packet uses its own path and no bandwidth reservations can be made. While this approach works for cooperative scientific applications, it may not be effective in commercial applications where the various tasks may not be cooperating. By reserving a certain amount of bandwidth on a virtual circuit, a particular task can be assured good performance.) We note that the supercomputer community has recently shown interest in constructing systems by interconnecting workstation-like nodes via high speed LANs [Lei93]. IBM's SP-2 is an example of such a supercomputer system. In view of the emergence of the ATM standard, which is based on virtual circuits, as a preferred architecture for high speed data networks, virtual circuit admission control and routing algorithms may become increasingly important for future supercomputers.

A large-scale video server can be constructed by using a supercomputer network to connect a large disk farm to a set of telecommunications lines. The network of the supercomputer is then used to route video (e.g., movies) to subscribers in real time. Oracle's Media Server, which currently runs on the NCube supercomputer, is an example of such a system [Buc94]. Each customer has a virtual circuit through the NCube that connects the disk containing the customer's movie to the customer's telecommunications port.

The admission control and routing problem encompasses many service models. We focus on the following models. We characterize a virtual circuit by its participating nodes and a bandwidth. *Unicast* communication refers to circuits that have two participating nodes (traditionally called a source and a destination). *Multicast* communication refers to circuits that have more than two participating nodes. This thesis considers both types of circuits. Multicast communication can be further subdivided into *on-line* multicast groups and *off-line* multicast groups. For on-line multicast groups all participants arrive and leave at the same time (e.g., a teleconference call). For off-line multicast groups participants arrive and leave independently (e.g., ESPN). We develop admission control and routing algorithms for both types of multicast groups.

When virtual circuit requests (unicast or multicast) arrive to the network we immediately make the admission control and routing decision. Once made, that decision is not changed. Thus, we do not consider rerouting or preemption. (Preemption removes an existing virtual circuit from the network in favor of new, potentially more valuable, virtual circuits.)

In characterizing a virtual circuit by its participating nodes and a bandwidth, we have simplified several issues. For example, we are ignoring the stochastic properties of bursty connections. In particular, many data sources, such as video, will have bandwidth requirements that can vary dramatically over short time frames. Furthermore, some virtual circuits may want to reserves other resources in addition to bandwidth. For example, buffer space might be important for a circuit that wishes to have low data loss. A favorable position in a priority system may be important for a circuit that needs low delay. Many of these complications can be abstracted away using the concept of *effective bandwidth* [AG90, GAN90, EM93, EM95] (Some of the literature, e.g., [AG90, GAN90] uses the term *equivalent bandwidth*). Effective bandwidth essentially determines how much bandwidth a circuit should reserve based on the stochastic properties of the circuit and the QoS requirements (e.g., packet loss, delay) of the circuit. We assume that circuit requests specify their effective bandwidth.

Our model for multicast communication is also simplified. In particular, we construct a single multicast tree for the entire multicast group. If the multicast group has multiple sources, it will have to use its own mechanism to manage the coordination of the transmission on its tree. A multiple source multicast group could establish a separate multicast group for each source. However, our model does not provide for any coordination of admission control decisions across multicast groups.

1.2 Goals

Admission control and routing algorithms can have a variety of goals. We focus on the following two goals: maximize the number of accepted virtual circuit requests and maximize the total amount of accepted bandwidth. The total amount of accepted bandwidth is the sum of the requested bandwidth of the accepted virtual circuits. We note that these optimization goals ignore certain fairness issues. For example, node pairs that are only connected by long paths may experience a higher rate of rejection than node pairs that are connected by short paths. Other optimization goals, such as minimizing the total delay in the network [BG92] or maximizing the total accepted "value", when each virtual circuit is characterized by a "value" [AAP93], have also been discussed in the literature.

The traffic patterns for the networks and the applications described in Section 1.1 will not be known in advance since the usage patterns for these applications and networks are currently not well understood. Furthermore, the traffic patterns can vary dramatically over short periods of time. Hence, advance knowledge of the traffic pattern may be difficult to obtain. Therefore, an important goal for our algorithms is to not require advance knowledge of the traffic pattern. This design goal motivates both our use of concepts from competitive analysis and the manner in which we make use of stochastic properties. Our approach stands in contrast to existing algorithms for general topology networks, which all require advance knowledge about the traffic patterns.

As a secondary goal, we wish to minimize use of dynamic state information. The use of detailed dynamic state information such as current link utilizations can significantly complicate the implementation of the algorithm in a distributed setting. We seek to minimize the use of that information by using static state information, e.g., number of links, to decide among the paths that meet the admission control criteria.

1.3 Approach

Non-greedy admission control and routing. The simplest approach to admission control and routing is to make the decisions in a *greedy* manner. In other words, always route a circuit if there is a path with sufficient bandwidth; always use the minimum-hop path among the paths with sufficient bandwidth. Unfortunately, the greedy approach can lead to poor performance. For example, it will accept a virtual circuit request even if that request can only be accommodated along an excessively long path that might be more efficiently used by some future virtual circuits. The alternative is to use *non-greedy* admission control and routing. Non-greedy admission control and routing may choose to reject a circuit request even if there exists a path with sufficient bandwidth. Furthermore, it may not use the shortest path with sufficient bandwidth. Non-greedy admission control can be used to improve the number of virtual circuits that the network accepts. This observation was first made in the context of symmetric loss networks by Krupp [Kru82]. More recently, this observation has been extended to general topology networks [OK85, Kel88, SD94, GKR95]. The admission control and routing algorithms that we develop all use a non-greedy approach. Our simulations and our lower bound results lend further support to the notion that non-greedy approaches provide superior performance.

Competitive analysis. Admission control and routing decisions need to be made in an *on-line* fashion. In particular, each decision must be made without knowledge of future requests. *Competitive analysis* is an important theoretical framework in which the performance of on-line algorithms is analyzed [ST85, KMRS88]. The performance measure used in competitive analysis is the *competitive ratio*. The competitive ratio of an on-line virtual circuit routing and admission control algorithm is the maximum over all request sequences of the ratio of the number of requests accepted by the optimal algorithm for that sequence to the number of requests accepted by the on-line algorithm for the same sequence. An algorithm with a low competitive ratio is one that performs close to the optimal algorithm on all request sequences. Informally, the competitive ratio measures how much the performance of the on-line algorithm suffers in comparison to the optimal algorithm due to the fact that the on-line algorithm cannot predict future requests, since, for example, it does not know the traffic pattern.

Competitive analysis can be extended to randomized algorithms [BLS87], i.e., algorithms that use randomization in their decision process. Let $E[A(\sigma)]$ be the expected performance of randomized algorithm A on request sequence σ. Then the competitive ratio for A is the maximum over all request sequences σ of $O(\sigma)/E[A(\sigma)]$, where $O(\sigma)$ is the performance of the optimal off-line algorithm on request sequence σ. This competitive ratio is called *oblivious* since the request sequence is chosen independently of the random choices made by A.

An important motivation for using competitive analysis is the fact that it does not make assumptions about the circuit requests, such as assumptions about the traffic pattern. (Recall the goals from Section 1.2.) Since it does not make any assumptions, competitive analysis provides a robust *worst-case* performance measure. However, an algorithm that leads to the best worst-case performance clearly may not lead to the best performance in practice. The ultimate goal of the thesis is to provide *practical* admission control and routing algorithms. Thus, in an effort to construct practical algorithms, we will modify some of the algorithms that have a good competitive ratio using heuristics that work well in practice. We make these modifications in spite of the fact that the heuristics compromise the theoretical performance, i.e., competitive ratio, of the algorithm. The theory in this thesis should be viewed as providing general algorithmic principles rather than specific algorithms.

Stochastic analysis. Stochastic analysis is used to motivate some of the heuristics that we use in our algorithms. In particular, we use some techniques developed in the context of symmetric loss networks to estimate the expected effect of routing a particular virtual circuit on the likelihood that future virtual circuits must be rejected due to capacity constraints.

1.4 Related Work

We divide our discussion of previous work into three areas: competitive analysis, statistical approaches, and multicast communication.

Competitive analysis. Virtual circuit admission control and routing has been considered extensively in the context of competitive analysis. Garay and Gopal [GG92] and Garay, Gopal, Kutten, Mansour, and Yung [GIK+92] developed competitive algorithms for admission control and routing in a scenario where preemption is allowed and the network is constrained to be a straight line. When preemption is allowed, the network may decide to terminate any virtual circuit at any time. Preemption is undesirable in most of the applications (such as video-servers) that we mention in Section 1.1.

Awerbuch, Azar and Plotkin [AAP93] develop competitive algorithms for general networks, but with the restriction that every virtual circuit request at most $1/\log n$ of the capacity of the lowest capacity link. They provide an $O(\log n)$ competitive algorithm, where n is the number of nodes in the network.

Aspnes, Azar, Fiat, Plotkin and Waarts [AAF+93] consider a slightly different model. Here there is no admission control problem since all requests are accepted. [AAF+93] presents a competitive algorithm that on any link requires at most $O(\log n)$ more capacity than is required by the optimal off-line algorithm, where n is the number of nodes in the network. The virtual circuits in [AAF+93] all have infinite duration. The result is extended to virtual circuit with finite duration in [AKP+93]. Both [AAF+93] and [AAP93] use minimum cost routing where the cost metric is an exponential function of the link utilization [SM90].

In [ABFR94] Awerbuch, Bartal, Fiat and Rosen consider the admission control and virtual circuit routing problem on trees. Their basic algorithm focuses on virtual circuits that request the entire bandwidth of a link and have infinite duration. The algorithm is randomized and has an $O(\log n)$ competitive ratio. For the line, they show a matching lower bound of $\Omega(\log n)$. By combining their basic algorithm with the algorithm in [AAP93], the authors provide an $O(\log^2 n)$ competitive algorithm for virtual circuits of arbitrary bandwidth. Blum, Fiat, Karloff, and Rabani report a deterministic $O(n)$ algorithm for the $n \times n$ mesh, a deterministic $\Omega(\sqrt{n})$ lower bound on the $n \times n$ mesh, and an $O(\log n)$ deterministic algorithm with preemption for n node trees [BFKR93].

Statistical approaches. Non-greedy admission control was first considered in the context of symmetric loss networks. Symmetric loss networks have a complete graph topology and equal capacity on each link. Furthermore, each source/destination pair has the same rate of virtual circuit arrivals. The virtual circuits have a Poisson arrival process and an exponential departure process. Virtual circuits always use the direct link if it is available, otherwise they try to find an available path consisting of two links; paths consisting of more than two links are not considered. Admission control on direct paths is greedy. However, a two-link path is used only if both links have sufficiently low utilization. The importance of using non-greedy admission control for the two-link paths in symmetric loss networks was first discussed by Krupp [Kru82] and has since received extensive attention [Aki84, MS91, MG92, MGH93]. An example of an admission control scheme based on the ideas developed in the context of symmetric loss networks is the Real Time Network Routing algorithm (RTNR) Ash et. al. [ACF+92] used in the AT&T long distance network.

A key advantage of symmetric loss networks is that they permit a detailed analytic analysis. In particular, symmetric less networks tend to be modeled by fixed point equations that are easily solved numerically [Kat67, Aki84, MGH93]. Using the numerical solutions to the fixed point equations one can determine the utilization levels at which two-link paths should no longer be used. Mitra and Gibbens [MG92] provide analytic results (i.e. non-numeric) for certain asymptotic regions. Some initial work on extending the fixed point techniques to general topology networks has been done by Greenberg and Srikant [GS95].

Statistical approaches to routing and admission control for general topology networks have also received attention. A cost based routing algorithm for general topology networks was developed by Ott and Krishnan [OK85]. Roughly speaking, their algorithm is based on the concept of costs that reflect the expected effect of routing and admission control decisions on the system performance. The expected effect is determined by making statistical assumptions about the virtual circuit arrival processes as well as using advance knowledge about the traffic patterns. Their algorithm requires *complete* current state information for its path selection. An alternative approach was recently proposed by Sibal and DeSimone [SD94]. Their approach does not use cost functions. Rather, it is an extension of the ideas from symmetric loss networks to general topology networks. Specifically, like algorithms from symmetric loss networks they determine threshold utilization values above which only "direct" traffic is permitted to use a link. In their algorithm, the path selection is based on static criteria, with dynamic state information relevant only to the actual admission control decision. However, their admission control criteria still require advance knowledge of the traffic pattern. In particular, the traffic pattern is used to calculate the threshold values of the utilization.

Multicast. There has been much work on multicast communication that has focused on membership protocols and multicast tree maintenance in the face of dynamic membership. However, we are not aware of any work that directly addresses the issue of admission control for multicast communication. However, we point to some related work. The RSVP protocol [ZDE+93] is a signalling mechanism for admission control. The RSVP protocol assumes the existence of a separate mechanism that makes the actual admission control decisions. Other related work by Herzog, Shenker, and Estrin [HSE95] provides a mechanism for distributing the cost associated with a multicast tree among the members of the multicast group. Such a mechanism might be useful as a component of a cost based admission control algorithm.

An influential set of distributed multicast protocols was developed by Deering et. al. [WDP88, DC90, Dee91, Moy92]. The focus of these protocols is the maintenance of the multicast tree in the presence of dynamically changing multicast membership. In particular, for each multicast group the protocols periodically check each region of the network to determine if nodes in that region wish to be members of the multicast group. If some do, the packets for that multicast group are send to the new members. The protocols were developed for the Internet environment. (They are currently being used for the MBONE [Cas94].) As a result, no resources are reserved when a new node joins a multicast group. Furthermore, the protocols have no admission control feature. For routing, the protocols make use of the underlying unicast routing mechanism. In particular, the tree established by the protocols is a shortest path tree, where the shortest path is determined by the unicast routing mechanisms.

For the purpose of admission control and routing for multicast communication, the Core Based Tree (CBT) approach of Ballardie, Tsuchiya and Crowcroft [BTC92] is essentially the same as the Deering approach. However the CBT mechanism for maintaining the multicast tree in the presence of dynamically changing multicast membership is more efficient when only a small percentage of the nodes are members of the multicast group. The Deering approach in [WDP88, DC90, Dee91, Moy92] and the Core Based Tree approach are combined in Protocol Independent Multicast (PIM) [DEF+94].

More sophisticated approaches to routing of multicast groups are described by Noronha and Tobagi [NT94]. Their paper describes various routing algorithms for multicast groups that attempt to optimize both cost and delay considerations. Furthermore, they take link capacity considerations into account. In particular, each multicast group has a bandwidth associated with it. That bandwidth is reserved along the tree picked for the multicast group. Their admission control procedure is greedy. The algorithms in [NT94] are evaluated using simulations. The work by Verma and Gopal [VG93] considers the additional issue of multicast groups that have different bandwidth requirements for traffic to and from the source. They develop various heuristic approaches and evaluate them with simulations. A summary of additional work on multicast routing strategies can be found in [NT94].

1.5 Lower Bounds

This thesis provides several lower bounds in the context of competitive analysis of on-line algorithms. In particular, we provide some lower bounds on the competitive ratio that any admission control and routing algorithm can achieve. In other words, no admission control and routing algorithm can achieve a competitive ratio that is better (lower) than the competitive ratio given by the lower bound. These lower bounds measure the total number of virtual circuits that are accepted. We provide an $\Omega(\log d)$ lower bound on the oblivious competitive ratio for lines of length d and trees of diameter d. For $n \times n$ meshes we provide an $\Omega(\log n)$ lower bound on the oblivious competitive ratio. For the $\log n$ dimensional hypercube we prove an $\Omega(\log \log n)$ lower bound on the oblivious competitive ratio. Finally, we prove an $\Omega(\log \log n)$ lower bound on the oblivious competitive ratio for n^2-leaf trees of meshes and n^2-leaf fat-trees. In each case, the lower bound for a greedy admission control and routing algorithm is higher than the oblivious lower bound that we provide on all algorithms. Furthermore, for trees and meshes, the lower bound on the deterministic competitive ratio higher than the oblivious lower bound. This suggests that non-greedy approaches can lead to better performance.

We also define a network type called a *hierarchical backbone network*. Informally, a hierarchical backbone network is a network that can be decomposed into a set of *access networks* and a *backbone network* that connects the access networks. We show that a competitive ratio lower bound that applies to the backbone network or any of the access networks implies the same competitive ratio lower bound for the entire hierarchical backbone network.

Finally, we provide an $\Omega(n)$ lower bound on *greedy* admission control and routing algorithms for n node general topology networks.

1.6 Algorithms

The thesis provides several algorithms for admission control and routing. These algorithms all use non-greedy approaches. The first set of algorithms complements the lower bounds listed in Section 1.5. These algorithms are primarily of theoretical interest since they focus on the case where all requests have the same bandwidth and each virtual circuit has infinite duration.

For trees with radius d, we provide a randomized admission control and routing algorithm with a competitive ratio of $O(\log d)$. This matches the $\Omega(\log d)$ lower bound. Furthermore, the algorithm overcomes the trivial $\Omega(d)$ lower bound on the competitive ratio for deterministic and greedy algorithms on trees. The n^2-leaf fat-tree can be seen as a special case of a hierarchical backbone network. We present a general technique for constructing admission control and routing algorithms for hierarchical backbone networks. Using this technique, we develop an algorithm that achieves an $O(\log \log n)$ competitive ratio for n^2-leaf fat-trees. This matches our $\Omega(\log \log n)$ lower bound.

This thesis also presents a very practical algorithm for admission control and routing on general topology networks. We call this algorithm EXP. The EXP algorithm is based on the general topology algorithm in [AAP93]. The algorithm in [AAP93] addresses two of our goals (cf. Section 1.2). It does not require advance knowledge of the traffic pattern, and it uses non-greedy admission control to maximize the number of accepted requests. Unfortunately, the algorithm in [AAP93] has several disadvantages that prevent it from being practical. First, the algorithm deals only with admission control and does not address routing. Second, it requires that each circuit request specify its duration. Third, each link must maintain and distribute large amounts of state information. Finally, the algorithm is optimized for the worst-case situation and does not work well in common situations. The EXP algorithm substantially modifies the algorithm in [AAP93] to address its shortcomings.

Finally, we provide non-greedy admission control and routing algorithms for multicast communication. We provide three algorithms. Each of the algorithms seeks to maximize the total amount of accepted bandwidth. The first algorithm, which achieves a competitive ratio of $O(\log^2 n)$, only considers batched multicast groups. The second algorithm, which has a competitive ratio of $O(\log^6 n)$, allows both batched and a restricted form of on-line multicast groups. Specifically, the on-line requests from different multicast groups may not be interleaved. In other words, all requests to join a specific multicast group occur without any intervening requests to join another multicast group. Our third algorithm removes this restriction for on-line multicast groups. That algorithm also achieves a competitive ratio of $O(\log^6 n)$. While our algorithms for multicast communication achieve a good competitive ratio, they will not perform well in practice if used in their present form. However, we believe that the algorithms will serve as the basis for a practical algorithm in much the same way that the algorithm in [AAP93] does. In fact, we believe that many of the techniques developed for the EXP algorithm will also be useful for multicast communications. We discuss some of the issues that arise in constructing practical multicast admission control and routing algorithms.

1.7 Simulations

This thesis provides an extensive set of simulations to evaluate the performance of our EXP algorithm over a wide range of situations. The simulations are based on an existing commercial data network and some artificially generated networks. Among other things, we explore the effect of circuit bandwidths, circuit durations, and the degree to which the network load matches the network topology. The simulations also illuminate some important characteristics of our algorithm. For example, we characterize the effect of the implicit routing effects of the admission control part of our algorithm.

1.8 Organization

The thesis is organized as follows. Our model for on-line algorithms and their complexity measure, the competitive ratio, are presented in Chapter 2. Chapter 3 presents our lower bounds on the competitive ratio of admission control and routing algorithms for various topologies. Some admission control and routing algorithms that have optimal or near optimal competitive ratio are the focus of Chapter 4. Chapter 5 describes our practical admission control and routing algorithm for general topology networks. The algorithm is evaluated using the simulations presented in Chapter 6. Our competitive algorithms for multicast communication are presented in Chapter 7. Finally, Chapter 8 offers some concluding remarks as well as suggestions for future research.

1.9 Authorship

This thesis is based on several papers, each of which is co-authored with several people. Chapters 3 and 4 are based on [AGLR94]. The work in that paper was done by Yuval Rabani and myself with consultation from Tom Leighton and Baruch Awerbuch. Chapters 5 and 6 are based on [GKPR95b]. I am the primary contributor to the algorithm of Chapter 5. The simulations of Chapter 6 are joint work of Anil Kamath and myself. Serge Plotkin and K.G. Ramakrishman provided consultation to the project. The work in Chapter 7 is based on [AAG94]. This paper is join work involving Baruch Awerbuch, Yossi Azar, and myself.

CHAPTER 2

The Model

This chapter introduces the formal framework for on-line algorithms. Specifically, we define on-line algorithms and the competitive ratio, a performance measure for on-line algorithms. The chapter also provides a formal definition of the admission control and routing problem in the framework of on-line algorithms.

Section 2.1 defines deterministic on-line algorithms. This definition is extended to randomized on-line algorithms in Section 2.2. Section 2.3 defines the admission control and routing problem. The competitive ratio of an on-line algorithm is defined in Section 2.4. Section 2.5 extends the notion of a competitive ratio to randomized on-line algorithms. Finally, Section 2.6 introduces some techniques for proving lower bounds for randomized competitive ratios.

2.1 Deterministic On-line Algorithms

An *on-line* algorithm provides responses to inputs (requests) as they arrive. In determining the response, the algorithm cannot take future requests into account[1].

Definition 2.1.1 (on-line algorithm) *Let Q be a set of* requests *and R be a finite set of* responses*. An* on-line algorithm *A for Q and R is a sequence of functions*[2] *A_n from Q^n to R*

[1] The definitions in this section borrow heavily from the approach used in [BDBK$^+$90].

[2] We note that functions can "solve" Turing undecidable problems. However, the problems we consider are all Turing decidable, though, in general, NP-complete. The on-line algorithms that we propose all have polynomial Turing time complexity.

for $n \in \mathbb{N}$, where $\mathbb{N}$ is the natural numbers.

A *request sequence* $\sigma = \sigma_0\sigma_1 \ldots \sigma_{|\sigma|-1}$ is a finite sequence of elements from Q. Let $A(\sigma)$ be the sequence of responses produced by on-line algorithm A on request sequence σ. Formally, $A(\sigma) = \rho = \rho_0\rho_1 \ldots \rho_{|\sigma|-1}$ where $A_i(\sigma_0\sigma_1 \ldots \sigma_{i-1}) = \rho_{i-1}$ for all $i \geq 1$. We call ρ a *response sequence*. The fact that ρ_{i-1} is based only on the prefix $\sigma_0\sigma_1 \ldots \sigma_{i-1}$ formalizes the fact that the on-line algorithm does not know anything about future requests.

Definition 2.1.2 (off-line algorithm) *Let Q be a set of* requests *and R be a finite set of* responses. *An* off-line algorithm *A for Q and R is a sequence of functions A_n from Q^n to R^n for $n \in \mathbb{N}$.*

For a request sequence $\sigma = \sigma_0\sigma_1 \ldots \sigma_{|\sigma|-1}$ and off-line algorithm A define $A(\sigma) = A_{|\sigma|}(\sigma)$.

Next we define the notion of a *problem*. A problem characterizes both the types of requests that will be generated as well as the allowable responses to those requests.

Definition 2.1.3 (problem) *A problem $\mathcal{P}$ is a triple (Q, R, S) consisting of a request set Q, a response set R, and a set S of request sequence, response sequence pairs (σ, ρ) such that $|\sigma| = |\rho|$. Denote by dom$(\mathcal{P})$ the set $\{\sigma \mid \exists \rho$ such that $(\sigma, \rho) \in S\}$.*

For some types of problems it will be useful to add some probabilistic restrictions on the request sequences. Our probabilistic restrictions require that the request sequences be chosen based on some set of allowable distributions over request sequences. We define the notion of a *probabilistic problem.*

Definition 2.1.4 (probabilistic problem) *A probabilistic problem $\mathcal{P}'$ is a tuple $(\mathcal{P}, \mathcal{D})$ consisting of a problem $\mathcal{P}$ and a set of distributions $\mathcal{D}$ over dom$(\mathcal{P})$. Define dom$(\mathcal{P}')$ = dom$(\mathcal{P})$.*

Next we define what it means for an on-line (off-line) algorithm to solve a problem.

Definition 2.1.5 (solves) *Consider problem $\mathcal{P} = (Q, R, S)$. An on-line (off-line) algorithm A for Q and R* solves *problem $\mathcal{P}$ if for all $\sigma \in$ dom$(\mathcal{P})$, $(\sigma, A(\sigma)) \in S$.*

We denote A solves $\mathcal{P}$ by $A \sqsubseteq \mathcal{P}$.

If $\mathcal{P}' = (\mathcal{P}, \mathcal{D})$ is a probabilistic problem then on-line (off-line) algorithm A solves $\mathcal{P}'$ if A solves $\mathcal{P}$. For probabilistic problems we also define a weaker version of the solution relationship.

Definition 2.1.6 (solves with probability at least q) *Let $\mathcal{P}' = (\mathcal{P}, \mathcal{D})$ be a probabilistic problem. Let $\mathcal{P} = (Q, R, S)$. An on-line (off-line) algorithm A for Q and R* solves *problem $\mathcal{P}'$ with probability at least q if for all $D \in \mathcal{D}$,*

$$Pr_D\left[\{\sigma \mid (\sigma, A(\sigma)) \in S\}\right] \geq q.$$

We denote A solves $\mathcal{P}'$ with probability at least q by $A \sqsubseteq_p \mathcal{P}'$.

Finally, we consider the *performance* of an on-line (off-line) algorithm. The performance of an on-line (off-line) algorithm is given by a function P from request and response sequences to $\Re^{>0}$, the positive real numbers.

Definition 2.1.7 (performance function) *Let $\mathcal{P} = (Q, R, S)$ be a problem. A* performance function *for problem $\mathcal{P}$ is a function from $\cup_{n \in \mathbb{N}}(Q^n \times R^n)$ to $\Re^{>0}$.*

The performance of on-line (off-line) algorithm A on request sequence σ as measured by P is $P(\sigma, A(\sigma))$. In this thesis, the goal is to achieve the highest possible value for $P(\sigma, A(\sigma))$. (Some literature on on-line algorithms measures the "cost" of a response sequence. In that case, the goal is to find on-line algorithms that yield the lowest cost response sequences.)

2.2 Randomized On-line Algorithms

Randomization is a powerful algorithmic tool. For many problems randomized algorithms can provide much more efficient solutions than are possible with deterministic algorithms [Rab63, Rab76]. Randomization can be useful in the context of on-line algorithms [RS89, BDBK$^+$90]. In particular, randomization can often be used to improve the achievable performance for certain problems instead of using the probabilistic restrictions on request sequences given for probabilistic problems [Yao77]. A *randomized* on-line algorithm A_r is simply a distribution D over deterministic algorithms.

Definition 2.2.1 (randomized on-line algorithm) *Let Q be a set of* requests *and R be a finite set of* responses. *A* randomized *on-line algorithm $A_r = (\mathcal{A}, D)$ for Q and R is a pair consisting of a set $\mathcal{A}$ of on-line algorithms, each for Q and R, and a distribution D over $\mathcal{A}$.*

A randomized on-line algorithm $A_r = (\mathcal{A}, D)$ solves a problem $\mathcal{P}$ if all on-line algorithms in $\mathcal{A}$ solve $\mathcal{P}$:

Definition 2.2.2 (solves) *Consider problem $\mathcal{P} = (Q, R, S)$. A randomized on-line algorithm $A_r = (\mathcal{A}, D)$ for Q and R* solves *problem $\mathcal{P}$ if for all $A \in \mathcal{A}$, $A \sqsubseteq \mathcal{P}$.*

We denote A_r solves $\mathcal{P}$ by $A_r \sqsubseteq \mathcal{P}$.

Finally, we consider the *performance* of a randomized on-line algorithm. The performance measure for randomized on-line algorithms is the same as the performance measure for deterministic on-line algorithms, except that we take the expectation. Consider a performance function P. The performance of randomized on-line algorithm $A_r = (\mathcal{A}, D)$ on request sequence σ is $E_D[P(\sigma, A(\sigma))]$, where E_D is the expectation over the distribution D.

2.3 Admission Control and Routing Problems

An admission control and routing problem requires an on-line algorithm. In particular, an algorithm for admission control and routing must decide whether to accept or reject a virtual circuit request without knowledge of future requests. In this thesis, we consider a variety of admission control and routing problems. In particular, we consider unicast communication and multicast communication. Furthermore, we explore the problem on various special topologies. Consider the following formal definition of the unicast admission control and routing problem for a set of graphs $\mathcal{G}$.

Definition 2.3.1 (admission control and routing for a set of graphs $\mathcal{G}$) *Let $\mathcal{G}$ be a set of graphs ranging over a node alphabet $\mathcal{V}$. If $G \in \mathcal{G}$, we describe $G = (V, E)$ by a set V of nodes and a set E of undirected links between the nodes. Now define*

$$\begin{aligned} Q_1 &= \{(s, d, r) \mid s, d \in \mathcal{V}, r \in \Re^{>0}\}, \\ Q_2 &= \{((V, E), b) \mid (V, E) \in \mathcal{G} \text{ and } b : E \rightarrow \Re^{\geq 0}\}. \end{aligned}$$

If $\sigma_i \in Q_1$ and $\sigma_i = (s_i, d_i, r_i)$ then $s(\sigma_i) = s_i$, $d(\sigma_i) = d_i$ and $r(\sigma_i) = r_i$. Let (σ, ρ) be a request sequence, response sequence pair such that $\sigma_i \in Q_1$ for all $i \in [1, |\sigma|]$. Then, for all $j \in [1, |\sigma|]$,

$u_j(e)$ is defined to be $\frac{1}{b(e)} \sum_{1 \le i < j \,|\, \rho_i \ne \perp, e \in \rho_i} r(\sigma_i)$. Now define

$$
\begin{aligned}
Q &= Q_1 \cup Q_2, \\
R &= \{q \mid q \textit{ is a simple path over } \mathcal{V}\} \cup \{\perp\}, \\
S &= \{(\sigma, \rho) \mid \;\; 1.\, \sigma_0 \in Q_2, \textit{ and } \rho_0 = \perp, \\
&\qquad\qquad\quad 2.\, \sigma_i \in Q_1 \textit{ for all } i \in [1, |\sigma|), \\
&\qquad\qquad\quad 3.\, \textit{if } \sigma_0 = ((V,E),b) \textit{ then } s(\sigma_i), d(\sigma_i) \in V \textit{ for all } i \in [1, |\sigma|), \\
&\qquad\qquad\quad 4.\, \textit{if } \sigma_0 = (G,b) \textit{ then for all } i \in [1, |\sigma|), \textit{ if } \rho_i \ne \perp \textit{ then } \rho_i \textit{ is a path} \\
&\qquad\qquad\qquad \textit{in } G \textit{ with endpoints } s(\sigma_i), d(\sigma_i), \\
&\qquad\qquad\quad 5.\, \textit{if } \sigma_0 = ((V,E),b) \textit{ then for all } e \in E, u_{|\sigma|}(e) \le 1\}.
\end{aligned}
$$

In Definition 2.3.1 the tuple (s, d, r) is a *virtual circuit request* where s represents the source, d represents the destination, and r the bandwidth. The tuple (G, b) tells the algorithm about the network topology, given by G, and the capacity associated with each link, given by b. The symbol $\perp$ represents the response to the request (G, b) and also the response to a virtual circuit request that is rejected. By returning a path, the algorithm accepts the request. There are five conditions listed for S. The first condition states that the first request consists of the network topology and the capacity information for the network and gets the response $\perp$. The second condition states that each subsequent request is a virtual circuit request. The third condition ensures that the endpoints of the requested virtual circuits are actually nodes in the network. The fourth condition ensure that the responses use valid paths. The final condition enforces the capacity constraint. In particular, $u_i(e)$ represents the percent of the capacity of link e that has been used by the requests up to but not including request σ_i. We call $u_i(e)$ the *utilization* of link e just before request σ_i is handled.

Definition 2.3.2 (greedy admission control and routing for a set of graphs $\mathcal{G}$) *Let $\mathcal{G}$ be a set of undirected graphs. The* greedy admission control and routing problem for *$\mathcal{G}$ is the same as admission control and routing problem for $\mathcal{G}$ with the following additional condition for S. If $(\sigma, \rho) \in S$, then, if $\sigma_0 = (G, b)$ and $\rho_i = \perp$, there exists no path p from $s(\sigma_i)$ to $d(\sigma_i)$ in G such that $u_{i-1}(e) + r(\sigma_i)/b(e) \le 1$ for all links e on path p.*

The additional condition for the greedy admission control and routing problem ensures that a request will always be accepted if there exists a path with sufficient capacity.

The thesis considers several cases of Definition 2.3.1 and Definition 2.3.2. For the (greedy) admission control and routing problem for general topology networks, the set of graphs $\mathcal{G}$ in Definition 2.3.1 (Definition 2.3.2) is the set of all graphs ranging over the node alphabet $\mathcal{V}$. We also consider restrictions to specific topologies. For the admission control and routing problem for trees the set of graphs $\mathcal{G}$ in Definition 2.3.1 is the set of all trees ranging over the node alphabet $\mathcal{V}$. We also consider bandwidth restrictions. The admission control and routing problem for unit capacity trees is the same as the admission control and routing problem for trees except that the capacity of each link (given by the function b) must be 1, and the bandwidth associated with each request (the third term in a request tuple) must be 1. The definition for the multicast admission control and routing problem is also similar to Definition 2.3.1. Roughly, the difference is that the requests now consist of a set of nodes that wish to be members of the multicast group, and the response is now a tree that spans the members. Rather than listing all the specific problem definitions here, we give the specific problem definitions in the chapters that present the algorithms or lower bounds for the problems. The definitions will, in general, just mention how they differ from Definition 2.3.1.

Definition 2.3.3 (admission control and routing algorithm) *An* admission control and routing algorithm *is an on-line algorithm that solves the admission control and routing problem.*

We consider two types of admission control and routing algorithms: greedy and non-greedy.

Definition 2.3.4 (greedy admission control and routing algorithm) *An algorithm A is a* greedy admission control and routing algorithm *if it is an on-line algorithm that solves the admission control and routing problem and has the following property. Consider any request sequence σ and response sequence $A(\sigma) = \rho$. Then, for all $i \in [1, |\sigma|)$, if $\sigma_0 = (G, b)$ and $\rho_i = \perp$, then there exists no path p from $s(\sigma_i)$ to $d(\sigma_i)$ in G such that $u_{i-1}(e) + r(\sigma_i)/b(e) \leq 1$ for all links e on path p.*

Lemma 2.3.5 *An admission control and routing algorithm A solves the greedy admission control and routing problem iff A is greedy.*

Proof. The lemma follows immediately from Definition 2.3.2 and Definition 2.3.4. ∎

Definition 2.3.6 (non-greedy admission control and routing algorithm) *A* non-greedy admission control and routing algorithm *is an admission control and routing algorithm that is not greedy.*

This thesis focuses on two performance functions. One performance function determines the number of accepted requests in the response sequence. The other performance function measures the total amount of accepted bandwidth.

Definition 2.3.7 (number of accepted requests) *Let $\mathcal{P} = (Q, R, S)$ be the admission control and routing problem for general topology networks. Consider $(\sigma, \rho) \in \cup_n (Q^n \times R^n)$ for $n \in \mathbb{N}$. Then the number of accepted requests in ρ, $P_r(\sigma, \rho)$, is given by*

$$P_r(\sigma,\rho) = \begin{cases} |\{\rho_i \mid \rho_i \neq \perp, i \in [0, |\sigma|)\}| & \textit{if } (\sigma,\rho) \in S \textit{ and } |\{\rho_i \mid \rho_i \neq \perp, i \in [0,|\sigma|)\}| \neq 0 \\ \epsilon & \textit{otherwise} \end{cases}$$

for some $0 < \epsilon < 1$.

Definition 2.3.8 (amount of accepted bandwidth) *Let $\mathcal{P} = (Q, R, S)$ be the admission control and routing problem for general topology networks. Consider $(\sigma, \rho) \in \cup_n (Q^n \times R^n)$ for $n \in \mathbb{N}$. Then the amount of accepted bandwidth in ρ, $P_b(\sigma, \rho)$, is given by*

$$P_b(\sigma,\rho) = \begin{cases} \sum_{1 \leq i < |\sigma| \,|\, \rho_i \neq \perp} r(\sigma_i) & \textit{if } (\sigma,\rho) \in S \textit{ and } \sum_{1 \leq i < |\sigma| \,|\, \rho_i \neq \perp} r(\sigma_i) \neq 0 \\ \epsilon & \textit{otherwise} \end{cases}$$

for some $0 < \epsilon < 1$.

The ϵ ensures that the performance function returns a positive value even if no requests are accepted.

2.4 Deterministic Competitive Ratio

Competitive analysis [ST85] provides a way of analyzing the performance of on-line algorithms. Rather than using the traditional absolute measure of performance (e.g., how many requests does the on-line algorithm accept) it uses a relative measure of performance (e.g., how many requests does the on-line algorithm accept relative to the maximum number that could have been accepted). The relative performance measure is called the *competitive ratio.* Informally, the competitive ratio of an on-line algorithm A is the maximum over all request sequences of the ratio of the maximum possible performance for that request sequence to the performance of A for the same request sequence. Thus, competitive analysis provides a worst-case relative performance measure. An on-line algorithm with a low competitive ratio is one that exhibits close to the maximum possible performance on all request sequences.

Consider a problem $\mathcal{P} = (Q, R, S)$, a performance function P, and a request sequence $\sigma \in dom(\mathcal{P})$. Define the optimal performance, $P_o(\sigma)$, for request sequence σ as follows: $P_o(\sigma) = sup_{(\sigma,\rho)\in S}\{P(\sigma, \rho)\}$.

Definition 2.4.1 (competitive ratio) *Let $\mathcal{P}$ be a problem, P a performance function, and A an on-line algorithm that solves $\mathcal{P}$.*

The competitive ratio $\mathcal{C}_{\mathcal{P},P}(A)$ *of algorithm A with respect to problem $\mathcal{P}$ and performance function P is*

$$\mathcal{C}_{\mathcal{P},P}(A) = sup_{\sigma\in dom(\mathcal{P})}\left\{\frac{P_o(\sigma)}{P(\sigma, A(\sigma))}\right\}.$$

The competitive analysis literature generally refers to an *optimal off-line algorithm* that achieves the maximum possible performance, $P_o(\sigma)$, for every request sequence σ. An optimal off-line algorithm for problem $\mathcal{P}$ and performance measure P is an off-line algorithm A that solves $\mathcal{P}$ and for which $P(\sigma, A(\sigma)) = P_o(\sigma)$ for all $\sigma \in dom(\mathcal{P})$. Such an algorithm exists:

Lemma 2.4.2 *Consider a problem $\mathcal{P} = (Q, R, S)$ and a performance function P. Then there exists an off-line algorithm A that solves $\mathcal{P}$ and for which $P(\sigma, A(\sigma)) = P_o(\sigma)$ for all $\sigma \in dom(\mathcal{P})$.*

Proof. The off-line algorithm A is a sequence of functions $A_n : Q^n \to R^n$ for $n \in \mathbb{N}$, such that, if $A_n(\sigma) = \rho$, then $P(\sigma, \rho) = sup_{(\sigma,\rho')\in S}\{P(\sigma, \rho')\}$. This function exists since the result alphabet, R, is finite and the result sequences are finite. ∎

For each request sequence the competitive ratio compares the on-line algorithm to the optimal performance for that sequence. Thus, one way to view the optimal off-line algorithm is as an algorithm that sees the request sequence in advance and then behaves optimally for that request sequence.

Section 2.1 mentions that, for some problems, it will be useful to add probabilistic restrictions. A probabilistic problem requires that the request sequences be chosen based on some set of allowable distributions over request sequences. We now define the following weaker form of competitive ratio for probabilistic problems. This new competitive ratio is determined by the worst case distribution over request sequences, rather than the worst case request sequence.

Definition 2.4.3 (competitive ratio with probability q**)** *Let $\mathcal{P}' = (\mathcal{P}, \mathcal{D})$ be a probabilistic problem, P a performance function, and A an on-line algorithm that solves $\mathcal{P}'$.*

Algorithm A achieves competitive ratio C with probability q on probabilistic problem $\mathcal{P}'$ and performance function P if for all $D \in \mathcal{D}$

$$Pr_D\left[\left\{\sigma | \frac{P_o(\sigma)}{P(\sigma, A(\sigma))} \leq C\right\}\right] \geq q.$$

Generally, we will be interested in $q = 1 - O(1/m)$, where m is a measure of the size of the problem. Achieving such a high q may require length restrictions on the request sequence. Roughly, the reason is that bad requests, i.e., ones that might cause $\frac{P_o(\sigma)}{P(\sigma, A(\sigma))} \leq C$ no longer to be true, will eventually happen if enough requests arrive.

A property of the competitive ratio is the fact that lower bounds on the competitive ratio are preserved by relaxing the problem restrictions. In particular, the fact that the competitive ratio is based on the worst case request sequence, implies the following lemma.

Lemma 2.4.4 *Let $\mathcal{P}' = (Q', R', S')$ be a problem and P be a performance function for $\mathcal{P}'$. Let $\mathcal{P} = (Q, R, S)$ be a problem such that $Q \subseteq Q'$, $R \subseteq R'$, and $S \subseteq S'$.*

If $\mathcal{C}_{\mathcal{P},P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}$, then $\mathcal{C}_{\mathcal{P}',P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'$.

Proof. For request sequence $\sigma \in dom(\mathcal{P})$, let $P_o'(\sigma)$ and $P_o(\sigma)$ denote the performance of the optimal off-line algorithm for problems $\mathcal{P}'$ and $\mathcal{P}$ respectively. Specifically, $P_o'(\sigma) = sup_{(\sigma,\rho)\in S'}\{P(\sigma, \rho)\}$ and $P_o(\sigma) = sup_{(\sigma,\rho)\in S}\{P(\sigma, \rho)\}$. Since $S \subseteq S'$, we conclude that $P_o'(\sigma) \geq P_o(\sigma)$ for all $\sigma \in dom(\mathcal{P})$.

By way of contradiction, assume that there exists an algorithm A' that solves $\mathcal{P}'$ and has $\mathcal{C}_{\mathcal{P}',P}(A') < K$. As a consequence,

$$\frac{P_o'(\sigma)}{P(\sigma, A'(\sigma))} < K \text{ for all } \sigma \in dom(\mathcal{P}'). \tag{2.1}$$

Let A be an algorithm that solves $\mathcal{P}$ and "behaves" like A'. Specifically, A is the algorithm such that $A(\sigma) = A'(\sigma)$ for all $\sigma \in dom(\mathcal{P})$. By construction, $P(\sigma, A(\sigma)) = P(\sigma, A'(\sigma))$ for all $\sigma \in dom(\mathcal{P})$. Combining this with Equation 2.1 and the fact that $P'_o(\sigma) \geq P_o(\sigma)$ for all $\sigma \in dom(\mathcal{P})$ we have

$$\begin{aligned}
\frac{P_o(\sigma)}{P(\sigma, A(\sigma))} &\leq \frac{P'_o(\sigma)}{P(\sigma, A(\sigma))} \\
&= \frac{P'_o(\sigma)}{P(\sigma, A'(\sigma))} \\
&< K \qquad\qquad \text{for all } \sigma \in dom(\mathcal{P}).
\end{aligned}$$

Thus, $P_o(\sigma)/P(\sigma, A(\sigma)) < K$ for all $\sigma \in dom(\mathcal{P})$. However, this contradicts the fact that $\mathcal{C}_{\mathcal{P},P}(A) \geq K$ for all deterministic on-line algorithms that solve $\mathcal{P}$. Thus, it must be the case that $\mathcal{C}_{\mathcal{P},P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'$. ■

2.5 Randomized Competitive Ratio

The degree to which randomization can improve the achievable competitive ratio for a particular problem depends on the definition of the competitive ratio for a randomized on-line algorithm. This section considers two definitions. The first definition does allow randomization to improve the achievable competitive ratio. The second definition does not.

The first competitive ratio definition for randomized on-line algorithms is called the *oblivious* randomized competitive ratio. The intuition behind this definition is that the worst case request sequence does not take the random choices of the on-line algorithm into account. In other words, the worst case sequence is picked *before* the on-line algorithm makes its random choices.

Definition 2.5.1 (oblivious competitive ratio) *Let $\mathcal{P}$ be a problem, P a performance function, and $A_r = (\mathcal{A}, D)$ a randomized on-line algorithm that solves $\mathcal{P}$.*

The competitive ratio $\mathcal{C}^b_{\mathcal{P},P}(A_r)$ *of randomized on-line algorithm A_r with respect to problem $\mathcal{P}$ and performance function P is*

$$\mathcal{C}^b_{\mathcal{P},P}(A_r)) = sup_{\sigma \in dom(\mathcal{P})} \left\{ \frac{P_o(\sigma)}{E_D[P(\sigma, A(\sigma))]} \right\}.$$

In Lemma 2.4.4 we show that lower bounds on the competitive ratio are preserved by relaxing the problem restrictions. The same result also holds for the oblivious competitive ratio.

Lemma 2.5.2 *Let $\mathcal{P}' = (Q', R', S')$ be a problem and P be a performance function for $\mathcal{P}'$. Let $\mathcal{P} = (Q, R, S)$ be a problem such that $Q \subseteq Q'$, $R \subseteq R'$, and $S \subseteq S'$.*

If $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}$, then $\mathcal{C}^b_{\mathcal{P}',P}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'$.

Proof. The proof of this lemma is similar to the proof of Lemma 2.4.4. The only difference is in the construction of the algorithm A from A'. This construction needs to be generalized to randomized algorithms in the obvious way. ∎

A common objection to the oblivious randomized competitive ratio is that the outside world (i.e., the person/machine generating the requests) is not able to react to the randomized choices made by the on-line algorithm. In many situation the ability to react to the randomized choices would provide a more realistic performance model. For example, the likelihood that a particular customer makes a virtual circuit request in a few seconds will depend on whether the customer's current request was accepted. A performance measure called the *adaptive* competitive ratio attempts to address this criticism. Randomization provides no extra power in the context of the adaptive competitive ratio. Therefore, we will focus on the oblivious competitive ratio in this thesis. For completeness, we present the formal definition of the adaptive competitive ratio.

Informally, the adaptive competitive ratio determines the worst case request sequence after the randomized on-line algorithm makes its random choices (by choosing a specific deterministic on-line algorithm). Thus, it is possible to adjust the request sequence based on the randomized choices of the on-line algorithm.

Definition 2.5.3 (adaptive competitive ratio) *Let $\mathcal{P}$ be a problem, P a performance function, and $A_r = (\mathcal{A}, D)$ a randomized on-line algorithm that solves $\mathcal{P}$.*

The competitive ratio $\mathcal{C}^a_{\mathcal{P},P}(A_r)$ *of randomized on-line algorithm A_r with respect to problem $\mathcal{P}$ and performance function P is*

$$\mathcal{C}^a_{\mathcal{P},P}(A_r) = E_D[\mathcal{C}_{\mathcal{P},P}(A)].$$

It is easy to prove the following theorem. (The proof of this theorem can be found in [BDBK+90].)

Theorem 2.5.4 *If there exists a randomized on-line algorithm $A_r = (\mathcal{A}, D)$ with adaptive competitive ratio K for some problem $\mathcal{P}$ and performance function P, then there exists a deterministic on-line algorithm for problem $\mathcal{P}$ and performance function P with competitive ratio of at most K.*

Proof sketch. By way of contradiction assume that no such deterministic on-line algorithm exists. Then, $\mathcal{C}_{\mathcal{P},P}(A) > K$ for all deterministic on-line algorithms A. As a result, $\mathcal{C}^a_{\mathcal{P},P}(A_r) = \sum_A Pr_D(A)\mathcal{C}_{\mathcal{P},P}(A) > K$. Thus, we have a contradiction to the fact that there exists a randomized on-line algorithm with adaptive competitive ratio K. ∎

2.6 Randomized Lower Bounds

A result by Yao [Yao77] considerably simplifies the construction of lower bound proofs for randomized on-line algorithms. In particular, Yao notes that the complexity of randomized algorithms is connected to the complexity of deterministic algorithms on randomized inputs. Borodin et al. [BLS92] extend Yao's theorem to competitive analysis with a theorem, which states that the lower bound on the oblivious competitive ratio for a given problem is greater than the lower bound on the competitive ratio of deterministic on-line algorithms, when the request sequences for the problem are restricted to a distribution.

Theorem 2.6.1 *Consider some problem $\mathcal{P} = (Q, R, S)$ and performance function P. Furthermore, consider any distribution D_σ over $dom(\mathcal{P})$ and any randomized on-line algorithm A_r that solves $\mathcal{P}$. Then there exists a deterministic on-line algorithm A' that solves $\mathcal{P}$ and for which*

$$\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq \frac{E_{D_\sigma}[P_o(\sigma)]}{E_{D_\sigma}[P(\sigma, A'(\sigma))]}.$$

Proof. By definition, $P_o(\sigma)/E_D[P(\sigma, A(\sigma))] \leq \mathcal{C}^b_{\mathcal{P},P}(A_r)$ for all request sequences $\sigma \in dom(\mathcal{P})$. Hence, $P_o(\sigma) \leq \mathcal{C}^b_{\mathcal{P},P}(A_r)E_D[P(\sigma, A(\sigma))]$. Taking the expectation over D_σ we get: $E_{D_\sigma}[P_o(\sigma)] \leq \mathcal{C}^b_{\mathcal{P},P}(A_r)E_{D_\sigma}[E_D[P(\sigma, A(\sigma))]]$. Thus, $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq E_{D_\sigma}[P_o(\sigma)]/E_{D_\sigma}[E_D[P(\sigma, A(\sigma))]]$. Reversing the order of the expectations in the denominator, $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq E_{D_\sigma}[P_o(\sigma)]/E_D[E_{D_\sigma}[P(\sigma, A(\sigma))]]$. Thus, there exists some algorithm A' such that $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq E_{D_\sigma}[P_o(\sigma)]/E_{D_\sigma}[P(\sigma, A'(\sigma))]$. ∎

This theorem can only be used for lower bounds on the oblivious competitive ratio, not the adaptive competitive ratio.

2.7 General Lemmas for Admission Control and Routing

This section presents some general lemmas for admission control and routing problems.

The first lemma states that lower bounds are preserved by expanding the set of graphs to which the admission control and routing problem applies.

Lemma 2.7.1 *Let $\mathcal{P}'$ and $\mathcal{P}$ be the admission control and routing problems for the set of graphs $\mathcal{G}$ and $\mathcal{G}'$ respectively. Let $\mathcal{G} \subseteq \mathcal{G}'$. Furthermore, let P be a performance function for $\mathcal{P}'$.*

If $\mathcal{C}_{\mathcal{P},P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}$, then $\mathcal{C}_{\mathcal{P}',P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'$. Similarly, if $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}$, then $\mathcal{C}^b_{\mathcal{P}',P}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'$.

Proof. The lemma follows immediately from Lemmas 2.4.4 and 2.5.2. ∎

Lower bounds that apply to a particular graph G extend to certain related graphs. Let $\mathcal{P}(\{G\})$ be the admission control and routing problem where the set of graphs $\mathcal{G}$ is $\{G\}$.

Lemma 2.7.2 *Consider the graphs G and G' such that G is a subgraph of G' and there exist no simple path $(v_0 \ldots v_n)$ in G' such that v_0 and v_n are in G, but v_i is not in G for some $0 < i < n$. Let P be the performance function of Definition 2.3.7.*

If $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G'\}),P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}(\{G'\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G'\}),P}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}(\{G'\})$.

Both statements also apply to the performance function in Definition 2.3.8.

Proof. For any request sequence $\sigma \in dom(\mathcal{P}(\{G\}))$, let $\sigma' \in dom(\mathcal{P}(\{G'\}))$ be the request sequence that is the same as σ but specifies G' as the graph in the first request. Specifically, $\sigma'_i = \sigma_i$ for $0 < i < |\sigma|$, and, if $\sigma_0 = (G, b)$, then $\sigma'_0 = (G', b)$. Let $M = \{\sigma' | \sigma \in dom(\mathcal{P}(\{G\}))\}$. Clearly, $M \subseteq dom(\mathcal{P}(\{G'\}))$. Let P be the performance function in Definition 2.3.7.

Let $\mathcal{P}(\{G\}) = (Q, R, S)$ and let $\mathcal{P}(\{G'\}) = (Q', R', S')$. If $(\sigma, \rho) \in S$, then $(\sigma', \rho) \in S'$. By Definition 2.3.7, $P(\sigma, \rho) = P(\sigma', \rho)$ for all $(\sigma, \rho) \in S$. As a consequence,

$$P_o(\sigma) \leq P_o(\sigma') \text{ for all } \sigma \in dom(\mathcal{P}(\{G\})). \tag{2.2}$$

By way of contradiction, assume that there exists an algorithm A' that solves $\mathcal{P}(\{G'\})$ and has $\mathcal{C}_{\mathcal{P}(\{G'\}),P}(A') < K$. As a consequence,

$$\frac{P_o(\sigma')}{P(\sigma', A'(\sigma'))} < K \text{ for all } \sigma' \in M. \tag{2.3}$$

Since $\mathcal{P}(\{G'\})$ is an admission control and routing problem, any result sequence produced by an algorithm that solves $\mathcal{P}(\{G'\})$ must include only simple paths. Thus, $A'(\sigma')$ includes only simple paths for all $\sigma' \in dom(\mathcal{P}(\{G\}'))$. Now, the definition of G and G' implies that the response sequence $A'(\sigma')$ for any $\sigma' \in M$ includes only paths that are in G. As a result, there exists an algorithm A that solves $\mathcal{P}(\{G\})$ and "behaves" like A'. Specifically, A is the algorithm such that $A(\sigma) = A'(\sigma')$ for all $\sigma \in dom(\mathcal{P}(\{G\}))$. By construction of A and the fact that $P(\sigma, \rho) = P(\sigma', \rho)$ for all $(\sigma, \rho) \in S$.

$$P(\sigma, A(\sigma)) = P(\sigma', A'(\sigma')) \text{ for all } \sigma \in dom(\mathcal{P}(\{G\})). \tag{2.4}$$

Now we combine Equations 2.3, 2.4 and 2.2 and the fact that $\sigma' \in M$ when $\sigma \in dom(\mathcal{P}(\{G\}))$ to conclude for all $\sigma \in dom(\mathcal{P}(\{G\}))$ that

$$\begin{aligned} \frac{P_o(\sigma)}{P(\sigma, A(\sigma))} &\leq \frac{P_o(\sigma')}{P(\sigma, A(\sigma))} \\ &= \frac{P_o(\sigma')}{P(\sigma', A'(\sigma'))} \\ &< K. \end{aligned}$$

Thus, $P_o(\sigma)/P(\sigma, A(\sigma)) < K$ for all $\sigma \in dom(\mathcal{P}(\{G\}))$. However, this contradicts the fact that $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms that solve $\mathcal{P}(\{G\})$. Thus, it must be the case that $\mathcal{C}_{\mathcal{P}(\{G'\}),P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}(\{G'\})$.

The proof showing the lemma for the oblivious competitive ratio is similar. The only difference is in the construction of the algorithm A from A'. This construction needs to be generalized to randomized algorithms in the obvious way.

The proof for the performance function in Definition 2.3.8 is exactly the same. ∎

CHAPTER 3

Lower Bounds

3.1 Introduction

This chapter provides lower bounds on the competitive ratio of admission control and routing algorithms for various topologies. The lower bounds in this chapter suggest that non-greedy admission control and routing algorithms can have better competitive ratios than greedy admission control and routing algorithms.

Consider the admission control and routing problem on a topology described by graph G. Section 3.2 proves a theorem which provides sufficient conditions for a lower bound on the oblivious competitive ratio of any greedy randomized algorithm for the admission control and routing problem on G and a lower bound on the oblivious competitive ratio of *any* randomized algorithm for the admission control and routing problem on G. The two lower bounds implied by the theorem have an exponential separation. In particular, if the theorem implies an $\Omega(Z)$ lower bound on the oblivious competitive ratio of any randomized algorithm, then it implies an $\Omega(2^Z)$ the lower bound on the oblivious competitive ratio of any greedy randomized algorithm. Thus, for graphs that meet the conditions of the theorem, non-greedy admission control strategies can potentially lead to significant performance improvements. In this chapter, we use the theorem of Section 3.2 to prove lower bounds for lines, trees, meshes, trees of meshes, fat-trees, and hypercubes. For each of these topologies, there is an exponential separation between the lower bound we present for the oblivious competitive ratio of any randomized algorithm and the lower bound we present for the oblivious competitive ratio of any greedy randomized algorithm. The advantages of non-greedy admission control are further demonstrated by the fact that, for several of the topologies, the non-greedy algorithms presented in Chapter 4 beat the greedy lower bounds of this chapter and meet the lower bounds on any algorithm presented in this chapter.

Section 3.3 gives an $\Omega(\log d)$ lower bound on the oblivious competitive ratio of any algorithm for lines of length d and trees of diameter d. For greedy algorithms we provide an $\Omega(d)$ lower bound on the oblivious competitive ratio. Furthermore, we show an $\Omega(d)$ the lower bound on the deterministic competitive ratio. Section 3.4 considers meshes. We prove an $\Omega(\log n)$ lower bound on the oblivious competitive ratio of any algorithm for $n \times n$ meshes. For greedy algorithms we provide an $\Omega(n)$ lower bound on the oblivious competitive ratio. In Section 3.5 we give an $\Omega(\log\log n)$ lower bound on the oblivious competitive ratio of any algorithm for n^2-leaf trees of meshes and n^2-leaf fat-trees. For greedy algorithms we provide an $\Omega(\log n)$ lower bound on the oblivious competitive ratio. Based on the $O(\log\log n)$ competitive ratio of our non-greedy deterministic algorithm for fat-trees (cf. Chapter 4) we can conclude that the lower bound on the deterministic competitive ratio for the n^2-leaf fat-tree is $\Omega(\log\log n)$. For the $\log n$ dimensional hypercube, Section 3.6 gives an $\Omega(\log\log n)$ lower bound on the oblivious competitive ratio for any algorithm. For greedy algorithms we provide an $\Omega(\log n)$ lower bound on the oblivious competitive ratio. The table in Figure 3-1 summarizes the lower bounds presented in this chapter. (The lower bound shown in Figure 3-1 on the competitive ratio of any deterministic algorithm for the $n \times n$ mesh is presented in [BFKR93].)

	greedy randomized	any deterministic	any randomized
length d line	$\Omega(d)$	$\Omega(d)$	$\Omega(\log d)$
diameter d tree	$\Omega(d)$	$\Omega(d)$	$\Omega(\log d)$
$n \times n$ mesh	$\Omega(n)$	$\Omega(\sqrt{n})$	$\Omega(\log n)$
n^2-leaf tree of meshes	$\Omega(\log n)$		$\Omega(\log\log n)$
n^2-leaf fat-tree	$\Omega(\log n)$	$\Omega(\log\log n)$	$\Omega(\log\log n)$
n dimensional hypercube	$\Omega(\log n)$		$\Omega(\log\log n)$

Figure 3-1: Summary of known competitive ratio lower bounds for greedy randomized, any deterministic, and any randomized admission control and routing algorithms on various topologies.

Section 3.7 considers *hierarchical backbone networks* (cf. Definition 3.7.1). A hierarchical backbone network can be decomposed into many low diameter regions (access networks) and an arbitrary region (backbone network) that connects the access networks. Hierarchical backbone networks can be used to model several important networks including the telephone network, the Internet, and fat-trees. Section 3.7 shows that a lower bound on the competitive ratio for the backbone network or for any of the access networks implies the same lower bound on the competitive ratio for the entire hierarchical backbone network.

Section 3.8 presents further evidence for the importance of non-greedy admission control in the context of competitive analysis. The section provides an $\Omega(n)$ lower bound on the competitive ratio of any deterministic *greedy* admission control and routing algorithm for a n node general topology network. Such a lower bound is trivial if the optimal off-line algorithm may use non-greedy admission control. However, Section 3.8 shows that the linear lower bound still holds when the optimal off-line algorithm must use greedy admission control. This lower bound, together with the $O(\log n)$ competitive non-greedy algorithm for general topologies (see Chapter 4), underscores the importance of using non-greedy admission control for general topology networks.

3.2 A Theorem for Greedy and Randomized Lower Bounds

This section provides tools for proving oblivious randomized lower bounds on all algorithms and on greedy algorithms. Given a topology and a set of sequences of virtual circuit requests for that topology that meet certain conditions (cf. Theorem 3.2.4) this section provides a theorem that implies a lower bound on the oblivious competitive ratio of any algorithm and a lower bound on the oblivious competitive ratio of any greedy algorithm. The lower bound provided for the oblivious competitive ratio of any greedy algorithm is exponentially greater than the lower bound provided for the oblivious competitive ratio of any algorithm. Thus, our theorem helps identify topologies where non-greedy admission control and routing algorithms may outperform greedy admission control and routing algorithms. The lower bound on any algorithm is proven in Lemma 3.2.1. The lower bound on any greedy algorithm is proven in Lemma 3.2.3. Theorem 3.2.4 combines the results of Lemma 3.2.1 and Lemma 3.2.3.

The proof for the oblivious randomized lower bound on any algorithm constructs a probability distribution D_σ over request sequences using the provided sequences of virtual circuit requests. Using the probability distribution, we provide a lower bound for $E_{D_\sigma}[P_o(\sigma)]$, the expected number of circuits accepted by an optimal off-line algorithm and an upper bound for $E_{D_\sigma}[P(\sigma, A(\sigma))]$, the expected number of requests any deterministic on-line algorithm A can accept. These bounds, combined with Theorem 2.6.1, imply the desired lower bound on the oblivious competitive ratio.

Lemma 3.2.1 *Let $\mathcal{P} = (Q, R, S)$ be the admission control and routing problem for a unit capacity graph $G = (V, E)$. Let P be the performance function that determines the number of accepted virtual circuit requests. For each $i \in [0, Z]$, let $\overline{\sigma}_i$ be a sequence of requests with the following properties:*

- $|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ *if* $i \neq 0$.
- *there exists a response sequence ρ such that $(\sigma_0\overline{\sigma}_i, \rho) \in S$ and $P(\sigma_0\overline{\sigma}_i, \rho) = |\overline{\sigma}_i|$*
- *there is a set of* $2^Z|\overline{\sigma}_0|$ critical links *in E such that, for any request $(s, d, 1)$ in $\overline{\sigma}_i$, any path from s to d uses at least 2^{Z-i} critical links.*

Then, any randomized on-line algorithm, A_r, for problem $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(Z)$.

Proof. The proof proceeds as follows. We use the sequences $\overline{\sigma}_i$ to construct a probability distribution D_σ over the request sequences in $dom(\mathcal{P})$. Then, we provide a lower bound for $E_{D_\sigma}[P_o(\sigma)]$, the expected number of requests accepted by an optimal off-line algorithm and an upper bound for $E_{D_\sigma}[P(\sigma, A(\sigma))]$, the expected number of requests any deterministic on-line algorithm A can accept. These bounds, combined with Theorem 2.6.1, imply the desired lower bound on the oblivious competitive ratio.

First, we define the probability distribution D_σ over request sequences. By definition of $\mathcal{P}$, the first request, σ_0, of every request sequence is (G, b) where $b : E \rightarrow \{1\}$. With probability 2^{-Z}, $\sigma = \sigma_0$, i.e., there are no circuit requests. All other request sequences with non-zero probability are of the form $\sigma = \sigma_0\overline{\sigma}_0 \ldots \overline{\sigma}_i$, where $Pr_{D_\sigma}[\sigma = \sigma_0\overline{\sigma}_0] = 1/2$ and $Pr_{D_\sigma}[\sigma = \sigma_0\overline{\sigma}_0 \ldots \overline{\sigma}_i] = \frac{1}{2}Pr_{D_\sigma}[\sigma = \sigma_0\overline{\sigma}_0 \ldots \overline{\sigma}_{i-1}]$, for every $i \in [1, Z]$. Thus, for $i \in [0, Z]$, the probability of the sequence $\sigma_0\overline{\sigma}_0 \ldots \overline{\sigma}_i$ is 2^{-i-1}.

Next, we provide a lower bound for $E_{D_\sigma}[P_o(\sigma)]$. Consider an off-line strategy that accepts all the requests in $\overline{\sigma}_i$ given that the request sequence is $\sigma_0\overline{\sigma}_0\ldots\overline{\sigma}_i$. The number of requests in $\overline{\sigma}_i$ is $2^i|\overline{\sigma}_0|$. Furthermore, $\sigma_0\overline{\sigma}_0\ldots\overline{\sigma}_i$ is the actual request sequence with probability 2^{-i-1}. Thus, the expected number of requests accepted by this strategy is given by

$$\sum_{i=0}^{Z} Pr_{D_\sigma}[\sigma = \sigma_0\overline{\sigma}_0\ldots\overline{\sigma}_i]2^i|\overline{\sigma}_0| = \sum_{i=0}^{Z} 2^{-i-1}2^i|\overline{\sigma}_0| \leq \frac{Z}{2}|\overline{\sigma}_0|. \tag{3.1}$$

Thus, $E_{D_\sigma}[P_o(\sigma)] \geq \frac{Z}{2}|\overline{\sigma}_0|$.

Next, we provide an upper bound on $E_{D_\sigma}[P(\sigma, A(\sigma))]$ for all deterministic on-line algorithms A. Any request accepted from sequence $\overline{\sigma}_i$ consumes at least 2^{Z-i} critical links. Therefore, if k critical links remain unused by requests accepted from $\overline{\sigma}_0\ldots\overline{\sigma}_{i-1}$, and requests from $\overline{\sigma}_i$ arrive, we can hope to accept at most $k/2^{Z-i}$ of those requests. Conditioning upon the fact that the actual request sequence σ includes $\overline{\sigma}_i$, denote by $B(i,k)$ the maximum expected number of requests accepted from $\overline{\sigma}_i\overline{\sigma}_{i+1}\ldots$, where the maximization is taken over all possible ways to accept requests from $\overline{\sigma}_0\ldots\overline{\sigma}_{i-1}$ so that at most k critical links are free. We can bound $B(i,k)$ with the following recurrence relation, where the first term in the maximum, ℓ, represents the number of requests accepted from $\overline{\sigma}_i$ and the second term, $\frac{1}{2}B(i+1, k-\ell 2^{Z-i})$, represents the maximum expected number of requests accepted from $\overline{\sigma}_{i+1}\overline{\sigma}_{i+2}\ldots$, given that $k-\ell 2^{Z-i}$ free critical links remain:

$$B(i,k) \leq \max_{\ell \leq k/2^{Z-i}} \left\{\ell + \frac{1}{2}B(i+1, k-\ell 2^{Z-i})\right\},$$

with the initial condition

$$B(Z,k) \leq k.$$

The factor of $1/2$ in front of the term $B(i+1, k-\ell 2^{Z-i})$ results from the fact that the probability that σ includes the sequence $\overline{\sigma}_{i+1}$, given that σ already includes $\overline{\sigma}_i$, is $1/2$.

Since there are $2^Z|\overline{\sigma}_0|$ critical links, $B(0, 2^Z|\overline{\sigma}_0|)$ is an upper bound on $E_{D_\sigma}[P(\sigma, A(\sigma))]$, the expected number of requests that any deterministic on-line algorithm A accepts. We prove in Claim 3.2.2 that $B(i,k) \leq k/2^{Z-i}$ for all $0 \leq i \leq Z$ and all k. Thus, $B(0, 2^Z|\overline{\sigma}_0|) \leq |\overline{\sigma}_0|$. As a consequence, $E_{D_\sigma}[P(\sigma, A(\sigma))] \leq |\overline{\sigma}_0|$ for any deterministic on-line algorithm A.

Since $E_{D_\sigma}[P_o(\sigma)] \geq \frac{Z}{2}|\overline{\sigma}_0|$ and $E_{D_\sigma}[P(\sigma, A(\sigma))] \leq |\overline{\sigma}_0|$ for any deterministic on-line algorithm A, Theorem 2.6.1 implies that $\mathcal{C}^b_{\mathcal{P},P}(A_r) \geq \Omega(Z)$ for all randomized on-line algorithms A_r. ■

The basic idea of using sequences of requests where the number of requests using some number of resources is twice the number of requests using twice the resources was used in the lower bound proofs in [AAP93]. The lower bound proofs in [AAP93] consider line networks.

Claim 3.2.2 *If, for all k,*

$$B(i,k) \leq \begin{cases} \max_{\ell \leq k/2^{Z-i}} \{\ell + \frac{1}{2}B(i+1, k-\ell 2^{Z-i})\} & \textit{for } i \in [0, Z) \\ k & \textit{for } i = Z \end{cases},$$

then $B(i,k) \leq k/2^{Z-i}$ for all $i \in [0, Z]$ and all k.

Proof. The proof is by induction on $j = Z - i$. The base case ($j = 0, i = Z$) follows immediately from the initial condition on B. Now, assume that for all k, $B(i+1, k) \leq k/2^{Z-i-1}$. We have that

$$\begin{aligned} B(i,k) &\leq \max_{\ell \leq k/2^{Z-i}} \left\{\ell + \frac{1}{2}B(i+1, k-\ell 2^{Z-i})\right\} \\ &\leq \max_{\ell \leq k/2^{Z-i}} \left\{\ell + \frac{k-\ell 2^{Z-i}}{2(2^{Z-i-1})}\right\} \\ &\leq \frac{k}{2^{Z-i}}. \end{aligned}$$

■

Having considered the oblivious competitive ratio of any randomized admission control and routing algorithm, we now provide the conditions needed for a lower bound on the oblivious competitive ratio of any *greedy* randomized admission control and routing algorithm.

Lemma 3.2.3 *Let $\mathcal{P} = (Q, R, S)$ be the admission control and routing problem for a unit capacity graph $G = (V, E)$. Let P be the performance function that determines the number of accepted virtual circuit requests. Let $\overline{\sigma}$ and $\overline{\sigma}'$ be two sequences of requests with the following properties:*

- $|\overline{\sigma}'| = 2^Z|\overline{\sigma}|$.

- *there exists a response sequence ρ such that $(\sigma_0\overline{\sigma}, \rho) \in S$ and $P(\sigma_0\overline{\sigma}, \rho) = |\overline{\sigma}|$*

- *there exists a response sequence ρ such that $(\sigma_0\overline{\sigma}', \rho) \in S$ and $P(\sigma_0\overline{\sigma}', \rho) = |\overline{\sigma}'|$*

- *for every response sequence ρ such that $(\sigma_0\overline{\sigma}\overline{\sigma}', \rho) \in S$ and $\rho_i \neq \perp$ for all $\sigma_i \in \overline{\sigma}$, it is the case that $\rho_i = \perp$ for all $\sigma_i \in \overline{\sigma}'$.*

Then, any greedy randomized on-line algorithm, A_r, for problem $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(Z)$.

Proof. By definition of $\mathcal{P}$, the first request, σ_0, of every request sequence is (G, b) where $b : E \to \{1\}$. Consider any greedy algorithm A_r and request sequence $\sigma = \sigma_0\overline{\sigma}\overline{\sigma}'$. Since A_r is greedy, it must accept the requests in $\overline{\sigma}$, and thus cannot accept any request in $\overline{\sigma}'$. Now consider an optimal off-line algorithm that rejects the requests in $\overline{\sigma}$ but accepts the $2^Z|\overline{\sigma}|$ requests in $\overline{\sigma}'$. Thus, $P_o(\sigma)/E[P(A(\sigma))] \geq \frac{2^Z|\overline{\sigma}|}{|\overline{\sigma}|} = 2^Z$. ■

Finally, we combine the results of Lemma 3.2.1 and Lemma 3.2.3.

Theorem 3.2.4 *Let $\mathcal{P} = (Q, R, S)$ be the admission control and routing problem for a unit capacity graph $G = (V, E)$. Let P be the performance function that determines the number of accepted virtual circuit requests. For each $i \in [0, Z]$, let $\overline{\sigma}_i$ be a sequence of requests with the following properties:*

- *$|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ if $i \neq 0$.*

- *there exists a response sequence ρ such that $(\sigma_0\overline{\sigma}_i, \rho) \in S$ and $P(\sigma_0\overline{\sigma}_i, \rho) = |\overline{\sigma}_i|$*

- *there is a set of $2^Z|\overline{\sigma}_0|$* critical links *in E such that, for any request $(s, d, 1)$ in $\overline{\sigma}_i$, any path from s to d uses at least 2^{Z-i} critical links.*

Then, any randomized on-line algorithm, A_r, for problem $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(Z)$. Furthermore, any greedy randomized on-line algorithm, A_r, for problem $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(2^Z)$.

Proof. The lower bound on any randomized algorithm follows immediately from Lemma 3.2.1. The lower bound on any greedy randomized algorithm follows from Lemma 3.2.3 where $\overline{\sigma} = \overline{\sigma}_0$ and $\overline{\sigma}' = \overline{\sigma}_Z$. ■

3.3 Lines and Trees

This section proves an $\Omega(\log d)$ lower bound on the oblivious competitive ratio of any admission control and routing algorithm for lines of length d and trees of diameter d with respect to the performance function that determines the number of accepted virtual circuit requests. The section also proves an $\Omega(d)$ lower bound on the oblivious competitive ratio of any greedy admission control and routing algorithm.

We use the following proof strategy. We first consider the admission control and routing problem for unit capacity lines of length d. (Recall that the unit capacity admission control and routing problem restricts each link to have a capacity (given by the function b) of 1 and restricts the bandwidth associated with each request (the third term in a request tuple) to be 1.) We prove $\Omega(\log d)$ and $\Omega(d)$ lower bounds for the admission control and routing problem for unit capacity lines of length d. By Lemma 2.7.2 this will imply the same lower bounds on the admission control and routing problem for any particular unit capacity tree of diameter d. Then, Lemma 2.5.2 extends the lower bounds to the admission control and routing problem for diameter d trees without the bandwidth and capacity restriction.

We note that [ABFR94] already prove an $\Omega(\log d)$ lower bound for the unit capacity line of length d. We present our proof since it illustrates on a simple example the proof technique that we use for our lower bound results on meshes, trees of meshes fat-trees, and hypercubes.

To prove our lower bounds for the unit capacity line of length d, we construct a set of sequences, $\overline{\sigma}_0, \overline{\sigma}_1, \ldots, \overline{\sigma}_{\log d}$, which will allow us to make use of Theorem 3.2.4. Consider a line of length d where d is a power of two. Let $(v_0, v_1 \ldots v_d)$ be the path of length d in the line. Sequence $\overline{\sigma}_0$ consists of a single circuit request of bandwidth 1 between v_0 and v_d. To construct $\overline{\sigma}_i$, divide the path between v_0 and v_d into 2^i equal length paths and request a bandwidth 1 virtual circuit between the endpoints of each of the 2^i paths. Formally:

$$\begin{aligned}
\overline{\sigma}_0 &= (v_0, v_d, 1) \\
\overline{\sigma}_1 &= (v_0, v_{d/2}, 1)(v_{d/2}, v_d, 1) \\
&\vdots \\
\overline{\sigma}_i &= (v_0, v_{d/2^i}, 1)(v_{d/2^i}, v_{2d/2^i}, 1)(v_{2d/2^i}, v_{3d/2^i}, 1)\ldots(v_{(2^i-1)d/2^i}, v_d, 1) \\
&\vdots \\
\overline{\sigma}_{\log d} &= (v_0, v_1, 1)(v_1, v_2, 1)\ldots(v_{d-1}, v_d, 1).
\end{aligned}$$

We can now prove our lower bounds on the unit capacity line of length d using Theorem 3.2.4 and the sequences constructed for the line of length d.

Lemma 3.3.1 *Let d be a power of two. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity line of length d has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log d)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(d)$.*

Proof. The sequences $\overline{\sigma}_i$ for $0 \le i \le \log d$ constructed for the line of length d satisfy the conditions of Theorem 3.2.4 for $Z = \log d$. In particular, for all $0 \le i \le \log d$, $|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ if $i \ne 0$, all request in $|\overline{\sigma}_i|$ can be accepted when no other requests are accepted from other sequences, and the are $d = 2^{\log d}$ critical links (all of the links of the line) such that a request from $\overline{\sigma}_i$ requires $d/2^i = 2^{(\log d)-i}$ critical links. ∎

We extend of our lower bounds to lines with a length that is not a power of two.

Theorem 3.3.2 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity line of length d has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log d)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(d)$.*

Proof. Let d' be the largest power of two that is less than or equal to d. Clearly, $2d' > d$. Consider a length d' line embedded in the length d line. Now, the lemma follows immediately from Lemma 2.7.2 and Lemma 3.3.1. ∎

Now consider any unit capacity diameter d tree.

Lemma 3.3.3 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on any unit capacity diameter d tree has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log d)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(d)$.*

Proof. Let G be the line of length d. Let G' be any tree of diameter d. G is a subgraph of G'. Furthermore, there exists no simple path $(v_0 \ldots v_n)$ in G' such that v_0 and v_n are in G, but v_i is not in G for some $0 < i < n$. Now the lemma follows directly from Lemma 2.7.2 and Theorem 3.3.2. ∎

Finally, we consider diameter d trees without the bandwidth and capacity restriction.

Theorem 3.3.4 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on diameter d trees has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log d)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(d)$.*

Proof. The theorem follows immediately from Lemma 2.5.2 and Lemma 3.3.3. ∎

We mention that the trivial lower bound for deterministic on-line admission control and routing algorithms for lines of length d and trees of diameter d is $\Omega(d)$.

Proposition 3.3.5 *Any deterministic on-line algorithm, A, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity line of length d has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(d)$.*

Proof. Consider any deterministic algorithm A and request sequence $\sigma = \sigma_0\overline{\sigma}_0$. If A does not accept the request in $\overline{\sigma}_0$ then consider an optimal off-line algorithm that does accept the request in $\overline{\sigma}_0$. In this case, $P_o(\sigma)/P(A(\sigma)) \geq 1/\epsilon > d$. (Recall from Definition 2.3.7 that $P(\sigma) \geq \epsilon$ for all request sequences σ.) On the other hand, if A does accept the request in $\overline{\sigma}_0$ then consider a request sequence $\sigma = \sigma_0\overline{\sigma}_0\overline{\sigma}_{\log d}$ and an optimal off-line algorithm that rejects the request in $\overline{\sigma}_0$ but accepts the d requests in $\overline{\sigma}_{\log d}$. In this case, we also have $P_o(\sigma)/P(A(\sigma)) \geq d/1 = d$.) ∎

We extend this proposition to any unit capacity diameter d tree.

Proposition 3.3.6 *Any deterministic on-line algorithm, A, that solves the admission control and routing problem, $\mathcal{P}$, on any unit capacity diameter d tree has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(d)$.*

Proof. The proposition follows directly from Lemma 2.7.2 and Proposition 3.3.5. ■

Finally, we consider diameter d trees without the bandwidth and capacity restriction.

Proposition 3.3.7 *Any deterministic on-line algorithm, A, that solves the admission control and routing problem, $\mathcal{P}$, on diameter d trees has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(d)$.*

Proof. The proposition follows directly from Lemma 2.7.1 and Proposition 3.3.6. ■

The separation between the deterministic and greedy lower bounds ($\Omega(d)$) and the oblivious randomized lower bound ($\Omega(\log d)$) on trees underscores the importance of non-greedy randomized admission control strategies. In Chapter 4 we provide a randomized algorithm that has an oblivious competitive ratio of $O(\log d)$. Thus, the algorithm shows our lower bounds to be tight.

3.4 Meshes

The two dimensional $m \times n$ mesh has the set of nodes $\{(r, c) \mid 0 \leq r \leq m-1,\ 0 \leq c \leq n-1\}$. Two nodes are connected by a link if their Hamming distance is one. This section proves an $\Omega(\log n)$ lower bound on the oblivious competitive ratio of any admission control and routing algorithm for $(n+1) \times (n+1)$ meshes with respect to the performance function that determines the number of accepted virtual circuit requests. The section also proves an $\Omega(n)$ lower bound on the oblivious competitive ratio of any greedy admission control and routing algorithm.

We use the following proof strategy. We first consider the admission control and routing problem for the unit capacity $(n+1) \times (n+1)$ mesh. Then, Lemma 2.5.2 extends the lower bounds to the admission control and routing problem for $(n+1) \times (n+1)$ meshes with arbitrary link capacities and bandwidth requirements.

To prove our lower bounds for the unit capacity $(n+1) \times (n+1)$ mesh, we proceed as we did with the unit capacity line of length d. In particular, we construct a set of sequences, $\overline{\sigma}_0, \overline{\sigma}_1, \ldots, \overline{\sigma}_{\log n}$, which will allow us to make use of Theorem 3.2.4. Consider the $(n+1) \times (n+1)$ mesh where n is a power of 2. To construct σ_i, we divide the mesh into 4^i submeshes of size $(\frac{n}{2^i} + 1) \times (\frac{n}{2^i} + 1)$ for every $0 \leq i \leq \log n$. (See Figure 3-2.) Note that bordering submeshes share nodes. The top left hand submesh after the division into 4^i submeshes consists of the

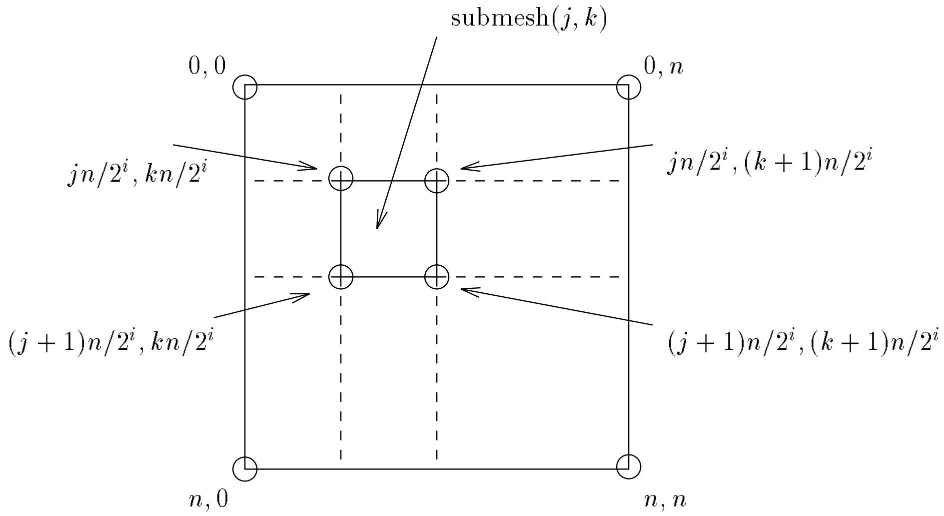

Figure 3-2: Submesh j,k, for $0 \leq j < 2^i$ and $0 \leq k < 2^i$, from the division of $(n+1) \times (n+1)$ mesh into 4^i submeshes of size $(\frac{n}{2^i}+1) \times (\frac{n}{2^i}+1)$.

nodes:

$$\begin{array}{lll} (0,0) & \ldots & (\frac{n}{2^i},0) \\ \vdots & & \vdots \\ (0,\frac{n}{2^i}) & \ldots & (\frac{n}{2^i},\frac{n}{2^i}) \end{array}$$

In general, the submesh j,k, for $0 \leq j < 2^i$ and $0 \leq k < 2^i$, consists of the nodes:

$$\begin{array}{lll} (j\frac{n}{2^i},k\frac{n}{2^i}) & \ldots & ((j+1)\frac{n}{2^i},1) \\ \vdots & & \vdots \\ (j\frac{n}{2^i},(k+1)\frac{n}{2^i}) & \ldots & ((j+1)\frac{n}{2^i},(k+1)\frac{n}{2^i}) \end{array}$$

Denote by $\overline{\sigma}_i$ the following requests. For each of the 4^i squares of size $(n/2^i+1) \times (n/2^i+1)$ request circuits from all top nodes to all bottom nodes, e.g.,

$$((j\tfrac{n}{2^i},k\tfrac{n}{2^i}),(j\tfrac{n}{2^i},(k+1)\tfrac{n}{2^i}),1)((j\tfrac{n}{2^i}+1,k\tfrac{n}{2^i}),(j\tfrac{n}{2^i}+1,(k+1)\tfrac{n}{2^i}),1)\ldots$$
$$(((j+1)\tfrac{n}{2^i},k\tfrac{n}{2^i}),((j+1)\tfrac{n}{2^i},(k+1)\tfrac{n}{2^i}),1)$$

for submesh j,k, and circuits from all left hand nodes to all right hand nodes, e.g.,

$$((j\tfrac{n}{2^i},k\tfrac{n}{2^i}),((j+1)\tfrac{n}{2^i},k\tfrac{n}{2^i}),1)((j\tfrac{n}{2^i},k\tfrac{n}{2^i}+1),((j+1)\tfrac{n}{2^i},k\tfrac{n}{2^i}+1),1)\ldots$$
$$(j\tfrac{n}{2^i},(k+1)\tfrac{n}{2^i}),((j+1)\tfrac{n}{2^i},(k+1)\tfrac{n}{2^i}),1)$$

for submesh j,k. Now we can prove the following lemma.

Lemma 3.4.1 *Let n be a power of two. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, for the unit capacity $(n+1) \times (n+1)$ mesh has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(n)$.*

Proof. The sequences $\overline{\sigma}_i$ for $0 \leq i \leq \log n$ constructed for the $(n+1) \times (n+1)$ mesh satisfy the conditions of Theorem 3.2.4 for $Z = \log n$. In particular, for all $0 \leq i \leq \log n$, $|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ if $i \neq 0$, all request in $|\overline{\sigma}_i|$ can be accepted when no other requests are accepted from other sequences, and the are $2(n+1)n = 2(n+1)2^{\log n}$ critical links (all of the links of the mesh) such that a request from $\overline{\sigma}_i$ requires $n/2^i = 2^{(\log n)-i}$ critical links. ■

Now consider the case where n is not a power of two.

Lemma 3.4.2 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, for the unit capacity $(n+1) \times (n+1)$ mesh has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(n)$.*

Proof. When n is not a power of 2, the proof is essentially the same as the proof for Lemma 3.4.1, except that it is more complicated notationally. When dividing the mesh into 4^i submeshes, the appropriate floor and ceiling operators need to be used. Furthermore, the sequences $\overline{\sigma}_i$ may only be defined for $0 \leq i \leq (\log n) - 1$ rather than $0 \leq i \leq \log n$ since the division of the mesh into $4^{\lfloor \log n \rfloor}$ submeshes may already have submeshes of size 1×1. ■

We extend our lower bounds to the admission control and routing problem for $(n+1) \times (n+1)$ meshes without the bandwidth and capacity restriction.

Theorem 3.4.3 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, for $(n+1) \times (n+1)$ meshes has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(n)$.*

Proof. The theorem follows immediately from Lemma 2.5.2 and Lemma 3.4.2. ■

We mention that the deterministic lower bound on the competitive ratio for $(n+1) \times (n+1)$ meshes is $\Omega(\sqrt{n})$ [BFKR93].

Lemma 3.4.4 *Any deterministic on-line algorithm, A, that solves the admission control and routing problem, $\mathcal{P}$, on a $(n+1) \times (n+1)$ mesh has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(\sqrt{n})$.*

The separation of the deterministic and greedy lower bounds ($\Omega(\sqrt{n})$ and $\Omega(n)$ respectively) from the oblivious randomized lower bound ($\Omega(\log n)$) for $(n+1) \times (n+1)$ meshes underscores the importance of non-greedy randomized admission control strategies.

3.5 Tree of Meshes and Fat-Tree

For n a power of 2, the $n \times n$ tree of meshes has the following structure. The network has a total of $2n^2 \log n$ nodes. These are arranged in $2 \log n$ levels, each containing n^2 nodes. The levels are numbered 0 through $2 \log n - 1$. Links connect nodes in the same level or in adjacent levels. Level i is a collection of disjoint $m_i \times n_i$ meshes, where $m_i = \frac{n}{2^{\lceil i/2 \rceil}}$ and $n_i = \frac{n}{2^{\lfloor i/2 \rfloor}}$. Notice that for even i $m_i = n_i = m_{i-1}$, and for odd i $2m_i = n_i = n_{i-1}$. Each level i mesh is connected to a unique pair of level $i+1$ meshes. For even i, nodes $(1,1)$, $(1,2)$, $\ldots$, $(1,n_i)$ of the level i mesh are connected to nodes $(1,1)$, $(1,2)$, $\ldots$, $(1,n_{i+1})$, respectively, of one of the level $i+1$ meshes, and nodes $(m_i,1)$, $(m_i,2)$, $\ldots$, (m_i,n_i) of the level i mesh are connected to nodes $(m_{i+1},1)$, $(m_{i+1},2)$, $\ldots$, (m_{i+1},n_{i+1}), respectively, of the other level $i+1$ mesh. For odd i, nodes $(1,1)$, $(2,1)$, $\ldots$, $(m_i,1)$ of the level i mesh are connected to nodes $(1,1)$, $(2,1)$, $\ldots$, $(m_{i+1},1)$, respectively, of one of the level $i+1$ meshes, and nodes $(1,n_i)$, $(2,n_i)$, $\ldots$, (m_i,n_i) of the level i mesh are connected to nodes $(1,n_{i+1})$, $(2,n_{i+1})$, $\ldots$, (m_{i+1},n_{i+1}), respectively, of the other level $i+1$ mesh. The top three levels of a $n \times n$ tree of meshes are shown in Figure 3-3.

A size n fat-tree is essentially the same as a $n \times n$ tree of meshes except that each mesh is replaced by a single node. The bandwidth between meshes is preserved. Thus, a size n fat-tree is complete binary tree of height $2 \log n$ (the tree has n^2 leaves). The root (level 0 node) is connected to each of its children by a link with capacity n. In general, level i nodes are connected to each of their children by a link of capacity $\frac{n}{2^{\lfloor i/2 \rfloor}}$. In practice, fat-trees are used rather than trees of meshes [LAD+92].

This section proves an $\Omega(\log \log n)$ lower bound on the oblivious competitive ratio of any admission control and routing algorithm for $n \times n$ trees of meshes and size n fat-trees with respect to the performance function that determines the number of accepted virtual circuit requests. The section also proves an $\Omega(\log n)$ lower bound on the oblivious competitive ratio of any greedy admission control and routing algorithm.

We use the same proof strategy as for the lower bounds on the mesh. In particular, we first prove the lower bounds on the admission control and routing problem for the unit capacity $n \times n$ tree of meshes and size n fat-tree. To prove these lower bounds, we construct a set of sequences, $\overline{\sigma}_0, \overline{\sigma}_1, \ldots, \overline{\sigma}_{\log\log n}$, which will allow us to make use of Theorem 3.2.4. The same set of sequences can be used for both the $n \times n$ tree of meshes and the size n fat-tree.

We construct the sequences on the $n \times n$ tree of meshes. Consider the $n \times n$ tree of meshes where $\log n$ is a power of two. To define $\overline{\sigma}_i$ on the tree of meshes, we need some additional notation. A $n \times n$ tree of meshes can be constructed from four $n/2 \times n/2$ trees of meshes. See Figure 3-3. In particular, the root mesh is a $n \times n$ mesh, the left and right children of the root are $n \times n/2$ meshes and the grandchildren of the root are four $n/2 \times n/2$ trees of meshes. We denote a $n \times n$ tree of meshes by T^d where $d = \log n$. Furthermore, we introduce the following notation to refer to each of the components of T^d. Node (i, j) in the top mesh is denoted by $(T^d, t, (i, j))$; node (i, j) in the left child mesh is denoted by $(T^d, l, (i, j))$; node (i, j) in the right child mesh is denoted by $(T^d, r, (i, j))$. The $n/2 \times n/2$ trees of meshes that are the grandchildren of T^d are numbered from 1 to 4 starting with 1 for the left most $n/2 \times n/2$ trees of meshes. Thus, the left most $n/2 \times n/2$ tree of meshes is denoted by T_1^{d-1}, and the right most $n/2 \times n/2$ tree of meshes is denoted by T_4^{d-1}.

For two nodes in the same mesh, for example, the nodes $(T, t, (i, j))$ and $(T, t, (i', j'))$, let $p((T, t, (i, j)), (T, t, (i', j')))$ be the row-column path from $(T, t, (i, j))$ to $(T, t, (i', j'))$. (A row-column path reaches $(T, t, (i', j'))$ by first moving in row i to column j' and then moving in column j' to node $(T, t, (i', j'))$.) For connected nodes at different levels, e.g., $(T, l, (1, 1))$ and $(T, t, (1, 1))$ let $p((T, l, (1, 1)), (T, t, (1, 1)))$ be the single link path between the nodes. See Figure 3-3.

Next, we define sets of paths in T^d. In particular, $p^l(T^d)$ defines an order set of $n = 2^d$ paths, where the i^{th} path, $p_i^l(T^d)$, starts at a leaf node and terminates in node $(T^d, t, (i, 1))$, the first node in the i^{th} row of the root $n \times n$ mesh of T^d. The leaf node of the path is picked so that the set of paths $p^l(T^d)$ has the following property: whenever possible go to the left child mesh. In Figure 3-4 the set of links that are used as a result of this property are indicated by

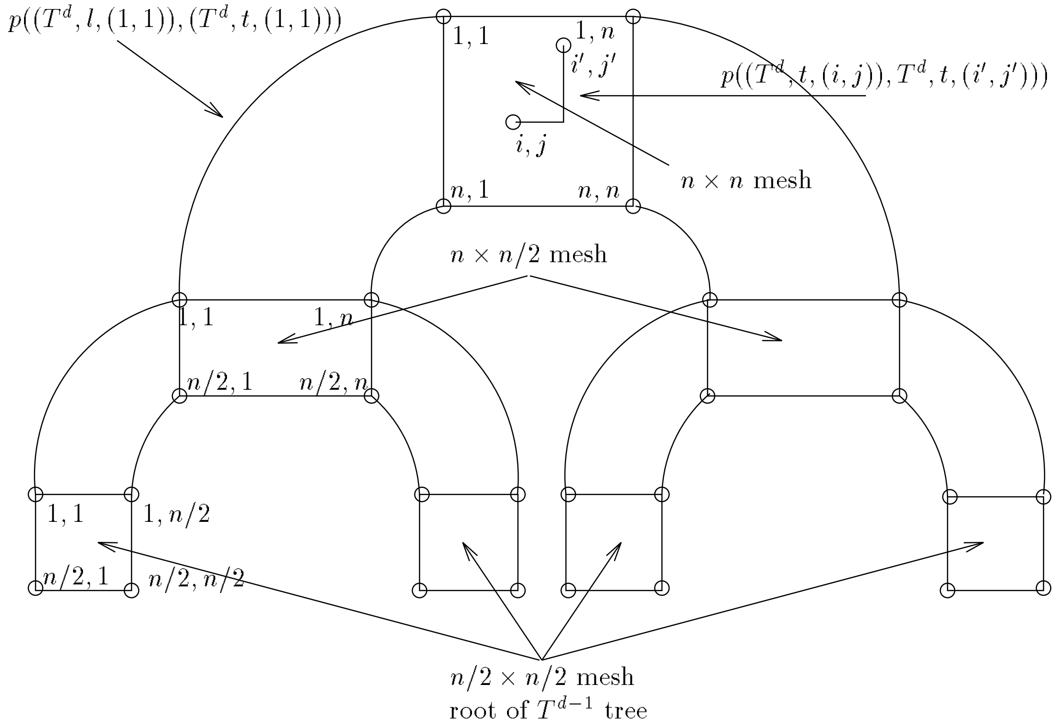

Figure 3-3: Structure and notation for $n \times n$ tree of meshes.

dark shaded circles for the top three levels of an $n \times n$ tree of meshes. This property has the following consequence. Level $i = 2\log(n/k)$ of a $n \times n$ tree of meshes consists of n^2/k^2 $k \times k$ meshes, each of which is the root of a $k \times k$ tree of meshes. As a result of picking left child mesh whenever possible, we use all k of the links that connect the $k \times k$ mesh to its parent mesh for n/k of the $k \times k$ meshes. We denote this set of links as the $p^l(T^d)$-*k-critical* links. We denote the $k \times k$ trees of meshes for which all k of the links to the parent mesh are used by $T_1^{\log k} \ldots T_{n/k}^{\log k}$, where the indices are assigned from left to right. Now define the $p^l(T^d)$-*critical* links to be the union over all $p^l(T^d)$-*k-critical* links for all k such that $\log k$ is odd. Clearly, there are $nd/2$ $p^l(T^d)$-*critical* links. The set $p^r(T^d)$ is defined analogously. In particular, $p^r(T^d)$ is an order set of $n = 2^d$ paths, where the i^{th} path, $p_i^r(T^d)$, starts at a leaf node and terminates in node $(T^d, t, (i, n))$, the last node in the i^{th} row of the root $n \times n$ mesh of T^d. The leaf node of the path is picked so that the set of paths $p^r(T^d)$ has the following property: whenever possible go to the right child mesh. See Figure 3-4.

We now provide a formal definition for $p^l(T^d)$ and $p^r(T^d)$. The sets can be constructed inductively as follows. First consider the base case. T^0 consists of a root node $(T^0, t, (1,1))$ and

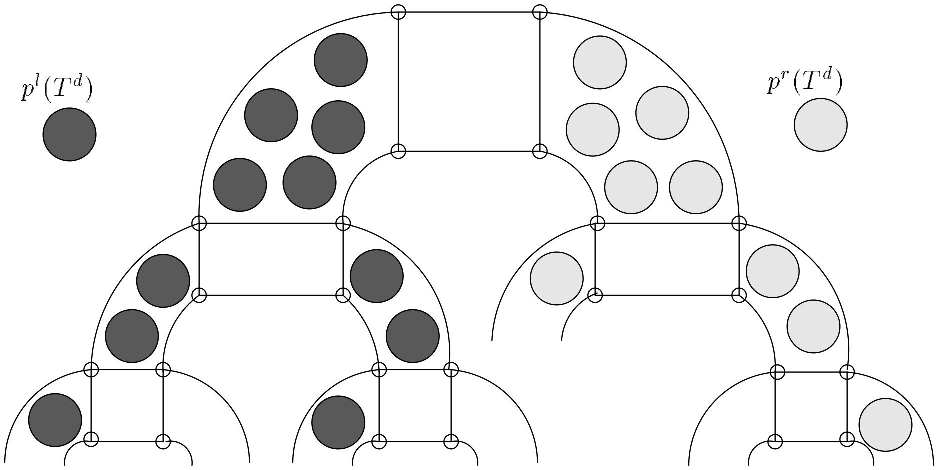

Figure 3-4: Paths used by $p_i^l(T^d)$ and $p_i^r(T^d)$.

the leave nodes $(T^0, l, (1,1))$ and $(T^0, r, (1,1))$. Thus, $p_1^l(T^0) = p((T^0, l, (1,1)), (T^0, t, (1,1)))$ and $p_1^r(T^0) = p((T^0, r, (1,1)), (T^0, t, (1,1)))$. Both $p^l(T^0)$ and $p^r(T^0)$ are singleton sets. We now construct $p^l(T^d)$ and $p^r(T^d)$ from $p^l(T^{d-1})$ and $p^r(T^{d-1})$. (See Figure 3-5.)

$$p_1^l(T^d) \quad = \quad p_1^l(T_1^{d-1})p((T_1^{d-1}, t, (1,1)), (T^d, l, (1,1)))p((T^d, l, (1,1)), (T^d, t, (1,1))).$$

In general, for $1 \leq i \leq n/2$,

$$\begin{aligned} p_i^l(T^d) \quad = \quad & p_i^l(T_1^{d-1})p((T_1^{d-1}, t, (i,1)), (T_1^{d-1}, t, (1,i)))p((T_1^{d-1}, t, (1,i)), (T^d, l, (i,1))) \\ & p((T^d, l, (i,1)), (T^d, l, (1,i)))p((T^d, l, (1,i)), (T^d, t, (i,1))). \end{aligned}$$

Furthermore, for $n/2 < i \leq n$,

$$\begin{aligned} p_i^l(T^d) \quad = \quad & p_i^l(T_2^{d-1})p((T_2^{d-1}, t, (i - n/2, 1)), (T_2^{d-1}, t, (1, i - n/2))) \\ & p((T_2^{d-1}, t, (1, i - n/2)), (T^d, l, (n + 1 - i, n))) \\ & p((T^d, l, (n + 1 - i, n)), (T^d, l, (1,i)))p((T^d, l, (1,i)), (T^d, t, (i,1))). \end{aligned}$$

Similarly consider $1 \leq i \leq n/2$ for $p_i^r(T^d)$,

$$\begin{aligned} p_i^r(T^d) &= p_i^r(T_3^{d-1})p((T_3^{d-1}, t, (n/2+1-i, n/2)), (T_3^{d-1}, t, (1, i))) \\ &\quad p((T_3^{d-1}, t, (1, i)), (T^d, r, (i, 1)))p((T^d, r, (i, 1)), (T^d, r, (1, i))) \\ &\quad p((T^d, r, (1, i)), (T^d, t, (n+1-i, n))). \end{aligned}$$

Finally consider $n/2 < i \leq n$ for $p_i^r(T^d)$,

$$\begin{aligned} p_i^r(T^d) &= p_{i-n/2}^r(T_4^{n/2})p((T_4^{n/2}, t, (n+1-i, n/2)), (T_4^{n/2}, t, (1, i-n/2))) \\ &\quad p((T_4^{n/2}, t, (1, i-n/2)), (T^n, r, (n+1-i, n))) \\ &\quad p((T^n, r, (n+1-i, n)), (T^n, r, (1, i)))p((T^n, r, (1, i)), (T^n, t, (n+1-i, n))). \end{aligned}$$

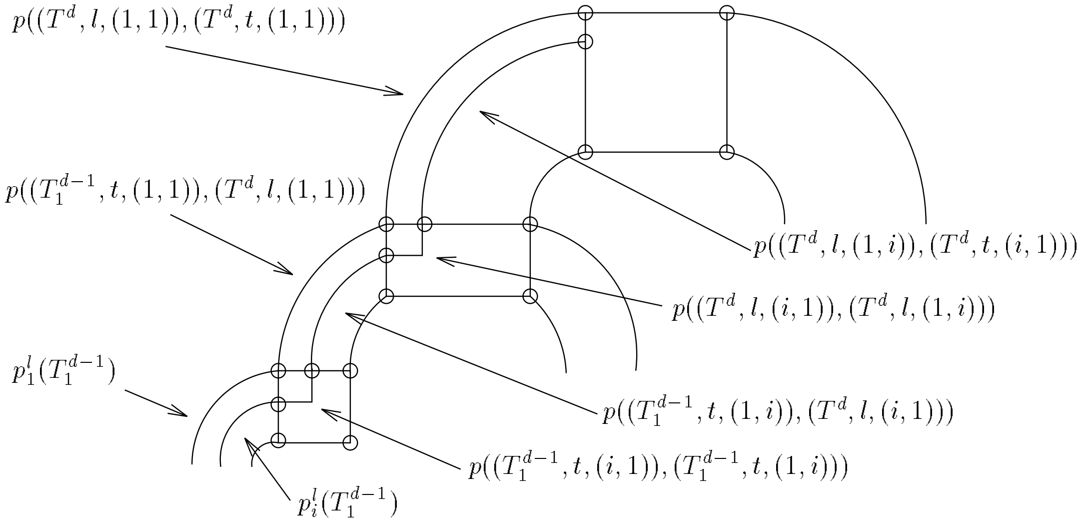

Figure 3-5: Construction of $p_i^l(T^d)$.

We define another ordered set of paths, $p(T^d)$, based on $p^l(T^d)$ and $p^r(T^d)$. In particular the set $p(T^d)$ consists of n leaf node to leaf node paths that each pass through the root $n \times n$ mesh. The i^{th} path is constructed by concatenating the i^{th} path from $p^l(T^d)$ with the $(n+1-i)^{th}$ path from $p^r(T^d)$. Formally, for $1 \leq i \leq n$,

$$p_i(T^n) = p_i^l(T^n)p((T^n, t, (i, 1)), (T^n, t, (i, n)))p_{n+1-i}^r(T^n).$$

To define $\overline{\sigma}_i$ we need to define one more ordered set of paths. Consider $T_i^{\log k}$ for some $1 \leq i \leq n/k$. Recall that $T_i^{\log k}$ is the i^{th} (counting from the left) $k \times k$ tree of meshes whose links to the parent mesh are among the $p^l(T^d)$-k-*critical* links. Now define $p(T_i^{\log k}, T^{\log k'})$, where $1 \le k' < k \le n$, to be the set of paths constructed by modifying the paths in $p(T_i^{\log k})$ to go to the right child mesh once they reach the $k' \times k'$ tree of meshes. See Figure 3-6. The important property of $p(T_i^{\log k}, T^{\log k'})$ is that the paths in $p(T_i^{\log k}, T^{\log k'})$ no longer intersect with the paths in $p(T_j^{\log k''})$ for $\log k'' < \log k'$. To provide the formal definition of $p(T_i^{\log k}, T^{\log k'})$, we require once piece of additional notation. Consider $T_j^{\log k'}$ for some $1 \le j \le n/k'$. Since each $T_i^{\log k}$ contains k/k' $T_j^{\log k'}$, the $T_j^{\log k'}$ contained in $T_i^{\log k}$ have index j between $(i-1)k/k'$ and ik/k'. Thus, the j^{th} path in $p(T_i^{\log k})$ goes through the root of $T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}$. Define $p_j(T_i^{\log k}) \mid k'$ to be the j^{th} path of $p(T_i^{\log k})$ where the segment from the left leaf to the top node of $T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}$, $(T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}, t, (1,i))$, is removed. See Figure 3-6. Now define $p_i(T_i^{\log k}, T^{k'})$ as follows. For $1 \le j \le k$,

$$
\begin{aligned}
&p_j(T_i^{\log k}, T^{\log k'}) = \\
&\quad p^r_{j-k'\lfloor\frac{j}{k'}\rfloor}(T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}) \\
&\quad p((T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}, t, (k'+1-j-k'\lfloor\frac{j}{k'}\rfloor, k)), (T^{\log k'}_{(i-1)\frac{k}{k'}+\lfloor\frac{j}{k'}\rfloor}, t, (1, j-k'\lfloor\frac{j}{k'}\rfloor)) \\
&\quad p_j(T_i^{\log k}) \mid k'.
\end{aligned}
$$

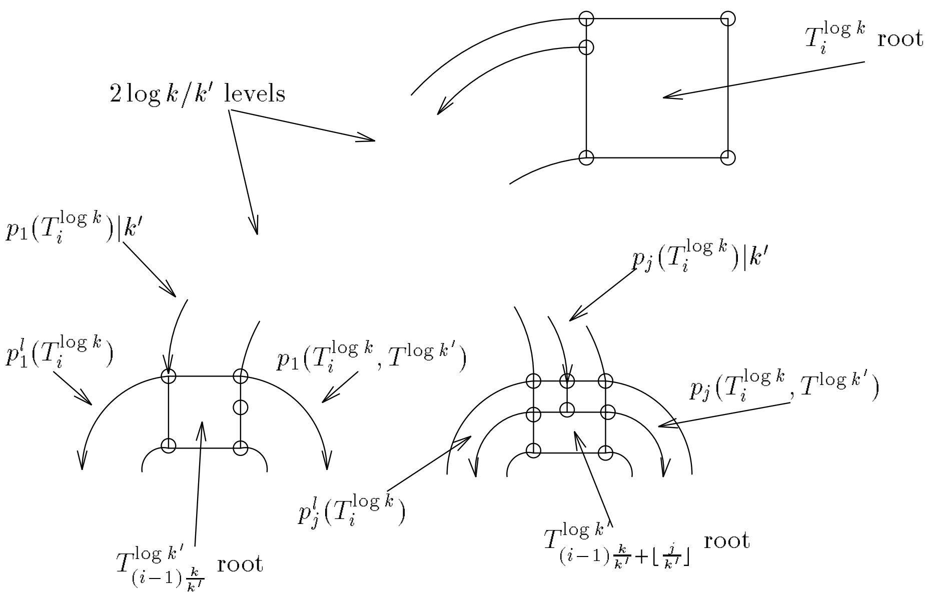

Figure 3-6: Construction of $p(T_i^{\log k}, T^{\log k'})$.

We now define σ by defining the sequences $\overline{\sigma}_i$. The sequence $\overline{\sigma}_0$ consists of requests generated from the paths in $p(T^d)$. Define $s(p_i(T^d))$ to be the left leaf of path $p_i(T^d)$ and $d(p_i(T^d))$ to be the right leaf of $p_i(T^d)$. Then $\overline{\sigma}_0$ is defined as follows:

$$\overline{\sigma}_0 = (s(p_1(T^d)), d(p_1(T^d)), 1) \ldots (s(p_n(T^d)), d(p_n(T^d)), 1).$$

In general, $\overline{\sigma}_i$ consists of the requests constructed from the paths in

$$\begin{array}{l} p(T_1^{d/2^i}), \ldots, p(T_{n/(2^{d/2^i})}^{d/2^i}), \\ p(T_1^{2d/2^i}, T^{d/2^i+1}), \ldots, p(T_{n/(2^{2d/2^i})}^{2d/2^i}, T^{d/2^i+1}), \\ \ldots, \\ p(T_1^{(j+1)d/2^i}, T^{jd/2^i+1}), \ldots, p(T_{2^{(j+1)d/2^i}}^{(j+1)d/2^i}, T^{jd/2^i+1}), \\ \ldots, \\ p(T_1^d, T^{(2^i-1)d/2^i+1}). \end{array}$$

Now we can prove the lower bounds for the unit capacity $n \times n$ tree of meshes using Theorem 3.2.4.

Lemma 3.5.1 *Let $d = \log n$ and let d be a power of 2. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity $n \times n$ tree of meshes has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The sequences $\overline{\sigma}_i$ for $0 \leq i \leq (\log d) - 1$ constructed for the $n \times n$ tree of meshes satisfy the conditions of Theorem 3.2.4 for $Z = (\log d) - 1$.

Each $p(T_i^{\log k}, T^{\log k'})$ contains k paths. Thus, the union over $p(T_i^{\log k})$ for $1 \leq i \leq n/k$ contains n paths. As a consequence, $\overline{\sigma}_i$ contains $n2^i$ requests. Thus, $|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ if $i \neq 0$. Furthermore, it is possible to accept all requests in $\overline{\sigma}_i$ when no other requests are accepted from other sequences. Finally, there are $nd/2 = n2^{(\log d)-1}$ $p^l(T^d)$-*critical* links. Any request from sequence $\overline{\sigma}_i$ consumes $d/2^{i+1} = 2^{(\log d)-1-i}$ $p^l(T^d)$-*critical* links. ∎

Now consider the case where $\log n$ is not a power of two.

Lemma 3.5.2 *Let $d = \log n$. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity $n \times n$ tree of meshes has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. When d is not a power of 2, the proof is essentially the same as the proof for Lemma 3.5.1, except that it is more complicated notationally. In particular, when constructing the sequences $\overline{\sigma}_i$ the appropriate floor and ceiling operators need to be used to divide the number space between 1 and d into 2^i nearly equal sized intervals. (If the boundaries of one such interval are x,y, where $x < y$, the sequence $\overline{\sigma}_i$ would include, among others, the requests constructed from the paths in $p(T_j^y, T^x)$.) Furthermore, the sequences $\overline{\sigma}_i$ may only be defined for $0 \leq i \leq (\log d) - 2$ rather than $0 \leq i \leq (\log d) - 1$ since the division of the number space between 1 and d into $2^{\lfloor \log d \rfloor - 1}$ intervals may already have several intervals where $|x - y| = 1$. ∎

We extend our lower bounds to the admission control and routing problem for $n \times n$ trees of meshes without the bandwidth and capacity restriction.

Theorem 3.5.3 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on $n \times n$ trees of meshes has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log \log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The theorem follows immediately from Lemma 2.5.2 and Lemma 3.5.2. ∎

The separation of the greedy lower bound ($\Omega(\log n)$) from the oblivious randomized lower bound ($\Omega(\log \log n)$) for $n \times n$ trees of meshes underscores the importance of non-greedy admission control strategies.

Next, we consider fat-trees. Let the admission control and routing problem for the unit bandwidth size n fat-tree be the admission control and routing problem for the size n fat-tree with the restriction that all requests have a bandwidth requirement of 1.

Lemma 3.5.4 *Let $d = \log n$. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, for the unit bandwidth size n fat-tree has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log \log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The same set of sequences that is used in Lemma 3.5.2 can be used. The argument that Theorem 3.2.4 is applicable to the set of sequences is unchanged. ∎

We extend our lower bounds to the admission control and routing problem for size n fat-trees without any bandwidth restriction.

Theorem 3.5.5 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on size n fat-trees has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, that is in $\Omega(\log\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The theorem follows immediately from Lemma 2.5.2 and Lemma 3.5.4. ■

The separation of the greedy lower bound ($\Omega(\log n)$) from the oblivious randomized lower bound ($\Omega(\log\log n)$) for size n fat-trees underscores the importance of non-greedy admission control strategies. In Chapter 4 we provide a non-greedy algorithm that has a competitive ratio of $O(\log\log n)$. Thus, the algorithm shows our lower bounds to be tight.

3.6 Hypercube

The $\log n$ dimensional hypercube is a graph consisting the set of nodes $\{0,1\}^{\log n}$ and edges connecting every pair of nodes with Hamming distance 1. This section proves an $\Omega(\log\log n)$ lower bound on the oblivious competitive ratio of the admission control and routing problem for the $\log n$ dimensional hypercube with respect to the performance function that determines the number of accepted virtual circuit requests. The section also proves an $\Omega(\log n)$ lower bound on the oblivious competitive ratio of any greedy admission control and routing algorithm.

We use the same proof strategy as for the lower bounds on the mesh. In particular, we first prove the lower bounds on the admission control and routing problem for the unit capacity $\log n$ dimensional hypercube. To prove these lower bounds, we construct a set of sequences, $\overline{\sigma}_0, \overline{\sigma}_1, \ldots, \overline{\sigma}_{\log\log n}$, which will allow us to make use of Theorem 3.2.4.

Consider the $\log n$ dimensional hypercube where $\log n$ is a power of two. Let $d = \log n$. Consider two nodes that differ in k bits. The shortest path between the nodes has length k. The sequence $\overline{\sigma}_i$ will consist of $n2^i$ requests between nodes that differ in $\frac{\log n}{2^i}$ bits. Consider sequence $\overline{\sigma}_0$. For one potential approach, $\overline{\sigma}_0$ consists of requests whose endpoints are bitwise complements of each other. The resulting paths have length $\log n$. However, if the required bandwidth of each request is one, routing the requests in this sequence would require twice the available capacity. Thus, $\overline{\sigma}_0$ should consists of half of all of the requests that can be constructed by using all n nodes and their bitwise complement as endpoints.

Consider a four node hypercube consisting of the nodes $(1,1)$, (1.0), $(0,1)$, and $(0,0)$. See Figure 3-7. The set of all possible requests whose endpoints are bitwise complements of each other is: $\{((1,1),(0,0)),((0,0),(1,1)),((1,0),(0,1)),((0,1),(1,0))\}$. It is easy to see that the set of requests $\{((1,1),(0,0)),((0,0),(1,1))\}$ can be routed, while the set of request $\{((1,1),(0,0)),$ $((0,1),(1,0))\}$ cannot. We now provide an inductive construction of a set of requests whose endpoints are bitwise complements of each other such that all of the requests in the set can be routed and all of the links in the network are needed to route the requests.

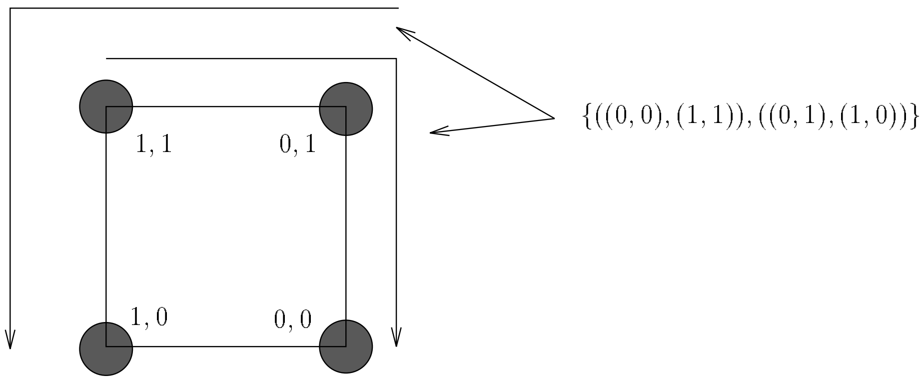

Figure 3-7: Problems with picking $\{((1,1),(0,0)),((0,1),(1,0))\}$ as the request set σ for a four node hypercube.

Let H_n be an n node $d = \log n$ dimensional hypercube. Let $S_d = (s_1, d_1)\ldots(s_n, d_n)$ and $S'_d = (s'_1, d'_1)\ldots(s'_n, d'_n)$ be two ordered sets of node pairs with the following properties:

1. Both S_d and S'_d consist of 2^d pairs. Specifically, $|S_d| = |S'_d| = 2^d = n$.

2. Each node in H_n appears exactly once as a source and once as a destination in $S_d \cup S'_d$. Specifically, for each node $v \in H_n$ there exists exactly one $(s,d) \in S_d \cup S'_d$ such that $v = s$. Furthermore, there exists exactly one $(s,d) \in S_d \cup S'_d$ such that $v = d$

3. Routing bandwidth one requests between each pair in S_d uses all links in H_n if routed on paths that complement bits from left to right. Similarly, routing bandwidth one requests between each pair in S'_d uses all links in H_n if routed on paths that complement bits from left to right.

4. For each $(s,d) \in S_d \cup S'_d$, s is the bitwise complement of d.

Define $S_1 = (1, 0)$ and $S'_1 = (0, 1)$. S_1 and S'_1 trivially satisfy Properties 1 – 4. Now define S_{i+1} and S'_{i+1} from S_i and S'_i.

$$S_{i+1} = (0s_1, 1d_1)\ldots(0s_{2^i}, 1d_{2^i})(1s'_1, 0d'_1)\ldots(1s'_{2^i}, 0d'_{2^i}).$$

Similarly,

$$S'_{i+1} = (1s_1, 0d_1)\ldots(1s_{2^i}, 0d_{2^i})(0s'_1, 1d'_1)\ldots(0s'_{2^i}, 1d'_{2^i}).$$

Lemma 3.6.1 *If S_i and S'_i satisfy Properties 1 – 4, then S_{i+1} and S'_{i+1} satisfy Properties 1 – 4*

Proof. Property 1 is satisfied since $|S_{i+1}| = |S'_{i+1}| = |S_i| + |S'_i|$.

Consider Property 2. For each node v in H_{2^i}, the hypercube $H_{2^{i+1}}$ contains two nodes $0v$ and $1v$. By property 2 for S_i and S'_i, v appears exactly once as a source in $S_i \cup S'_i$. Without loss of generality, assume that v appears as a source in S_i, such that $v = s_j$. By inspection of the definition of S_{i+1} and S'_{i+1} we see that $0s_j$ and $1s_j$ each appear exactly once in $S_{i+1} \cup S'_{i+1}$. Specifically, $0s_j$ appears as a source in S_{i+1} and $1s_j$ appears as a source in S'_{i+1}. The same argument shows that $0v$ and $1v$ each appear exactly once as a destination in $S_{i+1} \cup S'_{i+1}$.

Consider Property 3. The new links added to H_{2i+i} are between $0v$ and $1v$ for each $v \in H_{2i}$. Since we are routing by complementing bits from left to right, Property 3 requires that either $0v$ or $1v$ appear exactly once in S_{i+1} and exactly once in S'_{i+1}. This follows directly from the construction of S_{i+1} and S'_{i+1} and Property 2 for S_i and S'_i.

Consider Property 4. This property is trivially satisfied by the construction of S_{i+1} and S'_{i+1} and Property 4 for S_i and S'_i. ∎

Now we can constructs the sequence $\overline{\sigma}_i$. In particular, $\overline{\sigma}_0$ consists of all of the requests generated by requesting a bandwidth one circuit between each source and destination pair in S_d. To construct $\overline{\sigma}_i$ we construct 2^i requests for each request in $\overline{\sigma}_0$. Consider any request $(v_0, v_d, 1)$ in $\overline{\sigma}_0$. Let P be the path from v_0 to v_d that complements bits from left to right. We label the nodes along the path P from v_0 to v_d by $v_0, v_1 \ldots v_d$. (Recall our assumption that d is a power of 2.) To construct 2^i requests for $\overline{\sigma}_i$, divide the path between v_0 and v_d into 2^i equal length paths and request a bandwidth one virtual circuit between the endpoints of each of the

2^i paths. Let the duration of all the requests be infinite. Formally:

$$
\begin{aligned}
\overline{\sigma}_1 &= \bigcup_{(v_0,v_d,1)\in\overline{\sigma}_0} \{(v_0, v_{d/2}, 1)(v_{d/2}, v_d, 1)\} \\
&\vdots \\
\overline{\sigma}_i &= \bigcup_{(v_0,v_d,1)\in\overline{\sigma}_0} \{(v_0, v_{d/2^i}, 1)(v_{d/2^i}, v_{2d/2^i}, 1) \\
&\qquad\qquad (v_{2d/2^i}, v_{3d/2^i}, 1)\ldots(v_{(2^i-1)d/2^i}, v_d, 1)\} \\
&\vdots \\
\overline{\sigma}_{\log d} &= \bigcup_{(v_0,v_d,1)\in\overline{\sigma}_0} \{(v_0, v_1, 1)(v_1, v_2, 1)\ldots(v_{d-1}, v_d, 1)\}
\end{aligned}
$$

Lemma 3.6.2 *Let $d = \log n$ and let d be a power of 2. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity $\log n$ dimensional hypercube has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The sequences $\overline{\sigma}_i$ for $0 \le i \le \log d$ constructed for the $\log n$ dimensional hypercube satisfy the conditions of Theorem 3.2.4 for $Z = \log d$.

First, $|\overline{\sigma}_i| = 2|\overline{\sigma}_{i-1}|$ if $i \neq 0$. Furthermore, by construction of $\overline{\sigma}_i$ and Property 3, all requests in $\overline{\sigma}_i$ can be accepted when no other requests are accepted from other sequences. There are $dn/2 = \frac{n}{2}2^{\log d}$ critical links (all of the links of the hypercube). By construction of $\overline{\sigma}_i$ and Property 4 any request from sequence $\overline{\sigma}_i$ consumes $d/2^i = 2^{(\log d)-i}$ critical links. ∎

Now consider the case where $\log n$ is not a power of two.

Lemma 3.6.3 *Let $d = \log n$. Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on the unit capacity $\log n$ dimensional hypercube has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log\log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. When d is not a power of 2, the proof is essentially the same as the proof for Lemma 3.6.2, except that it is more complicated notationally. In particular, when constructing the sequences $\overline{\sigma}_i$ the appropriate floor and ceiling operators need to be used to divide the number space between d and 0 into 2^i nearly equal sized intervals. Furthermore, the sequences $\overline{\sigma}_i$ may only be defined for $0 \leq i \leq (\log d) - 1$ rather than $0 \leq i \leq (\log d)$ since the division of the d length path into $2^{\lfloor \log d \rfloor}$ intervals may already have several paths of length 1. ■

We extend our lower bound to the admission control and routing problem for $\log n$ dimensional hypercubes without the bandwidth and capacity restriction.

Theorem 3.6.4 *Any randomized on-line algorithm, A_r, that solves the admission control and routing problem, $\mathcal{P}$, on $\log n$ dimensional hypercubes has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log \log n)$. Furthermore, any greedy randomized on-line algorithm, A_r, that solves $\mathcal{P}$ has an oblivious competitive ratio, $\mathcal{C}^b_{\mathcal{P},P}(A_r)$, in $\Omega(\log n)$.*

Proof. The theorem follows immediately from Lemma 2.5.2 and Lemma 3.6.3. ■

The separation of the greedy lower bound ($\Omega(\log n)$) from the oblivious randomized lower bound ($\Omega(\log \log n)$) for $\log n$ dimensional hypercubes underscores the importance of non-greedy admission control strategies.

3.7 Hierarchical Backbone Networks

A hierarchical backbone network can be decomposed into many low diameter regions (access networks) and an arbitrary region (backbone network) that connects the access networks. A hierarchical backbone network is defined formally as follows.

Definition 3.7.1 (hierarchical backbone network) *A connected network G is a* hierarchical backbone network *if it can be decomposed into a connected* backbone network G^b *and a set of* access networks G^a_i *such that all access networks G^a_i and G^a_j for $j \neq i$ are disjoint (i.e. they have no nodes or links in common) and all access networks G^a_i have exactly one node in common with the backbone network G^b.*

The definition ensures that different access networks can only communicate with paths that use the backbone network. Furthermore, no simple path between two different nodes in the backbone network passes through any link in any access network.

Lemma 3.7.2 *Let G be a hierarchical backbone network consisting of the backbone network G^b and access networks G_i^a. Then there exists no simple path between two different nodes in G^b that passes through any link in any access network G_i^a.*

Proof. The lemma follows from the facts that access networks are disjoint and each have exactly one node in common with the backbone network. Thus, any path that originates and terminates in the backbone network G^b, but uses a link in the access network G_i^a must pass through the common node for G_i^a and G^b twice. Hence, the path is not simple. ■

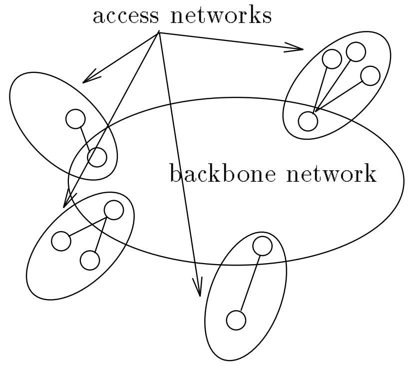

Figure 3-8: An example of a hierarchical backbone network.

We show that a lower bound on the competitive ratio for the backbone network or for any of the access networks implies the same lower bound on the competitive ratio for the entire hierarchical backbone network. Let $\mathcal{P}(\{G\})$ be the admission control and routing problem for G.

Theorem 3.7.3 *Let G^n be a hierarchical backbone network consisting of the backbone network G^b and access networks G_i^a for $i \in [0, I]$. Let P be the performance function of Definition 2.3.7.*

If $\mathcal{C}_{\mathcal{P}(\{G^b\}),P}(A) \geq K^b$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G^b\})$ and, for all $i \in [0, I]$, $\mathcal{C}_{\mathcal{P}(\{G_i^a\}),P}(A) \geq K_i^a$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G_i^a\})$, then $\mathcal{C}_{\mathcal{P}(\{G^n\}),P}(A') \geq max\{\{K^b\} \cup_{i \in [0,I]} \{K_i^a\}\}$ for all deterministic on-line algorithms A' that solve $\mathcal{P}(\{G^n\})$.

Further, if $\mathcal{C}^b_{\mathcal{P}(\{G^b\}),P}(A_r) \geq K^b$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G^b\})$ and, for all $i \in [0, I]$, $\mathcal{C}^b_{\mathcal{P}(\{G_i^a\}),P}(A_r) \geq K_i^a$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G_i^a\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G^n\}),P}(A'_r) \geq max\{\{K^b\} \cup_{i \in [0,I]} \{K_i^a\}\}$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}(\{G^n\})$.

Both statements also apply to the performance function in Definition 2.3.8.

Proof. We consider deterministic algorithms with the performance function of Definition 2.3.7. The arguments for randomized algorithms and the performance function in Definition 2.3.8 are the same.

Consider any graph, say G_i^a, from the set $\{\{G^b\} \cup_{i\in[0,I]} \{G_i^a\}\}$. Let $G = G_i^a$ and $G' = G^n$. Now, by Lemma 3.7.2, G and G' meet the conditions of Lemma 2.7.2. As a result, $\mathcal{C}_{\mathcal{P}(\{G^n\}),P}(A') \geq K_i^a$ for all deterministic on-line algorithms A' that solve $\mathcal{P}(\{G^n\})$. The lemma follows since $\mathcal{C}_{\mathcal{P}(\{G^n\}),P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}(\{G^n\})$ if $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$ and graph G is in the set $\{\{G^b\} \cup_{i\in[0,I]} \{G_i^a\}\}$. ■

3.8 General Topology Networks

The results in this section further support the importance of using non-greedy admission control in general topology networks. In this section, we consider the performance function that determines the number of accepted virtual circuit requests. The admission control and routing problem for general topology networks is the admission control and routing problem where the set of graphs, $\mathcal{G}$, ranges over all graphs. It is trivial to prove an $\Omega(n)$ lower bound for the competitive ratio of any *greedy* on-line admission control and routing algorithm for n node general topology networks. (Just consider the n node line and Theorem 3.3.2.) The lower bound for all (including non-greedy) admission control and routing algorithms on general topology networks is $\Omega(\log n)$ [AAP93], suggesting that either a randomized or non-greedy approach is needed. In fact, the non-greedy algorithm of [AAP93], presented in Section 4.2, achieves an $O(\log n)$ competitive ratio.

This section strengthens the support for non-greedy admission control by proving an $\Omega(n)$ lower bound for the *greedy* admission control and routing problem for general topology networks (cf. Definition 2.3.2). The key difference between the competitive ratio of a greedy on-line algorithm for admission control and routing and the competitive ratio of a greedy on-line algorithm for *greedy* admission control and routing is that the optimal off-line algorithm is also forced to use a greedy admission control strategy when determining the competitive ratio for greedy admission control and routing. Thus, this section shows that the $\Omega(n)$ lower bound for greedy on-line algorithms when the optimal off-line algorithm can use non-greedy admission control still holds even when the optimal off-line algorithm must use greedy admission control.

Our approach is to prove the lower bound for a specific network. Then, by Lemma 2.5.2, the lower bound extends to general topology networks. Consider the network $G = (V, E)$ in Figure 3-9 consisting of $2n + 2$ nodes. Let b be a function from E to $\{1\}$. Thus all links have capacity 1. Define $\sigma_0 = (G, b)$. Call the links between nodes v_i and v_{i+1} for $3 \leq i \leq n-1$ the *v-links* and the links between nodes u_i and u_{i+1} for $3 \leq i \leq n-1$ the *u-links*.

We use the following proof strategy. Consider any routing algorithm A. We construct a request sequence, σ, such that $P(\sigma, A(\sigma)) \leq O(1)$, i.e., A accepts $O(1)$ requests, while $P_o(\sigma) = \Omega(n)$, i.e., an optimal off-line strategy can accept $O(n)$ requests. The request sequence σ consists of σ_0 and the concatenation of four subsequences $\overline{\sigma}_0$, $\overline{\sigma}_1$, $\overline{\sigma}_2$, and $\overline{\sigma}_3$. Each request in σ has bandwidth $1/r$ for some even integer r. The construction of σ depends on A.

Define $\overline{\sigma}_0 = \sigma_1 \ldots \sigma_r$ such that $\sigma_i = (s, d, 1/r)$. In other words, $\overline{\sigma}_0$ consists of r requests from s to d with bandwidth requirement $1/r$ each. Since A is greedy, it will accept all requests in $\overline{\sigma}_0$. Consider the state of G after A routes the requests in $\overline{\sigma}_0$. Each request in $\overline{\sigma}_0$ must use either the *v-links* or the *u-links* once. Thus, either the *v-links* or the *u-links* are used by at least $r/2$ of the requests in $\overline{\sigma}_0$. Without loss of generality assume that A routes such that the *v-links* are used by at least $r/2$ of the requests in $\overline{\sigma}_0$.

Now construct $\overline{\sigma}_1$, $\overline{\sigma}_2$, and $\overline{\sigma}_3$. Subsequence $\overline{\sigma}_1$ consists $2r$ requests from u_2 to u_0 with bandwidth requirement $1/r$ each. Specifically, $\overline{\sigma}_1 = \sigma_{r+1} \ldots \sigma_{3r}$ such that $\sigma_i = (u_2, u_0, 1/r)$. Subsequence $\overline{\sigma}_2$ consists of $r/2$ requests from s to d with bandwidth requirement $1/r$ each. Thus, $\overline{\sigma}_2 = \sigma_{3r+1} \ldots \sigma_{7/2r}$ such that $\sigma_i = (s, d, 1/r)$. Finally, subsequence $\overline{\sigma}_3$ consists of $(n-3)r$ requests such that there are r requests between v_i and v_{i+1} for each $3 \leq i \leq n-1$. Specifically, $\overline{\sigma}_3 = \overline{\sigma}_{3,0} \ldots \overline{\sigma}_{3,n-4}$ consists of $n-3$ subsequences, where each subsequence consists of r requests. Define $\overline{\sigma}_{3,i}$ as follows: $\overline{\sigma}_{3,i} = \sigma_{1+(7+2i)/2r} \ldots \sigma_{(9+2i)/2r}$, where $\sigma_i = (v_{i+3}, v_{i+4}, 1/r)$.

Lemma 3.8.1 *$P(\sigma, A(\sigma)) \leq O(1)$.*

Proof. Consider subsequence $\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2$. In the best case, all virtual circuit requests are accepted, so $P(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2), A(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2)) \leq 4r$.

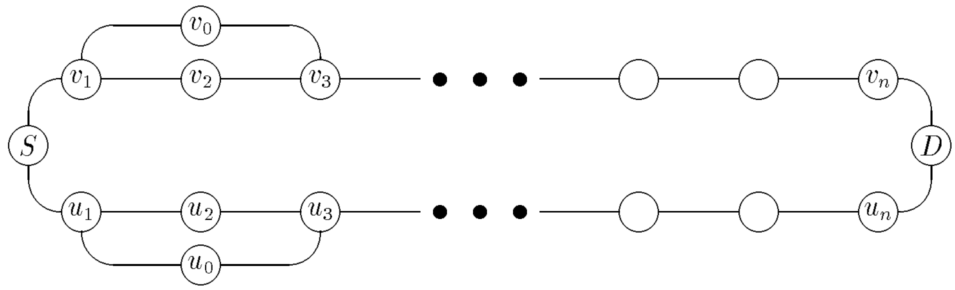

Figure 3-9: Network G with capacity 1 on each link.

Now consider $\overline{\sigma}_3$. The *v-links* can each carry r requests of bandwidth $1/r$. By construction, A routes the requests in $\overline{\sigma}_0$ such that each *v-link* carries at least $r/2$ requests after the requests in $\overline{\sigma}_0$ are routed. Furthermore, the $2r$ requests from u_2 to u_0 in $\overline{\sigma}_1$ ensure that links (u_1, u_2) and (u_1, u_0) carry r requests after A routes $\overline{\sigma}_1$. Thus, the *u-links* are not available for the requests from s to d in subsequence $\overline{\sigma}_2$. So the $r/2$ requests from s to d in subsequence $\overline{\sigma}_2$ must make use of the *v-link*. Since the *v-links* each already carry at least $r/2$ requests after the requests in $\overline{\sigma}_0$, the requests in $\overline{\sigma}_2$ ensure that the *v-links* carry the full r requests after A routes the requests in $\overline{\sigma}_2$. As a consequence, A is unable to route any of the requests in $\overline{\sigma}_3$. Thus, $P(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2\overline{\sigma}_3, A(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2\overline{\sigma}_3)) \leq 4r$. ∎

Lemma 3.8.2 $P_o(\sigma) \geq \Omega(n)$.

Proof. Consider the following off-line routing strategy. Route all r requests in subsequence $\overline{\sigma}_0$ using the *u-links*. As a consequence, none of the requests from u_2 to u_0 in $\overline{\sigma}_1$ can be accepted. Thus, $P_o(\overline{\sigma}_0\overline{\sigma}_1) = r$ and, after routing the requests in $\overline{\sigma}_0\overline{\sigma}_1$, the *v-links* are completely unused. Next route the $r/2$ requests from s to d in subsequence $\overline{\sigma}_2$ using the *v-links*. Thus, $P_o(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2) = 3/2r$ and, after routing the requests in $\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2$, the *v-links* can each still accommodate $r/2$ requests. Now the optimal off-line algorithm is able to accept half of the requests $(n-3)r$ in $\overline{\sigma}_3$, bringing $P_o(\sigma_0\overline{\sigma}_0\overline{\sigma}_1\overline{\sigma}_2\overline{\sigma}_3)$ to $r((n-3)/2 + 3/2) = \frac{n}{2}r$. ∎

We can now prove the lower bound result for G.

Lemma 3.8.3 *Any deterministic on-line algorithm, A, for the greedy admission control and routing problem, $\mathcal{P}$, on network G has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(n)$.*

Proof. The lemma follows directly from Lemma 3.8.1 and Lemma 3.8.1. ∎

We now extend the result to general topology networks.

Theorem 3.8.4 *Any deterministic on-line algorithm, A, for the greedy admission control and routing problem, $\mathcal{P}$, on n node general topology networks has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A)$, in $\Omega(n)$.*

Proof. The theorem follows from Lemma 2.7.1 and Lemma 3.8.3. ■

CHAPTER 4

Competitive Admission Control and Routing Algorithms

4.1 Introduction

This chapter presents several algorithms for admission control and routing. The algorithms illustrate the importance of using non-greedy admission control and randomization. This chapter is organized by topology. Section 4.2 considers general topologies, Section 4.3 considers hierarchical backbone networks (cf. Definition 3.7.1), Section 4.4 considers fat-trees, and Section 4.5 considers trees.

The problems considered in Sections 4.3, 4.4, and 4.5 are not special cases of the problem considered in Section 4.2. In particular, the $O(\log d)$ competitive algorithm for general topology networks of [AAP93], presented in Section 4.2, has the restriction that the bandwidth of every request must be less than $1/\log d$ of the capacity of the lowest capacity link, where d is the length of the longest simple path in the network. Lower bounds in [AAP93] show that this restriction cannot be removed without increasing the competitive ratio of the algorithm. Sections 4.3, 4.4, and 4.5 address the situation where each request may require up to the entire link capacity for some links.

For general topology networks we give a brief presentation of the admission control and routing algorithm of [AAP93]. We do not include any of the complexity or correctness proofs. The algorithm serves as the basis for the practical general topology admission control and routing algorithm presented in Chapter 5. The algorithm is also used as a subroutine by our algorithm for the fat-tree.

We provide a general mechanism for constructing admission control and routing algorithms for hierarchical backbone networks. Consider a specific hierarchical backbone network where the longest simple path in any access network consists of at most d links and there exists an algorithm A_b that achieves a competitive ratio of K on the backbone network. We show how to construct an algorithm that achieves a competitive ratio of $O(d+K)$ for the hierarchical backbone network. This construction can be used to extend the admission control and routing algorithm of [AAP93] to networks where some regions of the network do not meet the bandwidth restrictions imposed by [AAP93]. Hierarchical backbone networks can be used to model several important networks including the telephone network, the Internet, and fat-trees. In fact, we use the techniques developed for hierarchical backbone networks to construct an algorithm that achieves an $O(\log\log n)$ competitive ratio for the size n fat-tree. This matches the $\Omega(\log\log n)$ lower bound of Lemma 3.5.4. Furthermore, the algorithm overcomes the $\Omega(\log n)$ lower bound on the competitive ratio for any greedy admission control algorithm (cf. Lemma 3.5.4).

For trees with diameter d, we provide a randomized admission control and routing algorithm with an oblivious competitive ratio of $O(\log d)$. This matches the $\Omega(\log d)$ lower bound of Lemma 3.3.3. Furthermore, the algorithm overcomes the $\Omega(d)$ lower bound on the competitive ratio for deterministic and greedy algorithms on trees (cf. Proposition 3.3.6 and Lemma 3.3.3).

4.2 General Topology Networks

This section presents an algorithm for a slightly modified version of the admission control and routing problem for general topology networks. In particular, we introduce a bandwidth restriction. The algorithm we present is a simplified[1] version of an algorithm that was originally presented in [AAP93]. We call the algorithm IAAP. The performance function used in this section measures the number of accepted virtual circuit requests .

The *bandwidth restricted admission control and routing problem* is the same as the admission control and routing problem in Definition 2.3.1 except that the maximum bandwidth of any

[1]The actual algorithm presented in [AAP93], called AAP, can handle virtual circuits that have a finite duration. We introduce the AAP algorithm in Chapter 5.

circuit request is restricted. A lower bound in [AAP93] shows that this restriction on the maximum bandwidth cannot be relaxed.

Definition 4.2.1 (bandwidth restricted admission control and routing for $\mathcal{G}$) *Let $\mathcal{G}$ be a set of undirected graphs. The* bandwidth restricted admission control and routing problem for $\mathcal{G}$ *is the same as admission control and routing problem for $\mathcal{G}$ with the following additional condition for S. Let $r_{min}, r_{max} \in \Re^{>0}$ such that $r_{min} \leq r_{max}$. If $(\sigma, \rho) \in S$, then the following condition must hold. If $\sigma_0 = ((V, E), b)$, then $r(\sigma_i) \leq \min_{e \in E}\{b(e)/\log \mu\}$ and $r(\sigma_i) \in [r_{min}, r_{max}]$ for all $1 \leq i < |\sigma|$, where $\mu = 2dr_{max}/r_{min} + 1$ and d is the number of links used by the longest simple path in (V, E).*

The code for the IAAP algorithm is presented in Figure 4-1. The code for IAAP (and all other algorithms in this thesis) is presented using pseudocode. It is straightforward to transform the pseudocode into a sequence of functions A_n from Q^n to R for $n \in \mathbb{N}$. (Recall that Definition 2.1.1 formally defines an on-line algorithm A as a sequence of functions A_n from Q^n to R for $n \in \mathbb{N}$.) The pseudocode can be seen as a function f from a state space, $\mathcal{S}$, and the set of requests Q to the state space $\mathcal{S}$ and the set of responses R.

In the case of the IAAP algorithm, the state after request σ_i is encoded in the utilization function $u_{i+1}(e)$. Using the function f we can construct the sequence of functions used in Definition 2.1.1. Denote by ϱ_i the state just before request σ_i is handled. Let $f(\varrho_i, \sigma_i) = (\varrho_{i+1}, \rho_i)$ and $f(\varrho_0, \sigma_0) = (\varrho_1, \rho_1)$ for some initial state ϱ_0. Now, if $f(\varrho_i, \sigma_i) = (\varrho_{i+1}, \rho_i)$, then $A_{i+1}(\sigma_0 \ldots \sigma_i) = \rho_i$. Essentially, the state ϱ_i encodes the information needed from all of the prior requests $\sigma_0 \ldots \sigma_{i-1}$.

Our pseudocode uses several notational conventions. Assignment is denoted by '=' while logical comparison is denoted by '=='. (This convention is consistent with the C programming language.) Consider any state variable x. In this case x_i denotes the value of state variable x just before the i^{th} circuit request, σ_i, is handled. We use this naming convention for state variables throughout the thesis. Finally, we use the following formatting conventions. Sections of the thesis that present algorithms will contain a paragraph marked by **Description:**. This paragraph describes the code of algorithm. Some sections also contain a paragraph marked by **Restrictions:**. This paragraph will summarize any special restrictions that apply to the algorithm presented in that section.

IAAP(s_i, d_i, r_i):
  for all $\tau, e \in E: \; c(e) = r_i(\mu^{u(e)} - 1)$;
  if there exists a path p in G from s_i to d_i such that
    $\sum_{e \in p} c(e) \leq \varrho$ and $u(e) + r_i/b(e) < 1$ for all $e \in p$
  then <u>route</u> the requested virtual circuit on p, and set:
    for all $e \in E$,
      $u(e) = u(e) + \frac{r_i}{b(e)}$ **if** $e \in p$;
  else <u>reject</u> the requested virtual circuit.

Figure 4-1: The IAAP admission control algorithm.

Description: We now explain the code in Figure 4-1. The admission control and routing decision is based on the utilization of the network links. The utilization of link e as seen by the algorithm when routing the j^{th} circuit is defined as follows (cf. Definition 2.3.1):

$$u_j(e) = \frac{1}{b(e)} \sum_{\substack{1 \leq i < j \,| \\ \rho_i \neq \perp, e \in \rho_i}} r_i.$$

Based on the utilization, the algorithm computes the *exponential cost* of the links. The exponential cost of a link is a cost function that is exponential in the utilization of the links. In particular, the cost of link e as seen by the algorithm when routing the i^{th} circuit is defined by

$$c_i(e) = r_i(\mu^{u_i(e)} - 1),$$

where μ is a constant. (The constant μ used here is the same as the constant μ used in Definition 4.2.1. In the definition, μ is used to restrict the size of r_i. This restriction on r_i is needed by the proofs in [AAP93, Plo95].) If there exists a path p in the network such that the cost of the path, $\sum_{e \in p} c_i(e)$ is no greater than ϱ, where ϱ is a parameter, then the request is accepted along path p. The final step of the algorithm is to update the state.

Let d be the number of links in the longest simple path. In [Plo95] they show that choosing $\mu = 2dr_{max}/r_{min} + 1$ and $\varrho = dr_{max}$ guarantees a competitive ratio of $O(\log \mu) = O(\log(2dr_{max}/r_{min} + 1))$ when r_i is restricted such that $r_i \leq \min_e\{b(e)/\log \mu\}$ and $r_i \in [r_{min}, r_{max}]$. We summarize this result in the following theorem.

Theorem 4.2.2 *Let $\mathcal{P}$ be the restricted bandwidth admission control and routing problem for* *general topology networks. Let P measure the number of accepted requests. Then, the* IAAP *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(\text{IAAP})$, of $O(\log \mu)$.*

4.3 Hierarchical Backbone Networks

Consider a hierarchical backbone network. Assume that d is the number of links of the longest simple path in any access network and that there exists an algorithm A_b that achieves a competitive ratio of K on the backbone network when all requests have the same bandwidth. This sections shows that an algorithm that uses greedy admission control on the access networks and algorithm A_b on the backbone network has a competitive ratio of $O(d+K)$. In Section 4.4 we use this result to achieve an $O(\log\log n)$ competitive ratio for the size n fat-tree. In Chapter 8, we argue that the basic approach of this section can be used for many important modern networks like the Internet and the telephone system. The performance function used in this section measures the number of accepted virtual circuit requests

4.3.1 Algorithm

Restrictions: We consider the admission control and routing problem on a hierarchical backbone network G with the additional restriction that all circuit requests have the same bandwidth.

Let the first request, σ_0, be (G, b) where G is a hierarchical backbone network consisting of a backbone network G^b and access networks G_i^a. Let d be the number of links in the longest simple path of any access network. To describe our algorithm, A_G, we require some notation. Consider a node v such that v is in access network G_i^a. Then ACCESS(v) is the node that G_i^a has in common with G^b. Furthermore, let A_{G^b} be an algorithm that has a competitive ratio of K for the admission control and routing problem on network G^b where every request has the same bandwidth. The A_G algorithm uses the A_{G^b} algorithm as a subroutine. We classify requests as *access* requests and *backbone* requests. An access request is one where the source and destination are both in the same access network. All other requests are backbone requests. We denote by GREEDY$_G$ an arbitrary randomized greedy on-line admission control and routing algorithm for network G. Let p_1 and p_2 be two paths such that the last node of p_1 is the same as the first node of p_2. Then $p_1|p_2$ is the concatenation of p_1 and p_2.

Claim 4.3.1 *Let G be a hierarchical backbone network consisting of a backbone network G^b and access networks G_i^a. Then:*

1. *for any access request (s_j, d_j, r_j), any simple path between s_j and d_j uses only links in G_i^a for some i.*
2. *for any backbone request (s_j, d_j, r_j), any path between s_j and d_j uses at least one link in G^b.*
3. *for any backbone request (s_j, d_j, r_j), where $s_i, d_i \in G^b$, any simple path between s_j and d_j uses only links in G^b.*

Proof. 1 and 3 follow from the fact that all requests must use simple paths and the fact that no simple path between two different nodes in G^b passes through any node in any access network G_i^a (cf. Lemma 3.7.2). 2 follows from the fact that G_i^a and G_j^a are disjoint when $j \neq i$. ∎

Description: Consider a backbone request (s_i, d_i). (We omit the bandwidth r_i since all requests are assumed to request the same bandwidth.) If both s_i and d_i are in the backbone network, we let the A_{G^b} algorithm make the admission control and routing decision. If s_i is in an access network, we use a greedy admission control strategy to connect s_i to $\text{ACCESS}(s_i)$. Similarly, if d_i is in an access network, we use a greedy admission control strategy to connect d_i to the backbone network. Then, we use the A_{G^b} algorithm to make the admission control and routing decision within the backbone network. Finally, for access requests, we use greedy admission control. The code is given in Figure 4-2. We say that a backbone request is *offered* to the A_{G^b} algorithm, if the request is not rejected by the greedy algorithms used for any parts of the path that are in access networks.

4.3.2 Analysis

Consider any request sequence σ for the hierarchical backbone network G. Let ρ be any corresponding result sequence. (There can be many such result sequences since the $\text{GREEDY}_{G_i^a}$ algorithm used in the access networks may be randomized.) For the analysis we define the

$A_G(s_i, d_i)$:
  $p = \perp$;
  case $s_i, d_i \in G^b$:
    $p = A_{G^b}(s_i, d_i)$;
  case $s_i \in G^a_j$, $d_i \notin G^a_j$ and $d_i \in G^b$:
    $p_1 = \text{GREEDY}_{G^a_j}(s_i, \text{ACCESS}(s_i))$;
    if $p_1 \neq \perp$ **then** $p_2 = A_{G^b}(\text{ACCESS}(s_i), d_i)$;
      if $p_2 \neq \perp$ **then** $p = p_1|p_2$;
  case $s_i \in G^b$, $s_i \notin G^a_j$ and $d_i \in G^a_j$:
    $p_1 = \text{GREEDY}_{G^a_j}(\text{ACCESS}(d_i), d_i)$;
    if $p_1 \neq \perp$ **then** $p_2 = A_{G^b}(s_i, \text{ACCESS}(d_i))$;
      if $p_2 \neq \perp$ **then** $p = p_1|p_2$;
  case $s_i \in G^a_j$ and $d_i \in G^a_k$ with $j \neq i$:
    $p_1 = \text{GREEDY}_{G^a_j}(s_i, \text{ACCESS}(s_i))$;
    if $p_1 \neq \perp$ **then** $p_2 = \text{GREEDY}_{G^a_k}(\text{ACCESS}(d_i), d_i)$;
      if $p_2 \neq \perp$ **then** $p_3 = A_{G^b}(\text{ACCESS}(s_i), \text{ACCESS}(d_i))$;
        if $p_3 \neq \perp$ **then** $p = p_1|p_3|p_2$;
  case $s_i, d_i \in G^a_j$:
    $p = \text{GREEDY}_{G^a_j}(s_i, d_i)$;
  endcase
  if $p \neq \perp$ **then** route the requested virtual circuit on path p.
  else reject the requested virtual circuit.

Figure 4-2: The admission control algorithm for a hierarchical backbone network.

following quantities with respect to σ and ρ. Let σ^f be the subsequence of σ that consists of σ_0 and all of the backbone requests that are offered to the A_{G^b} algorithm by the A_G algorithm. A backbone request might not be offered to the A_{G^b} algorithm due to capacity constraints in the access networks. (See code in Figure 4-2.) Let $\sigma^{\neg f}$ be the subsequence of σ that consists of σ_0 and all requests not included in σ^f. Let σ^{f,G^b} be the sequence that consists of all requests in σ^f, such that source s_i is replaced by $\text{ACCESS}(s_i)$ if $s_i \notin G^b$ and destination d_i is replaced by $\text{ACCESS}(d_i)$ if $d_i \notin G^b$. Furthermore, let $\sigma_0^{f,G^b} = (G^b, b)$. Let $copt_f$ be the number of requests from σ^f accepted by the optimal off-line algorithm. Similarly, let $copt_{\neg f}$ be the number of requests from $\sigma^{\neg f}$ accepted by the optimal off-line algorithm. By definition, $P_o(\sigma) = copt_f + copt_{\neg f}$. Let *calg* be the number of requests accepted by our A_G algorithm. By definition, $P(\sigma, \rho) = calg$.

Lemma 4.3.2

$$calg \geq \frac{copt_{\neg f}}{2d+1}.$$

Proof. Since all requests have the same bandwidth requirement, we can define for each link e a number k_e which represents the maximum number of requests that can use link e without violating the capacity constraints. ($k_e = \lfloor r/b(e) \rfloor$, where r is the bandwidth requirement of each request.)

Let E' be the set of access links on which A_G routes k_e requests for request sequence σ and result sequence ρ. Since each request uses at most $2d$ access links, $calg \geq \frac{1}{2d} \sum_{e \in E'} k_e$. Let R be the set of requests in $\sigma^{\neg f}$ that are rejected by A_G and accepted by the optimal off-line algorithm. Consider a request in R. Since the A_G algorithm uses greedy admission control for the access networks there must exists some access network link e on the path used by the optimal off-line algorithm such that A_G routes k_e requests on link e. Thus, every request in R must use a link in E'. Since the optimal off-line algorithm must meet the capacity constraints, it must be the case that $|R| \leq \sum_{e \in E'} k_e$. Using the fact that $calg \geq \frac{1}{2d} \sum_{e \in E'} k_e$, we conclude that $2d(calg) \geq |R|$.

By definition $|R| + calg \geq copt_{\neg f}$. Therefore, $(2d+1)calg \geq copt_{\neg f}$. The lemma follows. ■

Lemma 4.3.3

$$calg \geq \frac{P_o(\sigma^{f,G^b})}{K}$$

Proof. The request sequence σ^{f,G^b} is a request sequence for the admission control and routing problem for graph G^b, where the bandwidth of each request is the same. The lemma now follows directly from the facts that the request sequence σ^{f,G^b} is offered to the A_{G^b} algorithm, that the A_{G^b} algorithm has a competitive ratio of K, and that $calg$ includes all of the backbone requests accepted by A_G. ■

Lemma 4.3.4

$$copt_f \leq P_o(\sigma^{f,G^b}).$$

Proof. Consider the routing decisions made by the optimal off-line algorithm for the requests in σ^f. Since G^b is connected and there is no simple path between two different nodes in G^b that uses the access networks, the admission control and routing decisions made for σ^f can be used for σ^{f,G^b}. ■

Lemma 4.3.5

$$\frac{P_o(\sigma)}{P(\sigma,\rho)} \leq O(\log\log n).$$

Proof.

$$\begin{aligned} P_o(\sigma) &= copt_{\neg f} + copt_f \\ &\leq (2d+1)(calg) + P_o(\sigma^{f,G^b}) && \text{(by Lemmas 4.3.2 and 4.3.4)} \\ &\leq (2d+1)(calg) + K(calg) && \text{(by Lemma 4.3.3)} \\ &= O(d+K)P(\sigma,\rho). \end{aligned}$$

■

We summarize the results of this section in the following Theorem.

Theorem 4.3.6 *Consider the admission control and routing problem $\mathcal{P}$ on a hierarchical backbone network G where the bandwidth requirement of each request is the same. Decompose G into backbone network G^b and access networks G_i^a such that the number of links in the longest simple path in any access network is at most d. Let A_{G^b} achieve a competitive ratio of K for the admission control and routing problem on G^b when the bandwidth requirement of each request is the same. Then, A_G achieves a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(A_G)$, of $O(d+k)$.*

Proof. The theorem follows directly from Lemma 4.3.5. ■

4.3.3 Applications and Extensions

The $O(d+K)$ competitive ratio of Theorem 4.3.6 applies to hierarchical backbone networks where the number of links in the longest simple path in any access network is at most d. Theorem 4.3.6 can be generalized as follows. Consider the admission control and routing problem for any hierarchical backbone network. Now restrict the problem so that any request may not use any path that includes a total of more than d links in any given access network. Then, the A_G algorithm also achieves a $O(d+K)$ competitive ratio. The proof proceeds exactly as the proof for Theorem 4.3.6. This generalization means that the topological restrictions in Theorem 4.3.6 can be replaced by path length restrictions in the actual admission control and routing problem.

The A_G algorithm can be used to extend the admission control and routing algorithm of [AAP93] to networks where some regions of the network do not meet the bandwidth restrictions imposed by [AAP93]. In particular, consider some hierarchical backbone network G that can be decomposed into a backbone network G^b that meets the bandwidth restrictions of [AAP93] and several access networks G_i^a that may not meet the bandwidth restrictions of [AAP93]. Let n be the number of links in the longest simple path in G^b and let d be the number of links in the longest simple path in any G_i^a. (Alternatively, we could restrict the problem so that any request may not use any path that includes a total of more than d links in any given access network.) Then Theorem 4.2.2 and Theorem 4.3.6 imply that A_G can achieve a competitive ratio of $O(d + \log n)$ for the admission control and routing problem on G. We use this approach for the fat-tree algorithm in Section 4.4.

4.4 Fat-Trees

This section presents our algorithm for admission control and routing on the unit bandwidth size n fat-tree. The structure of the size n fat-tree is explained in Section 3.5. Our non-greedy algorithm achieves a competitive ratio of $O(\log \log n)$. Thus, our algorithm matches the lower bound in Lemma 3.5.4. Since the lower bound for any greedy admission control algorithm is $\Omega(\log n)$ (cf. Lemma 3.5.4), we conclude that non-greedy admission control leads to better performance for fat-trees. The performance function used in this section measures the number of accepted virtual circuit requests

4.4.1 Algorithm

Restrictions: Since we consider the unit bandwidth size n fat-tree, the bandwidth of each request is restricted to be 1.

Our algorithm is based on our approach to hierarchical backbone networks. In particular, a fat-tree can be seen as a hierarchical backbone network. Let T be the size n fat-tree. Define the access networks to be the $2^{2\log n - 8\log\log\log n}$ subtrees rooted at the nodes of height $8\log\log\log n$. Define the backbone network to be the network consisting of the nodes with height $8\log\log\log n$ and above.

Description: We consider two cases: $n < 16$ and $n \geq 16$. When $n < 16$ we use greedy admission control. For $n \geq 16$, we use the algorithm in Figure 4-2, where the IAAP algorithm takes the place of the A_{G^b} algorithm. We call our algorithm FATTREE.

4.4.2 Analysis

We first consider the case where $n \leq 16$.

Lemma 4.4.1 *Let $n < 16$. For the admission control and routing problem $\mathcal{P}$ on a unit bandwidth size n fat-tree, the* FATTREE *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}$(*FATTREE*), of $O(\log\log n)$.*

Proof. When $n < 16$ a greedy admission control algorithm trivially leads to a constant competitive ratio. In particular, any request accepted by the greedy algorithm can block at most 16 of the requests accepted by the optimal off-line algorithm. ∎

Now consider the general case. The analysis of the general case relies primarily on Theorem 4.3.6.

Theorem 4.4.2 *For the admission control and routing problem $\mathcal{P}$ on a unit bandwidth size n fat-tree, the* FATTREE *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}$(*FATTREE*), of $O(\log\log n)$.*

Proof. We consider two case: $n < 16$ and $n \geq 16$. Lemma 4.4.1 proves the theorem for $n < 16$. Next consider $n \geq 16$.

Since we are using the unit bandwidth version of the admission control and routing problem, each request has a bandwidth requirement of 1. Furthermore, FATTREE divides the size n fat-tree into backbone network G^b and access networks G_i^a such that the number of links in the longest simple path of any access network is at most $16 \log\log\log n$.

The capacity of the lowest capacity link in G^b is $(\log\log n)^4$. Furthermore, the length of the longest simple path in G^b is less than $2\log\log n$ and the capacity of each request is 1. Now $\frac{(\log\log n)^4}{\log(2\log n+1)} \geq 1$ when $n \geq 16$. Now, Theorem 4.2.2 implies that the IAAP algorithm achieves a competitive ratio of $O(\log\log n)$ for the admission control and routing problem on G^b, when the bandwidth of each request is restricted to be 1.

Now, by Theorem 4.3.6, we see that the FATTREE algorithm has a competitive ratio of $O(\log\log n + \log\log\log n)$ for the admission control and routing problem $\mathcal{P}$ on a unit bandwidth size n fat-tree. The theorem follows. ∎

4.5 Trees

This section presents our algorithm for the unit capacity admission control and routing problem on trees. Our randomized non-greedy algorithm achieves an oblivious competitive ratio of $O(\log d)$ for any tree of diameter d. Thus, our algorithm matches the lower bound in Lemma 3.3.3. We note, from Proposition 3.3.6 and Lemma 3.3.3, that the lower bound for any greedy or deterministic admission control algorithm on diameter d unit capacity trees is $\Omega(d)$.

4.5.1 Preliminaries

Consider a tree T consisting of more than one link. We distinguish an arbitrary non-leaf node r of T and call it the *root* of T. This induces, for every vertex $v \neq r$ a unique parent, which we denote by $par(v)$. Consider a pair u, v of leaf nodes. Denote their least common ancestor in the rooted tree by $\textsc{lca}(u, v)$. Denote by $p(u, v)$ the unique simple path connecting u and v. Consider a node pair u, v such that node u is to the "left" of node v in a pictorial representation of the tree. In this case, we refer to u as the left node and v as the right node. Denote by $\textsc{tl}(u, v)$ the top left link of $p(u, v)$, i.e., the first link on the path $p(\textsc{lca}(u, v), u)$. Similarly, denote by $\textsc{tr}(u, v)$ the top right link of $p(u, v)$, i.e., the first link on the path $p(\textsc{lca}(u, v), v)$. Let p and p' be two paths in T. We say that p and p' *intersect* if they share a link.

For simplicity we describe the randomized on-line algorithm of this section as making random choices on a per request basis rather than choosing initially from among a set of deterministic algorithms. It is straightforward to translate the description of the algorithms used in this section to the formal model in Chapter 2. In particular, we can imagine the algorithm making all its random choices just before the first request arrives. By making all its random choices, the algorithm is effectively choosing a deterministic algorithm. Then, whenever the algorithm needs to make a random decision, it uses one of the previously made "random" choices to determine that decision. For example, consider an algorithm that must flip a random coin for each request. One can think of the algorithm as flipping many random coins before the first request and storing the results of those flips. Whenever it needs to flip a random coin for a request, it simply uses one of the stored results that it has not used before. Note that the strategy for choosing one of the unused stored results must be deterministic.

4.5.2 Algorithm

Restrictions: We describe our algorithm, called TREE, and prove the competitive ratio of our algorithm for a restricted version of the unit capacity admission control and routing problem on trees. In particular, we assume that the sources and destinations for all requests are leaf nodes. In Section 4.5.4 we will show how to extend our algorithm and the competitive analysis to the general case where the sources and destinations need not be leaf nodes.

Consider any request sequence σ. Let $\sigma_0 = (T, b)$ be the first request. If T consists of a single link, then our algorithm accepts the first request.

Lemma 4.5.1 *When T consists of a single link, the* TREE *algorithm has a competitive ratio of* 1.

Proof. The TREE algorithm accepts the first request. The optimal off-line algorithm accepts at most one request. As a result, we achieve a competitive ratio of 1. ∎

For the remaining discussion we assume that T contains more than one link.

Let $\sigma_i = (s_i, d_i, 1)$ be a request for $i > 0$. Each such request is associated with a unique path, $p(\sigma_i)$ in T. To simplify the discussion, assume that for any request $\sigma_i = (s_i, d_i, 1)$ node s_i is to the "left" of node d_i in a pictorial representation of the tree. (The assumption can be made without loss of generality since the tree is undirected.) Let S be a set of requests from σ. Then $<S, p(\sigma_i)>$ is the set of requests $\sigma_j \in S$ such that $p(\sigma_j)$ intersects $p(\sigma_i)$. Since we are considering the unit capacity version of the admission control and routing problem, the third term of any request is 1. Thus, in this section, we will denote a request simply by $\sigma_i = (s_i, d_i)$.

To describe the TREE algorithm, we introduce the concepts of a *roadblock* and a *special roadblock*. In response to a request, our algorithm may place roadblocks and special roadblocks on links of the tree. The existence of a roadblock or a special roadblock on a link blocks future requests whose paths use that link, and causes them to be rejected. The current state of the network is described by the $u_i(e)$ function, which describes the state of the links just before the i^{th} request is handled. A link can be in one of four states. If $u_i(e) = full$ the link is being used by a previously accepted request. If $u_i(e) = bk$ the link has a roadblock. If $u_i(e) = sbk$ the link has a special roadblock. Finally, if $u_i(e) = avail$ the link is available.

TREE(s_i, d_i):
  if $u(e) = avail$ for all $e \in p(s_i, d_i)$ **then**
    $\boxed{C = C - <C, p(s_i, d_i)> - \bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i));}$
    if RAND$(0,1) == 0$ **then** route the requested virtual circuit on $p(s_i, d_i)$, and set:
      for all $e \in p(s_i, d_i)$, $u(e) = full$;
    else reject the requested virtual circuit.
      $\ell_i = \text{RAND}(1, \log d)$;
      number links in $p(s_i, d_i)$ from 1 to $|p(s_i, d_i)|$ starting with $(s_i, par(s_i))$;
      for all $j \in (0, d/2^{\ell_i})$, $u(e_{j2^{\ell_i}}) = bk$;
      $u(e) = sbk$ **if** $e \in \text{TL}(\sigma_i) \cup \text{TR}(\sigma_i)$;
      $\boxed{C = C \cup \{\sigma_j \mid tok(\sigma_j) = (\sigma_i, z) \text{ for some } z\};}$
      $\boxed{C = C - sp(\sigma_i);}$
  else reject the requested virtual circuit.

Figure 4-3: The admission control algorithm for diameter d trees.

Description: Consider a particular virtual circuit request $\sigma_i = (s_i, d_i)$. If all links e in the path $p(s_i, d_i)$ are available, i.e., $u_i(e) = avail$, then σ_i is considered a *candidate*. Otherwise, σ_i is rejected. If the request becomes a candidate, accept it with probability $1/2$. Otherwise, reject it and place roadblocks as follows. Pick a random integer ℓ_i uniformly in $[1, \log d]$, where d is the diameter of tree T. Consider the path $p(s_i, d_i)$. Number the links along the path from s_i to d_i with 1, 2, ... Place a roadblock on links numbered $j2^{\ell_i}$ for all $0 < j < d/2^{\ell_i}$. The roadblocks partition $p(s_i, d_i)$ into *segments* of equal length (except, perhaps, for the last segment). Also, place special roadblocks on the links TL(σ_i) and TR(σ_i). The code for our algorithm is given in Figure 4-3. The code enclosed in boxes is *accounting code*. It does not influence the admission control decisions of the algorithm. Rather, it is used for the competitive analysis. We will explain the notation used for the accounting code in Section 4.5.3. The function RAND(x, y) picks a random number uniformly in $[x, y]$.

The design of the TREE algorithm is based on the following considerations. Consider a greedy admission control strategy on a diameter d tree. Let the first virtual circuit request, σ_1, to such a strategy be a request, such that the path $p(\sigma_1)$ has length d. A greedy admission control strategy would accept this request. If this initial request is followed by d requests between all of the neighboring nodes along the path $p(\sigma_1)$, the greedy algorithm exhibits a competitive ratio of $\Omega(d)$. To avoid this problem non-greedy admission control is needed. However, if the request

σ_1 is rejected with certainty by a non-greedy admission control algorithm, the competitive ratio is again poor. In fact it is $1/\epsilon$. Just consider the request sequence consisting of only the request σ_1. Thus, the initial request σ_1 must be accepted probabilistically. (The TREE algorithm accepts it with probability 1/2.) However, a non-greedy algorithm that accepts each long request, such as σ_1, independently with some probability q will still not exhibit a good competitive ratio. Consider a request sequence σ that repeats the request σ_1 so often that the algorithm that accepts each long request with probability q will accept one of the long requests in σ with high probability. Now extend σ with d requests between all of the neighboring nodes along the path $p(\sigma_1)$. Thus, we again have a competitive ratio of $\Omega(d)$. The roadblocks avoid this problem. Once the first long request is rejected, the roadblocks ensure that no request that shares sufficiently many of its links with the rejected request will ever be accepted.

This discussion explains the reason for using roadblocks, but does not provide a justification for the special roadblocks. The justification for the special roadblocks is tied closely to the analysis. Hence we give this justification in Section 4.5.3.

4.5.3 Analysis

Before presenting the details of the analysis, we give an overview. The analysis is anchored around a bookkeeping device called a *token*. We define a token formally later in this section. For now, just think of a token as an arbitrary object that one request can assign to another request. The assignment of tokens does not affect the admission control decisions of the algorithm. Let C_0 be the requests accepted by the optimal off-line algorithm for request sequence σ. (C_0 is the initial value of the accounting variable C.) Throughout an execution of the TREE algorithm, tokens are assigned to requests in C_0. For any request sequence σ, our analysis will show two things.

1. The expected number of requests in C_0 that receive a token is at least $\frac{1}{2(\log d)}$ times the expected number of requests in C_0 that are not candidates.

2. The number of candidate requests is at least 1/7 times the number of requests in C_0 that receive tokens.

Thus, the expected number of candidates is at least $\frac{1}{O(\log d)}$ times the number of requests in C_0. We will use this fact, combined with the fact that each candidate request is accepted with probability $1/2$, to show that the expected number of accepted requests is at least $\frac{1}{O(\log d)}$ times the number of requests in C_0. The $O(\log d)$ oblivious competitive ratio will then follow.

To describe why the expected number of requests in C_0 that receive a token is at least $\frac{1}{2(\log d)}$ times the expected number of requests in C_0 that are not candidates we given an informal description of how tokens are assigned. (The formal description is presented with the actual proof.) Let σ_j be a request in C_0 that is not a candidate. Then there must exist a candidate request that is handled before σ_j and whose path overlaps the path of σ_j. Let σ_i be the first such request to be handled. Then σ_i might assign a token to σ_j. If σ_i is rejected and ℓ_i is the random number picked by σ_i to space its roadblocks, σ_i places roadblocks on every 2^{ℓ_i} links of its path. The roadblocks break the path of σ_i into segments. If the path of σ_j intersects the middle link of some segment, but does not intersect any link on which σ_i places a roadblock, σ_j receives a token from σ_i.

For any candidate request, there are $\log d$ possible spacings for the roadblocks. The densest spacing places a roadblock on every second link. The next spacing places a roadblock on every fourth link, etc. Finally, the least dense spacing places no roadblocks. Therefore, for any pair of requests σ_j and σ_i whose paths intersect, there can only be one spacing of roadblocks on the path of request σ_i such that the path of σ_j intersects the middle link of some segment, but does not intersect any link on which σ_i places a roadblock. Since each possible spacing is selected with probability $1/\log d$, a request σ_j in C_0 that is not a candidate receives a token from σ_i with probability $1/\log d$ if request σ_i is rejected, σ_i is the first of the candidates that are handled before σ_j, and the path of σ_i intersects the path of σ_j. Now the fact that the candidate σ_i is rejected with probability $1/2$ leads to the conclusion that the expected number of requests in C_0 that receive a token is at least $\frac{1}{2(\log d)}$ times the expected number of requests in C_0 that are not candidates.

The most difficult step in the proof is showing that the number of candidate requests is at least $1/7$ times the number of requests in C_0 that receive tokens. A straightforward approach to this step would be to prove that, for each execution of the TREE algorithm, the path of every candidate intersects the path of at most 7 requests in C_0 that receive tokens. Unfortunately, this is not true in general. However, if the path of a candidate intersects the paths of more than 7 requests in C_0 that receive tokens, then the paths of other candidates also intersect the paths of the requests with tokens. This suggests an amortized argument.

Our amortized argument is structured around subsets of C_0. In particular, the set C_i is a subset of C_0 that consists of requests that are still feasible just before request σ_i is handled. In other words, the requests in C_i are all handled after σ_{i-1} and their paths have not been blocked by previous requests, roadblocks, or special roadblocks. (C_i is the value of the accounting variable C just before request σ_i is handled. We give a formal definition of C_i when we present the actual proof.) Initially, all requests in C_0 are feasible so, $C_1 = C_0$. In constructing C_{i+1} from C_i, candidate request σ_i removes from C_i all requests whose paths intersect the path of σ_i. Furthermore, σ_i returns to C_i the requests from C_0 to which it assigns tokens. (Every request in C_0 can receive a token from only one candidate request.) Finally, requests whose paths are blocked by special roadblocks are removed from C_i. Our amortized argument now proceeds by showing the following for each execution of the TREE algorithm. We show that each candidate removes at most 7 requests with tokens from C_i while constructing C_{i+1}. Furthermore, we show that every request that receives a token, say from σ_i, is added to C_i in the construction of C_{i+1}. Finally, if σ_{k-1} is the last request in the request sequence, we show that $C_k = \emptyset$. Therefore, since for every execution of the TREE algorithm, every request that receives a token is added to C_i for some i, every candidate request σ_i removes at most 7 requests with tokens from C_i while constructing C_{i+1}, and $C_k = \emptyset$ when σ_{k-1} is the last request in the request sequence, we will be able to conclude that in every execution, the number of candidate requests is at least 1/7 the number of requests in C_0 that receive tokens.

Now we can explain the need for the special roadblocks. In particular, the special roadblocks are needed to show the fact that each candidate σ_i removes at most 7 requests with tokens from C_i while constructing C_{i+1}. Without the special roadblocks, there can be situations where σ_i removes more than 7 requests with tokens from C_i while constructing C_{i+1}. For example, consider Figure 4-4. Assume for the moment that the TREE algorithm does not use special roadblocks. Figure 4-4 shows the paths for the requests from the sequence $\sigma_1 \ldots \sigma_{2n+1}$. Let C_0 consist of the n requests $\sigma_{n+2} \ldots \sigma_{2n+1}$. Assume that each request σ_i for $i \in [1, n]$ assigns a token to σ_{i+n+1}. Thus, C_{n+1} consists of the n requests σ_{i+n+1} for $i \in [1, n]$ that have been assigned tokens. Assume further that σ_{n+1} is a candidate. In other words, the requests $\sigma_1 \ldots \sigma_n$ do not place any roadblocks on the path of σ_{n+1}. When constructing C_{n+2} from C_{n+1}, request σ_{n+1} must remove from C_{n+1} the requests whose paths intersect its path. Thus, if $n > 7$, request σ_{n+1} is a candidate that removes more than 7 requests with tokens from C_{n+1} while constructing C_{n+2}. However, recall that we want to prove that each candidate σ_i removes at most 7 requests with tokens from C_i while constructing C_{i+1}. Towards this end, the special roadblocks are used to avoid the situation described in Figure 4-4. In particular, the special roadblocks that are placed by the requests $\sigma_1 \ldots \sigma_{n-1}$ ensure that the request σ_{n+1} cannot be a candidate. In fact, the special roadblocks ensure that for every candidate σ_i the requests in C_i that have tokens and that are removed by σ_i must have received the tokens from a request whose path overlaps the top left hand link or top right hand link of the path of σ_i. (The request σ_n in Figure 4-4 is an example of such a request.) With this fact, we will be able to show that each candidate σ_i removes at most 7 requests with tokens from C_i while constructing C_{i+1}.

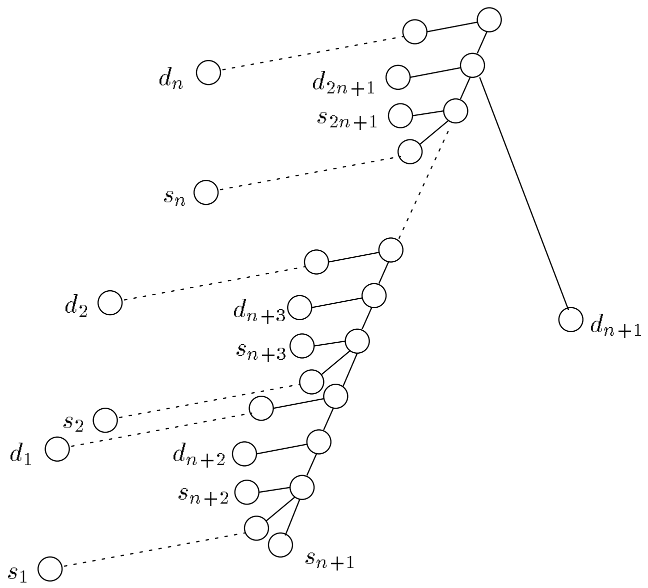

Figure 4-4: A situation where the special roadblocks are needed to ensure that each candidate σ_i removes at most 7 requests with tokens from C_i while constructing C_{i+1}.

Consider any execution of the TREE algorithm for request sequence $\sigma = \sigma_0\sigma_1 \ldots \sigma_{|\sigma|-1}$. Let ρ be the result sequence. We now start the formal proof with several definitions.

$\boldsymbol{C_0}$: The set of requests accepted by the optimal off-line algorithm. We call these requests the *optimal requests*. (Note that $P_o(\sigma) = |C_0|$). C_0 is the initial value of the accounting variable C.

$\boldsymbol{C^c}$: The set of requests that are candidates in the execution of TREE.

$\boldsymbol{C^f}$: The set of requests that are *free* in the execution of TREE. A request σ_i is *free* if $\sigma_i \in C_0$ and if there exists no candidate request $\sigma_j \in C^c$ such that $j < i$ and $p(\sigma_j)$ intersects $p(\sigma_i)$. The first condition states that the request is an optimal request. The second condition ensures that the request is a candidate before any request with an intersecting path is a candidate.

$\boldsymbol{C^{\neg f}}$: The set of requests that are *non-free* in the execution of TREE. A request σ_i is *non-free* if $\sigma_i \in C_0$ and $\sigma_i \notin C^f$

Based on the definitions of C^f and $C^{\neg f}$ it is easy to see that $C^f \cup C^{\neg f} = C_0$. Furthermore, C^f and $C^{\neg f}$ are disjoint.

Fact 4.5.2 *$C_0 = C^f \cup C^{\neg f}$.*

Next, we describe how tokens are distributed in the execution of TREE. The description requires some additional notation.

$\mathbf{level}(\boldsymbol{\sigma_j, \sigma_i})$: Consider two requests σ_j and σ_i such that $p(\sigma_j)$ and $p(\sigma_i)$ intersect. Number the links in $p(\sigma_j)$ from 1 to $|p(\sigma_j)|$ starting with $(s_j, par(s_j))$. Let L_ℓ be the set of links numbered $k2^\ell$ for all $k \in (0, d/2^\ell)$, where d is the diameter of the tree. The request σ_i has *level* ℓ with respect to request σ_j, i.e., $level(\sigma_j, \sigma_i) = \ell$, if $p(\sigma_i)$ includes no link in L_ℓ but does include a link in $L_{\ell-1}$. Informally, the request σ_i is not blocked by roadblocks spaced every 2^ℓ links on the path of σ_j, but is blocked by roadblocks spaced every $2^{\ell-1}$ links. (See code in Figure 4-3.)

$\mathbf{seg}(\boldsymbol{\sigma_j, \sigma_i})$: Consider requests σ_j, σ_i such that $\sigma_j \in C^c$, $j < i$, $\rho_j = \perp$ and $p(\sigma_j)$ intersects $p(\sigma_i)$. Let ℓ_j be the random number picked by σ_j to determine the spacing of the roadblocks on $p(\sigma_j)$. Then, assume that $level(\sigma_j, \sigma_i) \leq \ell_j$. In other words, σ_j is a candidate, σ_j is handled before request σ_i, request σ_j is rejected, the paths for the requests intersect, and σ_j does not place any roadblock on the path $p(\sigma_i)$. Number the segments created by the roadblocks on $p(\sigma_j)$ starting with the segment that includes s_j as segment number 1. Now $seg(\sigma_j, \sigma_i)$ is the number of the segment in which $p(\sigma_i)$ and $p(\sigma_j)$ intersect. This segment number is uniquely defined because σ_i does not include any of the links on which σ_j places roadblocks.

$\mathbf{first}(\boldsymbol{\sigma_j})$: Let $\sigma_j \in C^{\neg f}$. Then the set $H = \{\sigma_i \mid i < j, \sigma_i \in C^c,\ p(\sigma_i) \text{ intersects } p(\sigma_j)\}$ is non-empty. In other words, H contains all candidate requests that are handled before σ_j and whose paths intersects the path of σ_j. Let $first(\sigma_j) = \sigma_i$ if $\sigma_i \in H$ and $i \leq k$ for all $\sigma_k \in H$. In other words, $\mathit{first}(\sigma_j)$ is the first candidate request that is handled before σ_j and whose path intersects the path of σ_j.

Tokens are distributed by candidate requests to optimal requests. In particular, let σ_i be a candidate request that is rejected. I.e., $\sigma_i \in C^c$ and $\rho_i = \perp$. In this case, σ_i may distribute tokens to some of the optimal requests σ_j for which $\mathit{first}(\sigma_j) = \sigma_i$. Thus, any optimal request can receive a token from at most one candidate request. Whether or not an optimal request receives a token is indicated by the function *tok*.

tok($\boldsymbol{\sigma_j}$): Let σ_j be an optimal request such that $\sigma_j \in C^{\neg f}$. Let σ_i be a candidate request such that $\sigma_i = \mathit{first}(\sigma_j)$ and such that $\rho_i = \perp$. Let ℓ_i be the random number picked by σ_i to space its roadblocks. The optimal request σ_j receives a token from request σ_i in segment number z iff $\mathit{level}(\sigma_i, \sigma_j) = \ell_i$, and $\mathit{seg}(\sigma_i, \sigma_j) = z$. We denote the fact that σ_j receives a token from request σ_i in segment number z by setting $\mathit{tok}(\sigma_j) = (\sigma_i, z)$. Conversely, we denote the fact that σ_j does not receive a token in the execution of TREE by $\mathit{tok}(\sigma_j) = \perp$.

We now show that, for any request σ_i and segment number z, there exists at most one optimal request σ_j such that $\mathit{tok}(\sigma_j) = (\sigma_i, z)$. Consider segment $z = \mathit{seg}(\sigma_i, \sigma_j)$. Since $\mathit{level}(\sigma_i, \sigma_j) = \ell_i$, where ℓ_i is the random number picked by σ_i to determine the spacing of the roadblocks, the path $p(\sigma_j)$ does not include the links with roadblocks that form the boundary of segment z. However, the fact that $\mathit{level}(\sigma_i, \sigma_j) = \ell_i$ implies that $p(\sigma_j)$ would intersect a link with a roadblock if σ_i had spaced the roadblocks half as many links apart. Thus, $p(\sigma_j)$ must include the middle link of segment z. The capacity constraint implies that there can be at most one optimal request that intersects any particular link. As a result, there can be at most one optimal request whose path includes the middle link of segment z.

Finally, we note that the optimal request σ_j can be in one of two states after request σ_i is handled. Either it is still *feasible*, i.e., $u_{i+1}(e) = \mathit{avail}$ for all links e on path $p(\sigma_j)$, or it is blocked by a special roadblock, i.e., there exists a link e on path $p(\sigma_j)$ such that $u_{i+1}(e) = \mathit{sbk}$. We prove this fact in the following lemma.

Lemma 4.5.3 *If $\mathit{tok}(\sigma_j) = (\sigma_i, z)$ then for all links e in $p(\sigma_j)$, $u_{i+1}(e) = \mathit{avail}$ or $u_{i+1}(e) = \mathit{sbk}$.*

Proof. Since $\sigma_i = \mathit{first}(\sigma_j)$, σ_i is the first candidate request such that the path $p(\sigma_i)$ intersects the path $p(\sigma_j)$. Thus, $u_i(e) = \mathit{avail}$ for all links e in $p(\sigma_j)$. If $\mathit{tok}(\sigma_j) = (\sigma_i, z)$, then $\rho_i = \perp$. Thus, the only way for $u_{i+1}(e) \neq \mathit{avail}$ and $u_{i+1}(e) \neq \mathit{sbk}$ for some link e in $p(\sigma_j)$ is by a roadblock placed by σ_i. However, since $tok(\sigma_j) = (\sigma_i, z)$, it is the case that $level(\sigma_i, \sigma_j) = \ell_i$. Therefore, σ_i places no roadblock on the path $p(\sigma_j)$. ∎

$\mathbf{tok}^{-1}(\boldsymbol{\sigma_i, z})$: Let $tok(\sigma_j) = (\sigma_i, z)$. Then $tok^{-1}(\sigma_i, z) = \{\sigma_j\}$. If there exists no request σ_j such that $tok(\sigma_j) = (\sigma_i, z)$ then $tok^{-1}(\sigma_i, z) = \emptyset$. The function tok^{-1} provides a way to determine to which optimal request, if any, a request σ_i gave a token in segment number z. From our discussion of the function *tok* we conclude that $|tok^{-1}(\sigma_i, z)| \leq 1$ for all σ_i and z.

$\boldsymbol{C^t}$: The set of requests σ_j for which $tok(\sigma_j) \neq \perp$. In other words, the set of optimal requests that receive tokens.

Let S be any subset of C_0. Then, $tok(S) = \{\sigma_i \mid \sigma_i \in S, tok(\sigma_i) \neq \perp\}$. To describe the accounting code in Figure 4-3 that modifies C we need to define two more sets (S_i and $sp(\sigma_i)$).

$\boldsymbol{S_i}$: Let $\sigma_i \in C^c$. Then, $S_i = \{\sigma_j \mid j < i, \sigma_j \in C^c, \rho_i = \perp, p(\sigma_j) \text{ intersects } p(\sigma_i)\}$. In other words, for any request σ_i that is a candidate, S_i is the set of all previously rejected candidate requests whose paths intersect the path of σ_i.

$\mathbf{sp}(\boldsymbol{\sigma_i})$: The set of optimal requests whose paths include the links $\mathrm{TL}(\sigma_i)$ and $\mathrm{TR}(\sigma_i)$. The capacity constraint implies that the set $sp(\sigma_i)$ has cardinality of at most two.

$\boldsymbol{C_i}$: A subset of C_0. Each request $\sigma_j \in C_i$ is *feasible* just before request σ_i is handled in the sense that $j \geq i$ and, for all the links e in the path of σ_j, $u_i(e) = avail$. (See Lemma 4.5.4.) C_i is defined by the accounting code of Figure 4-3. We now explain that code. Since the first request just gives the topology, $C_1 = C_0$. So, consider C_i for $i > 1$. If the request σ_i is a not a candidate, then $C_{i+1} = C_i$. If σ_i is a candidate, then we remove from C_i all of the requests in C_i whose paths intersect the path $p(\sigma_i)$. Furthermore, for any previously rejected candidate σ_j for which $p(\sigma_i)$ and $p(\sigma_j)$ intersect, consider the segment of $p(\sigma_j)$ in which the intersection occurs. This segment is given by $seg(\sigma_j, \sigma_i)$. If there is an optimal request in C_i that has a token $(\sigma_j, seg(\sigma_j, \sigma_i))$ associated with it, i.e., it received the token from σ_j in segment $seg(\sigma_j, \sigma_i)$, we remove the optimal request from C_i. Such a request is denoted by $tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i))$. We provide an example of such a request in Figure 4-5. Finally, we make some additional changes to C_{i+1}. In particular, we add the requests that received tokens from σ_i and remove any requests that are blocked by special roadblocks placed by σ_i.

Lemma 4.5.4 *Consider request sequence $\sigma_0 \ldots \sigma_{k-1}$. Then, $C_k = \emptyset$.*

Proof. The proof proceeds by showing that all requests in C_i are feasible.

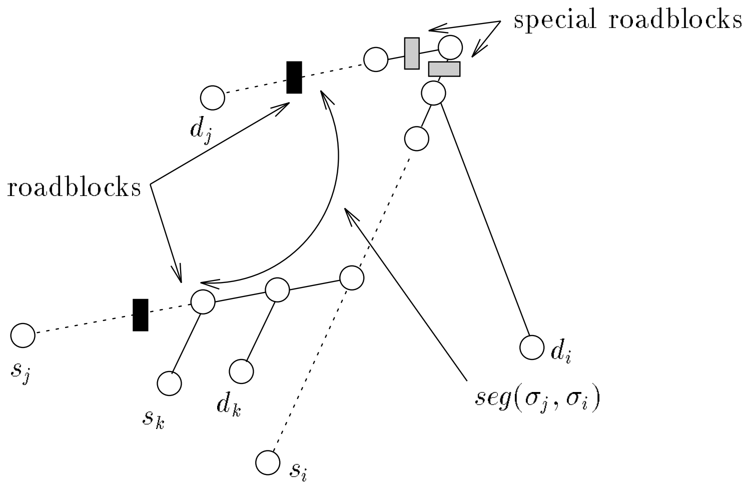

Figure 4-5: A set of requests σ_i, σ_j, and σ_k such that $\sigma_k = tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i))$. The segment $seg(\sigma_j, \sigma_i)$ is demarcated by the two roadblocks shown in the figure.

We proceed by induction. First consider the base case. All requests in $C_0 = C_1$ are feasible since C_0 consists of the optimal requests. For the inductive step assume that all requests in C_i are feasible for $i \geq 1$. We remove all optimal requests from C_i whose paths intersect the path $p(\sigma_i)$. (Note, we also remove σ_i if it is an element of C_i.) Furthermore, we only return to C_i optimal requests σ_j for which $tok(\sigma_j) = (\sigma_i, z)$ for some z. By Lemma 4.5.3, for all optimal requests σ_j for which $tok(\sigma_j) = (\sigma_i, z)$ for some z, $u_{i+1}(e) = avail$ or $u_{i+1}(e) = sbk$ for all links e in $p(\sigma_j)$. The optimal requests for which $u_{i+1}(e) = sbk$ for some link e are removed from C_i. (See the code in Figure 4-3.) Thus, only the requests σ_j where $u_{i+1}(e) = avail$ for all links e in $p(\sigma_j)$ remain in C_i. By definition, these requests are still feasible. ∎

The following lemma bounds the number of requests in $C_i \bigcap \left(\bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right)$. (This is a subset of the requests removed from C_i by candidate request σ_i.)

Lemma 4.5.5 *Consider any execution of the* TREE *algorithm for request sequence* $\sigma_0 \ldots \sigma_{k-1}$. *Then, for all* $0 < i < k$ *such that* $\sigma_i \in C^c$,

$$\left| C_i \bigcap \left(\bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right) \right| \leq 2.$$

Proof. When $S_i = \emptyset$, the lemma is vacuously true. So, consider the case $S_i \neq \emptyset$. Let $x = \text{LCA}(s_i, d_i)$. We divide S_i into two sets, S_i^l and S_i^r. The set S_i^l includes any $\sigma_j \in S_i$ such that its path intersects the left path of σ_i, i.e., it intersects $p(s_i, x)$. Similarly, the set S_i^r includes any $\sigma_j \in S_i$ such that its path intersects the right path of σ_i, i.e., it intersects $p(x, d_i)$. First consider the case where $S_i^l \neq \emptyset$.

Consider any $\sigma_j \in S_i^l$. Due to the special roadblocks, it must be the case that $p(\sigma_j)$ includes the top left link, $\textsc{tl}(\sigma_i)$, of $p(\sigma_i)$. (See Figure 4-4.)

Let σ_l be the highest index request in S_i^l. Since the path of each request in S_i^l includes the link $\textsc{tl}(\sigma_i)$, it must be the case that the paths of the requests in S_i^l also intersect the path of σ_l. Furthermore, they must intersect in the same segment. Specifically,

$$S_i^l \subseteq S_l \cup \{\sigma_l\},$$
$$seg(\sigma_j, \sigma_i) = seg(\sigma_j, \sigma_l) \text{ for all } \sigma_j \in S_i^l - \{\sigma_l\}.$$

As a consequence,

$$\bigcup_{\sigma_j \in S_i^l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \subseteq \bigcup_{\sigma_j \in S_l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_l)) \cup tok^{-1}(\sigma_l, seg(\sigma_l, \sigma_i)).$$

The requests in $\bigcup_{\sigma_j \in S_l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_l))$ are already removed from C_l by request σ_l. Furthermore, we show that requests in $\bigcup_{\sigma_j \in S_l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_l))$, once removed from C_l, cannot appear in C_x for any $x > l$. In particular, consider request $\sigma_y \in \bigcup_{\sigma_j \in S_l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_l))$. Then, by definition, there exists $\sigma_j \in S_l$ such that $tok(\sigma_y) = (\sigma_j, seg(\sigma_j, \sigma_l))$. In other words, σ_y receives its token from σ_j. By inspection of the accounting code, σ_y can only be added to the set C by the request σ_j. However, by definition of S_l, request σ_j is handled before σ_l, i.e., $j < l$. Thus, once removed from C_l, σ_y cannot appear in C_x for any $x > l$. Now we can conclude that

$$C_i \bigcap \left(\bigcup_{\sigma_j \in S_l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_l)) \right) = \emptyset.$$

Thus, the only request that must be removed by σ_i is the request $tok^{-1}(\sigma_l, seg(\sigma_l, \sigma_i))$ added by σ_l. Since $|tok^{-1}(\sigma_l, seg(\sigma_l, \sigma_i))| \leq 1$,

$$\left| C_i \bigcap \left(\bigcup_{\sigma_j \in S_i^l} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right) \right| \leq 1.$$

Now consider the case where $S_i^r \neq \emptyset$. Exactly the same analysis as for the case where $S_i^l \neq \emptyset$ shows that

$$\left| C_i \bigcap \left(\bigcup_{\sigma_j \in S_i^r} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right) \right| \leq 1.$$

The lemma now follows from the fact that $S_i \subseteq S_i^l \cup S_i^r$. ∎

The next lemma states that the total number of requests currently associated with tokens that are removed by σ_i from C_i is at most 7.

Lemma 4.5.6 *Consider any execution of the* TREE *algorithm for request sequence* $\sigma_0 \ldots \sigma_{k-1}$. *Then, for all* $0 < i < k$ *such that* $\sigma_i \in C^c$,

$$\left| C_i \bigcap \left(tok(<C_i, p(\sigma_i)>) \cup tok(sp(\sigma_i)) \bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right) \right| \leq 7$$

.

Proof. By Lemma 4.5.5,

$$(4.1) \quad \left| C_i \bigcap \left(\bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right) \right| \leq 2.$$

Furthermore, the set $sp(\sigma_i)$ has cardinality at most two. Therefore,

$$(4.2) \quad |tok(sp(\sigma_i))| \leq 2.$$

Thus, we are left to consider $tok(<C_i, p(\sigma_i)>)$. Let $x = \text{LCA}(s_i, d_i)$. We split $tok(<C_i, p(\sigma_i)>)$ into two sets, C_i^r and C_i^l. The set C_i^l includes any $\sigma_j \in tok(<C_i, p(\sigma_i)>) - \{\sigma_i\}$ such that $p(\sigma_j)$ intersects the left path of σ_i, i.e., it intersects $p(s_i, x)$. Similarly, the set C_i^r includes any $\sigma_j \in tok(<C_i, p(\sigma_i)>) - \{\sigma_i\}$ such that $p(\sigma_j)$ intersects the right path of σ_i, i.e., it intersects $p(x, d_i)$. First consider the case where $C_i^l \neq \emptyset$.

Let σ_l be the request in C_i^l that, of all the requests in C_i^l, intersects $p(s_i, x)$ at the link closest to s_i. We consider two cases: $p(\sigma_l)$ includes the top left link, $\text{TL}(\sigma_i)$, of $p(\sigma_i)$ and $p(\sigma_l)$ does not include $\text{TL}(\sigma_i)$. First consider the case where $p(\sigma_l)$ includes $\text{TL}(\sigma_i)$. Since σ_l is the request that intersects $p(s_i, x)$ at the link closest to s_i and since σ_l includes $\text{TL}(\sigma_i)$, the path of any other request in C_i^l must intersect the path of σ_l. Now, since σ_l is a request accepted by the optimal off-line algorithm, the capacity constraint implies that no other request accepted by the optimal off-line algorithm intersects $p(s_i, x)$. Hence, $|C_i^l| = 1$. Now consider the case where $p(\sigma_l)$ does not include $\text{TL}(\sigma_i)$. Let $tok(\sigma_l) = (\sigma_j, z)$. Since σ_i is a candidate, σ_l can still be a candidate, and σ_j placed special roadblocks, it must be the case that $p(\sigma_j)$ intersects $p(s_i, x)$. Thus, σ_j is included in S_i. We now show that σ_l is already accounted for in Equation 4.1 by showing that $tok(\sigma_l) = (\sigma_j, seg(\sigma_j, \sigma_i))$. Since σ_i is a candidate, the links in which $p(\sigma_i)$ and $p(\sigma_j)$ intersect must be entirely within the segment $seg(\sigma_j, \sigma_i)$. Since $tok(\sigma_l) = (\sigma_j, z)$, the paths $p(\sigma_l)$ and $p(\sigma_j)$ intersect. Now, since $p(\sigma_l)$, $p(\sigma_j)$, and $p(\sigma_i)$ all intersect each other, the geometry of the tree implies that there exists a link used by all three paths. Therefore, since $p(\sigma_i)$ and $p(\sigma_j)$ intersect entirely within the segment $seg(\sigma_j, \sigma_i)$, the intersection between $p(\sigma_j)$ and $p(\sigma_l)$ must also use a link in segment $seg(\sigma_j, \sigma_i)$. Now we can conclude that $tok(\sigma_l) = (\sigma_j, seg(\sigma_j, \sigma_i))$. As a result, σ_l is already accounted for in Equation 4.1. The same argument holds for every request in C_i^l whose path does not include $\textsc{tl}(\sigma_i)$; in other words, every request in C_i^l that does not include $\textsc{tl}(\sigma_i)$ is already accounted for by Equation 4.1. Thus, we are left with requests whose paths include $\textsc{tl}(\sigma_i)$. However, the capacity constraint implies that there can only be one such request. Therefore, we conclude that

$$(4.3) \quad \left| C_i^l - \bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right| \leq 1.$$

Now consider the case where $C_i^r \neq \emptyset$. Exactly the same analysis as for the case where $C_i^l \neq \emptyset$ shows that

$$(4.4) \quad \left| C_i^r - \bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right| \leq 1.$$

Combining Equations 4.3 and 4.4 with the fact that $tok(<C_i, p(\sigma_i)>) \subseteq C_i^l \cup C_i^r \cup \{\sigma_i\}$, we get

$$(4.5) \quad \left| tok(<C_i, p(\sigma_i)>) - \bigcup_{\sigma_j \in S_i} tok^{-1}(\sigma_j, seg(\sigma_j, \sigma_i)) \right| \leq 3.$$

The lemma now follows from Equations 4.1, 4.2 and 4.5. ■

To proceed with the complexity analysis of the algorithm, we introduce some random variables. Let $ccand = |C^c|$, $ctoken = |C^t|$, $cfree = |C^f|$ and $cnfree = |C^{\neg f}|$.

Lemma 4.5.7 *Consider request sequence $\sigma_0 \ldots \sigma_{k-1}$. Then, $ccand \geq \frac{1}{7} ctoken$.*

Proof. Consider any execution of the $\textsc{tree}$ algorithm where for some σ_j, σ_i, and z, $tok(\sigma_j) = (\sigma_i, z)$. From the code in Figure 4-3 we see that σ_j is returned to C_i in the construction of

C_{i+1}. By Lemma 4.5.4, $C_k = \emptyset$. Thus, σ_j must be removed from C_x by some candidate request σ_x. (If σ_j is removed due to a special roadblock then $x = i$.) However, by Lemma 4.5.6, each candidate σ_x removes at most 7 requests with tokens from C_x. ∎

$E[P(\sigma, \text{TREE}(\sigma))]$ is the expected number of requests accepted by the TREE algorithm.

Lemma 4.5.8 *Consider request sequence $\sigma = \sigma_0 \ldots \sigma_{k-1}$. Then,*

$$E[P(\sigma, \text{TREE}(\sigma))] = \frac{1}{2} E[ccand].$$

Proof. We define the following random variables: $x_i = 1$ if σ_i is a candidate and $x_i = 0$ if σ_i is not a candidate; $y_i = 1$ if $\sigma_i \neq \perp$ and $y_i = 0$ if $\sigma_i = \perp$. Furthermore $X = \sum_{i=1}^{k-1} x_i$ and $Y = \sum_{i=1}^{k-1} y_i$. By definition, $ccand = X$ and $P(\sigma, \text{TREE}(\sigma)) = Y$. Since any candidate request is accepted with independent probability $1/2$, $Pr[y_i = 1 \mid x_i = 1] = 1/2$. Since a request can only be accepted if it is also a candidate, $Pr[y_i = 1, x_i = 1] = Pr[y_i = 1]$. This fact, combined with the fact that $Pr[y_i = 1 \mid x_i = 1] = Pr[y_i = 1, x_i = 1]/Pr[x_i = 1]$, implies that $Pr[y_i = 1] = \frac{1}{2} Pr[x_i = 1]$. Thus, $E[Y] = \frac{1}{2} E[X]$. The lemma follows from the fact that $ccand = X$ and $P(\sigma, \text{TREE}(\sigma)) = Y$. ∎

Lemma 4.5.9 *Consider request sequence $\sigma = \sigma_0 \ldots \sigma_{k-1}$. Then,*

$$E[ctoken + cfree] \geq \frac{P_o(\sigma)}{2(\log d)}.$$

Proof. Consider the event that $\sigma_j \in C^{\neg f}$. Let $\sigma_i = \mathit{first}(\sigma_j)$. With probability $1/2$, σ_i is rejected and assigns tokens. Let $\ell = \mathit{level}(\sigma_i, \sigma_j)$. With probability $1/(\log d)$, the level ℓ_i picked by σ_i for the spacing of the roadblocks is equal to ℓ. If $\ell = \ell_i$, then there exists a segment number z such that $\mathit{tok}(\sigma_j) = (\sigma_i, z)$. In other words, σ_j gets token (σ_i, z). Hence, in the event that $\sigma_j \in C^{\neg f}$, $\sigma_j \in C^t$ with probability $\frac{1}{2(\log d)}$. In other words, $Pr[\sigma_j \in C^t \mid \sigma_j \in C^{\neg f}] = \frac{1}{2(\log d)}$.

For every execution of the TREE algorithm, $C^t \subseteq C_0$, $C^f \subseteq C_0$, and $C^t \cap C^f = \emptyset$. Therefore, $E[ctoken + cfree] = \sum_{\sigma_j \in C_0} (Pr[\sigma_j \in C^t] + Pr[\sigma_j \in C^f])$. Since a request must be in $C^{\neg f}$ to receive a token, $Pr[\sigma_j \in C^t] = Pr[\sigma_j \in C^t, \sigma_j \in C^{\neg f}]$. Furthermore, $Pr[\sigma_j \in C^t, \sigma_j \in C^{\neg f}] = Pr[\sigma_j \in C^t \mid \sigma_j \in C^{\neg f}] Pr[\sigma_j \in C^{\neg f}]$. Since, $Pr[\sigma_j \in C^t \mid \sigma_j \in C^{\neg f}] = \frac{1}{2(\log d)}$, we can conclude that $E[ctoken + cfree] = \sum_{\sigma_j \in C_0} (\frac{Pr[\sigma_j \in C^{\neg f}]}{2(\log d)} + Pr[\sigma_j \in C^f])$. According to Fact 4.5.2,

$C_0 = C^f \cup C^{\neg f}$. Hence, if $Pr[\sigma_j \in C^{\neg f}] = x_j$, then $Pr[\sigma_j \in C^f] = 1 - x_j$. Now we can conclude that

$$E[\mathit{ctoken} + \mathit{cfree}] \geq \sum_{\sigma_j \in C_0} min_{x_j \in [0,1]} \{ \frac{x_j}{2(\log d)} + 1 - x_j \}.$$

Let $f(x) = \frac{x}{2(\log d)} + 1 - x$. Since $f'(x) < 0$ and $f(1) = \frac{1}{2(\log d)}$, $min_{x_j \in [0,1]} \{ \frac{x_j}{2(\log d)} + 1 - x_j \} = \frac{1}{2(\log d)}$. Using the fact that $|C_0| = P_o(\sigma)$ we now conclude that,

$$E[\mathit{ctoken} + \mathit{cfree}] \geq \sum_{\sigma_j \in C_0} min_{x_j \in [0,1]} \{ \frac{x_j}{2(\log d)} + 1 - x_j \} = \sum_{\sigma_j \in C_0} \frac{1}{2(\log d)} = \frac{P_o(\sigma)}{2(\log d)}.$$

■

Theorem 4.5.10 *The* TREE *algorithm achieves an oblivious competitive ratio of $O(\log d)$ for the admission control and routing problem on diameter d trees when all request are between leaves.*

Proof. Consider request sequence $\sigma = \sigma_0 \ldots \sigma_{k-1}$. By definition, $\mathit{ccand} \geq \mathit{cfree}$. Combining this with Lemmas 4.5.7, 4.5.8, and 4.5.9 we get:

$$\begin{aligned} E[P(\sigma, \text{TREE}(\sigma))] = \frac{1}{2} E[\mathit{ccand}] &\geq \frac{1}{2} \max \left\{ \frac{1}{7} E[\mathit{ctoken}], E[\mathit{cfree}] \right\} \\ &\geq \frac{1}{28} E[\mathit{ctoken} + \mathit{cfree}] \geq \frac{P_o(\sigma)}{56(\log d)}. \end{aligned}$$

■

4.5.4 General Case

In order to handle requests between interior nodes, we reduce the problem to the special case of requests between leaves.

Lemma 4.5.11 *The* TREE *algorithm achieves an oblivious competitive ratio of $O(\log d)$ for the admission control and routing problem on diameter d trees.*

Proof. Let T be a tree of diameter d. We define a new tree T' based on T. Initially, let $T' = T$. We modify T' as follows. Let v be a non-leaf node of T. Let $\{e_1, e_2, \ldots, e_k\}$ be the set of links adjacent to v. For every link e_j in that set, add to T' a new leaf v_j connected to v.

Repeat this process for each non-leaf node of T. This modification process does not increase the diameter of T'. Thus, the diameter of T is equal to the diameter of T'. Now consider any request sequence σ for T. We construct a new request sequence σ' as follows. The new request sequence will use T' and include only requests that go between leaf nodes. If $\sigma_0 = (T, b)$ then $\sigma'_0 = (T', b')$ where $b'(e) = 1$ for all $e \in T'$. Now consider a request $\sigma_i = (s_i, d_i)$ for $i \geq 1$. We construct σ'_i as follows. If s_i is an interior node for T, and consequently also for T', find the link e_j in T' adjacent to s_i through which the path of this request must go. Then replace s_i in that request by v_j. If d_i is an interior node for T, and consequently also for T', replace it in the same manner we replace s_i. With this construction the source and destination in σ'_i are leaf nodes.

Thus, any request sequence σ for T can be transformed into a request sequence σ' for T' such that all requests go between leaf nodes. The lemma now follows directly from Theorem 4.5.10, the fact that T and T' have the same diameter, and the fact that a path determined by TREE for request σ'_i in T' can be used by request σ_i in T. ∎

CHAPTER 5

A Practical Admission Control and Routing Algorithm

5.1 Introduction

This chapter introduces a new non-greedy admission control and routing algorithm for general topology networks. We call the algorithm EXP. The goal of our algorithm is to maximize the number of virtual circuit requests that the network accepts. The algorithm is evaluated in the simulations of Chapter 6.

Our algorithm integrates several different approaches. We use the cost-benefit framework developed as part of the admission control algorithm in [AAP93] (cf. Section 4.2). Furthermore, we extend the techniques developed in the context of reservation-based algorithms [OK85, SD94] and use these techniques to incorporate the stochastic properties of the offered traffic into the definition of the link costs used in the cost-benefit framework. It is important to note that the stochastic properties used by the algorithm do not depend on the network's traffic pattern.

5.2 Finite Durations

In the admission control and routing problems discussed so far, the virtual circuits have infinite duration. In many important applications (cf. Chapter 1), virtual circuits have finite durations. To accommodate finite duration requests, the definition of the admission control and routing

AAP$(s_i, d_i, r_i, t_i^s, t_i^f)$:
  for all $\tau, e \in E:\ c(\tau, e) = r_i(\mu^{u(\tau,e)} - 1)$;
  if there exists a path p in G from s_i to d_i such that
$$\int_{t_i^s \le \tau \le t_i^f} \sum_{e \in p} c(\tau, e) \le \varrho \text{ and } u(\tau, e) + r_i/b(e) < 1 \text{ for all } e \in p \text{ and } \tau \qquad (*)$$
  then route the requested virtual circuit on p, and set:
    for all $e \in E, t_i^s \le \tau \le t_i^f$: $u(\tau, e) = u(\tau, e) + \frac{r_i}{b(e)}$ **if** $e \in p$;
  else reject the requested virtual circuit.

Figure 5-1: The AAP admission control algorithm.

problem changes slightly. In particular, requests for the *finite duration admission control and routing problem* provide a *starting time* and an *ending time* with each request. The difference between the starting time and the ending time is the *duration* of the request. The condition that enforces the capacity constraints in Definition 2.3.1 needs to be appropriately modified to account for the finite duration of the virtual circuits.

In [AAP93], Awerbuch et al. present an algorithm for the finite duration admission control and routing problem on general topology networks. We call the algorithm AAP. (Section 4.2 presents a special case of the AAP algorithm, called IAAP.) We present the AAP algorithm since it provides the starting point for our EXP algorithm.

The i^{th} virtual circuit request to AAP is a five-tuple $(s_i, d_i, r_i, t_i^s, t_i^f)$ consisting of the source node s_i, destination node d_i, bandwidth requirement r_i, starting time t_i^s, and ending time t_i^f. For simplicity, we assume that the routing is done at exactly time t_i^s. The algorithm either accepts the request, allocating bandwidth r_i along an appropriate route, or rejects the request. The goal of the algorithm is to maximize the total number of accepted requests[1]. Let $t_i = t_i^f - t_i^s$ denote the "holding time" of the circuit. Finally, let t_{max} denote the maximum possible holding time, t_{main} denote the minimum possible holding time, r_{max} denote the maximum possible requested bandwidth, and r_{min} the minimum possible requested bandwidth.

The routing decision is based on current information about the current and future utilization of the network links. The utilization of link e at time τ as seen by the routing algorithm when

[1]It is easy to modify the algorithm to optimize a general "profit" measure, where each routed request brings a predefined profit.

routing the j^{th} circuit is defined as follows:

$$u_j(\tau, e) = \sum_{\substack{1 \le i < j \,|\, \rho_i \ne \perp, \\ \tau \in [t^s(\sigma_i), t^f(\sigma_i)), e \in \rho_i}} \frac{r_i}{b(e)}.$$

Using the utilization, the algorithm computes the *exponential cost.* The cost of link e at time τ as seen by the routing algorithm when routing the j^{th} circuit is defined by

$$c_j(\tau, e) = r_j(\mu^{u_j(\tau,e)} - 1),$$

where μ is a parameter. The cost of a path p for request σ_j is the sum of the link costs, $c_j(\tau, e)$, for all links $e \in p$ integrated over the duration of the request. (See Figure 5.2.). If there exists a path p in the network such that the cost of the path is no greater than ϱ, where ϱ is a parameter, then the request is accepted along path p. The final step of the algorithm is to update the state. The AAP algorithm is shown in Figure 5.2.

Let d be the number of links in the longest simple path in the network. The main result of [AAP93, Plo95] is that choosing $\mu = 2dr_{max}t_{max}/r_{min} + 1$ and $\varrho = dr_{max}t_{max}$ guarantees a competitive ratio of $O(\log \mu) = O(\log(dr_{max}t_{max}/r_{min}))$ when r_i is restricted such that $r_i \le \min_e\{b(e)/\log \mu\}$.

5.3 Algorithm

There are several aspects of the AAP algorithm that prevent it from being practical. First, the AAP algorithm deals only with admission control and does not address routing. Second, it requires a priori specification of the duration for each request. Third, it requires each link to maintain and distribute large amounts of state information. Finally, the AAP algorithm is optimized for the worst-case situation and does not work well in common situations. Addressing each of these issues lead us to the EXP algorithm, shown in Figure 5-2.

AAP is essentially only an admission control algorithm. The only requirement on a chosen route is that it meets the admission control requirements given in the starred line of Figure 5.2. Thus, AAP would permit choosing the longest path from among those meeting the admission control requirements. In contrast, EXP provides an explicit way to choose a route. Specifically,

EXP(s_i, d_i, r_i):
if there exists a minimum-hop path P in G from s_i to d_i s.t.

$$\sum_{e \in P} \mu^{u(e)} \leq \varrho \text{ and } u(e) + r_i/b(e) < 1 \text{ for all } e \in P$$

then route the requested virtual circuit on P, and set:
for all $e \in E$: $u(e) = u(e) + \frac{r_i}{b(e)}$ **if** $e \in P$;
else reject the requested virtual circuit.

Figure 5-2: The EXP admission control and routing algorithm.

EXP chooses the minimum-hop path that meets the admission control requirements. We make no claims about the optimality of this choice, but note the following advantages. A minimum-hop path uses the fewest physical resources. Furthermore, a minimum-hop path is determined by static rather than dynamic state information. This has advantages for distributed implementations of our algorithm (see Section 5.4). Section 6.7 provides simulation data that suggests that the use of a minimum-hop path leads to good performance over a wide range of network environments. The advantages of minimum-hop routing in the context of circuit networks are also discussed by Ahmadi et. al. [ACG91].

In AAP, the cost of a path is determined in the starred line of Figure 5.2. The cost is given by an integral over the duration of the virtual circuit. This approach has two problems: the duration of each circuit must be known in advance, and each link must maintain the ending time and bandwidth of each virtual circuit. To address these problems, we simplify the cost function. In particular, instead of using $\int_\tau r_i(\mu^{u(\tau,e)} - 1)$ we use $r_i \mu^{u(e)}$, eliminating the integration step. Eliminating the integration step can be justified in the context of competitive analysis if one makes statistical assumptions about the durations of the virtual circuits [GKPR95c]. Furthermore, for the moment, we restrict attention to the case where the bandwidth of each virtual circuit is the same (denoted by r) and the capacity of each link is the same (denoted by b). As a result, r_i becomes a constant that gets absorbed into the constant ϱ and hence not used in the description of the algorithm. (We will eventually remove some of these restrictions.)

The fact that AAP is optimized for the worst-case situation reflects itself in its poor choice for the constants ϱ and μ. To address this issue, we provide a new mechanism for choosing ϱ and μ. First we set the value of ϱ relative to μ. We observe that a path consisting of a single link provides the most efficient use of resources possible and therefore should always accept a circuit request. Since the cost of a single link path is at most μ, we set $\varrho = \mu$. This ensures that lack of capacity is the only reason that the admission control procedure does not accept a virtual circuit along a path consisting of a single link.

To define μ, we look at a specific situation and calculate the correct value of μ for that situation. Then, we argue why setting μ correctly for that specific situation will lead to good performance in general. The specific situation we consider is a network consisting of three nodes (two links) in series. Now define the *critical utilization*, u^*, to be the link utilization such that a two link path, where both links have at least utilization u^*, will reject a circuit request in favor of future single link requests. Given u^*, it is easy to calculate μ as follows. Recalling that $\varrho = \mu$, we define μ such that

$$2\mu^{u^*} \geq \mu.$$

Using an equality and solving for μ we have: $\mu = 2^{1/(1-u^*)}$.

To calculate u^*, we borrow from the stochastic analysis in [OK85, SD94]. Consider a single link that can accommodate b/r simultaneous circuits. Assume that circuit request arrivals are Poisson with rate λ and that the durations are exponentially distributed with mean 1. Assume further that there are currently j circuits using the link. Then the increase that accepting an additional circuit on the link will cause in the expected number of future virtual circuits rejected due to lack of capacity is given by:

$$\frac{B(b/r, \lambda)}{B(j, \lambda)}$$

where B is the standard b-erlang loss formula [OK85, SD94]. Now consider our two link network. For simplicity we will assume that the departures on each link are independent[2]. This assumption has become standard in the literature [Mar83, GK90]. Let λ be the Poisson arrival rate of virtual circuits requiring a single link path. Assume that the two links currently both carry j circuits. Since a two link path could potentially block two single link paths, we require the increase that accepting an additional circuit on the link will cause in the expected number of future virtual circuits rejected due to lack of capacity to be less than .5 for both links. In

[2]This is obviously an approximation since two link paths will create a dependency in the departure processes of the links.

other words, a two link path is rejected in favor of a one link path once $\frac{B(b/r,\lambda)}{B(j,\lambda)} > .5$. Notice that, when $\frac{B(b/r,\lambda)}{B(j,\lambda)} = .5$ on both links, the total increase that accepting an additional circuit on the link will cause in the expected number of future virtual circuits rejected due to lack of capacity is one. Thus, on an expected value basis, we are indifferent between the current virtual circuit request and the expected future virtual circuits that will be lost by accepting the current virtual circuit request. The utilization, $u^* = jb/r$, for which a two link path is rejected in this scenario can now be calculated if λ and b/r are given. (This above analysis is similar in spirit to the analysis in [SD94].)

The value of u^* depends on the values λ and b/r. The value for b/r is known as part of the network description. Determining the correct value for λ is more complicated. Above we define it as the arrival rate of single-link virtual circuit requests. Unfortunately, this arrival rate is highly dependent on the topology and traffic matrix of the network. Recall that one of our goals is not to require such a dependence. Consequently, we propose the following heuristic for setting λ. Discussions with engineers charged with operations for several commercial networks suggest that 2% is the highest loss rate that a network should ever produce. We use this 2% figure to calibrate λ. In particular, we assume that the arrival rate of single-link circuits to any link is never more than λ^*, where λ^* is the arrival rate needed to generate a 2% loss rate on a single link in the absence of any other traffic. We set u^* using λ^*. By using λ^*, we are essentially calibrating our algorithm for the most aggressive admission control policy that will realistically be required[3]. In Section 5.4 we discuss why this aggressive form of admission control does not compromise the performance of the algorithm in most situations. Also, the simulations in Section 6.8 explore the sensitivity of EXP to λ^*.

Finally, we remove the restriction that b/r be the same for all links. In particular, we note that the cost requirements on the path P chosen by EXP (cf. Figure 5-2) can be written as $\sum_{e \in P} \mu^{u(e)-1} \leq 1$ instead of $\sum_{e \in P} \mu^{u(e)} \leq \mu$. Now we observe that μ no longer appears outside the summation. As a consequence, we can make μ link specific. Thus, if b/r is known on a per link basis, μ can be calculated and used by the algorithm on a per link basis. Note, however, that b/r must still be known for *each* link. The simulation results in Chapter 6 suggest that

[3]This is not strictly true when there is a large number of alternate short paths for a single link path. In particular, the stochastic properties that keep λ^* significantly below the capacity bound become less important in thsi case.

removing the requirement that b/r be known for each link will be difficult.

5.4 Discussion

In defining EXP we have retained three key insights from the AAP algorithm. First, we have retained the cost-benefit framework for determining whether a circuit can be accepted on a particular path. The cost-benefit framework has the advantage that the use of a lightly loaded link does not penalize a circuit. As a comparison, consider the algorithm in [SD94], which establishes admission control criteria on a link-by-link basis. In particular, it rejects a virtual circuit request even if the admission control criteria fail on a single link of a path. Consider a two-link (non primary) path with a highly utilized link and a lightly utilized link. The algorithm in [SD94] will reject a circuit along this path if the admission control criteria are not met on the highly utilized link. However, it might not be prudent to reject the circuit in this case.

The intuition for not rejecting the circuit is that the admission control algorithm should only protect scarce resources. Since the path in this example includes only a single scarce link it should be treated similar to a single link path using a scarce link. Recall that a single link path should always be accepted since it provides the most efficient use of resources possible. Our algorithm has the correct behavior in this case. Since we use a cost function that is exponential in the utilization, the highly utilized link will essentially be the only contributor to the cost of the path.

The second insight from the AAP algorithm that we retain is the relationship between admission control and the path length. Consider a path of length L where each link along the path has the same utilization. We now ask the following question: what is the maximum utilization u for which the L-link path should satisfy the admission control criteria? To answer this question in the context of an exponential function based algorithm we solve for u in the equation $\mu = L\mu^u$ to get $u = 1 - (\log L)/(\log \mu)$. Thus, the maximum utilization for which a path satisfies the admission control criteria decreases logarithmically with the length of the path.

Finally, we retain the observation that the admission control requirements provide essentially all of the state specific feedback that is needed for routing. By restricting the set of paths on which a circuit may be routed, the admission control component of EXP makes some implicit routing decisions. Once the state dependent restrictions are made, EXP can use state independent criteria (e.g., hop count) for deciding between the paths that meet the admission control restrictions. The ability to use state independent criteria has some advantages for distributed implementations of our algorithm. In particular, a distributed EXP algorithm can try paths in order of hop count. Each time it tries a path it can send a "setup" packet along the path to see it the path meets the admission control requirements. If it does, the path is chosen. If not, the next path is tried. (In practice, only few paths need to be tried before one can reject the circuit [GKPR95a].) This approach is also used in [SD94]. We verify the sufficiency of using state independent criteria for deciding between the paths that meet the admission control restrictions with the simulations in Section 6.7.

Recall that our admission control algorithm is calibrated for very aggressive admission control since we assume that each link can reach a 2% rejection rate solely based on single link traffic. We provide an intuitive justification for this approach by considering two types of networks: one where the topology and the traffic matrix[4] are well matched and one where they are not well matched. In a network where the topology and the traffic matrix are well matched, there are direct links between source-destination pairs with large amounts of traffic. Thus, the assumption that most links service primarily single link traffic is reasonable, especially at high loads. On the other hand, this assumption does not hold when the topology and the traffic matrix are not well matched. Thus, one might expect our admission control algorithm to be too aggressive. Fortunately, this is not the case in practice. Since the network topology and the traffic matrix are not well matched, the load on the network links increases unevenly. Thus, while some links are heavily utilized, other links still have low utilization. Therefore, the primary effect of the admission control algorithm is to cause circuits to use the lightly loaded links. In other words, the primary contribution of admission control is its effect on the routing decisions. The simulations described in Section 6.6 confirm this effect.

The constant μ for our admission control currently depends on only a single parameter: b/r, the number of circuits a link can simultaneously carry. We plot μ and the *reservation level* $1-u^*$ as a function of b/r in Figures 5-3 and 5-4. The reservation level corresponds closely in

[4]The traffic matrix gives the percentage of the total network traffic that goes between each source-destination pair.

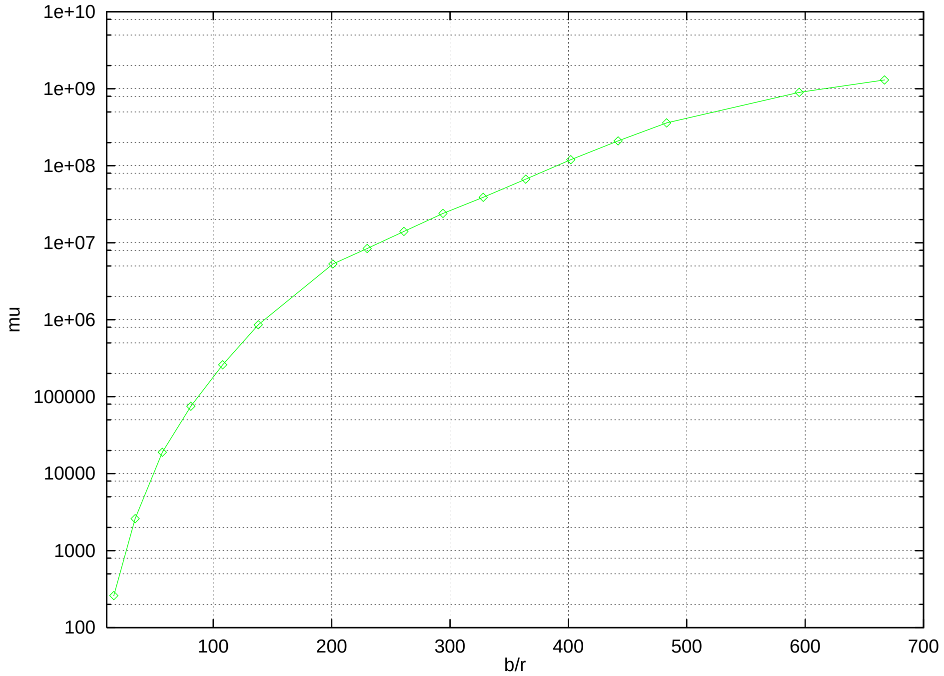

Figure 5-3: μ as a function of b/r, the number of circuits a link can simultaneously carry.

spirit to the *trunk reservation level* of the symmetric loss network literature.

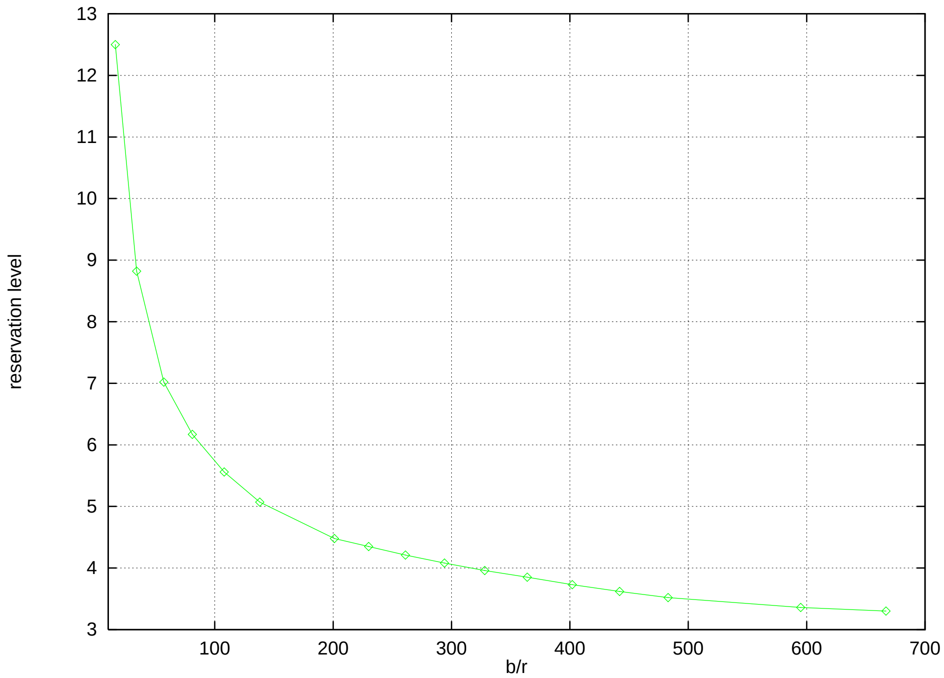

Figure 5-4: Reservation level as a function of b/r, the number of circuits a link can simultaneously carry.

CHAPTER 6

Simulation Results

6.1 Introduction

This section evaluates the performance of the EXP algorithm against a greedy admission control strategy that uses minimum-hop routing. Our simulations are based on an existing commercial topology. The simulations provide considerable insight into behavior of our algorithm.

6.2 An Existing Commercial Topology

The existing commercial network consists of 25 nodes and 61 links. The topology is pictured in Figure 6-1. The capacities of the links are all chosen to be 155 Mbps, which corresponds to SONET OC-3 service. The virtual circuits all require 1 Mbps in both directions. When we take into account the overhead from the ATM headers, each link can accommodate 140 simultaneous virtual circuits. Calculations described in the previous section imply that we should use a reservation level of 5% and $\mu = 9.4e5$ (see Figures 5-3 and 5-4). The holding times are exponentially distributed with a mean of 30 minutes. Virtual circuit requests arrive as a Poisson process. The traffic matrix corresponds to the actual current traffic on the network. We call this simulation scenario the *base case*. All simulation results have 99% confidence that they are within 5% of the sample mean.

In order to put the performance advantage of the EXP algorithm over the greedy strategy

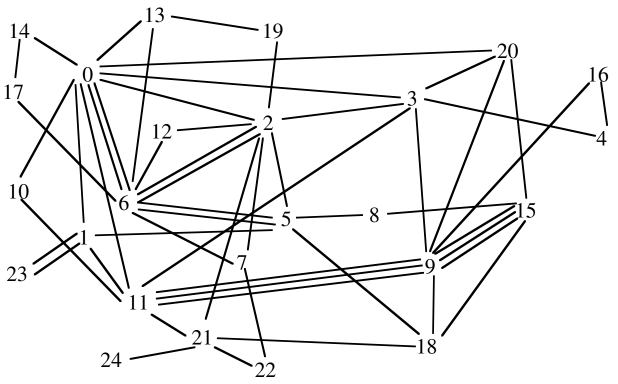

Figure 6-1: Topology of an existing commercial network.

into perspective we wish to compute the performance of the optimum off-line algorithm. Unfortunately, this computation is not tractable. Instead, we compute a lower bound on the optimum rejection rate by solving a multicommodity flow problem in which the objective function is to satisfy the maximum demand between node pairs without violating the capacity constraints, where the demand between node pairs is determined by the traffic matrix. In particular, the demand between nodes i and j is set to the average number of bits per second that are expected to be requested with i as the source and j as the destination. It is easy to see that the solution to this optimization problem is indeed a lower bound on the rejection rate. However, this lower bound may be far off from the true optimum since it does not take the stochastic properties of the circuit arrivals and departures into account. Furthermore, the multicommodity flow bound corresponds to the case where we are allowed to split a single virtual circuit over several paths.

Figure 6-2 compares the performance of the EXP algorithm with various reservation levels to the performance of a greedy minimum-hop algorithm and our lower bound on the performance of the optimum algorithm. The X-axis gives the aggregate arrival rate in virtual circuits per second and the Y-axis gives the percentage of virtual circuits that are rejected. It can be seen that the EXP algorithm has a significant performance advantage over the greedy algorithm for a wide range of arrival rates. The EXP algorithm can maintain a much higher arrival rate given a target rejection (loss) rate. For a target maximum rejection rate of 2%, the EXP algorithm with the reservation level set at 5% ($\mu = 9.4e5$) can sustain an arrival rate that is approximately 8% higher than the arrival rate that can be sustained by the greedy algorithm. Taking our bound on the optimum algorithm as 100%, EXP achieves approximately 88% throughput, while the

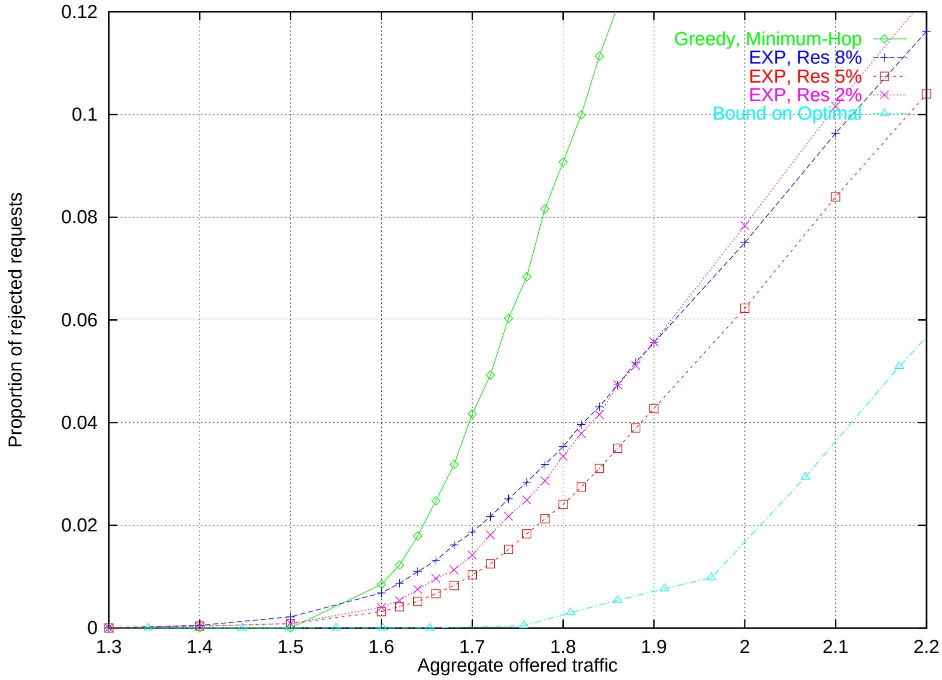

Figure 6-2: Simulation results for the commercial network.

greedy algorithm achieves only 81%. We would like to reiterate that our bound on the optimum is quite optimistic and thus we believe that EXP achieves substantially more than 88% of the real optimum throughput.

The relative performance advantage of the EXP algorithm increases with arrival rate. For example, the improvement for a target maximum rejection rate of 10% is 20%. In this case, EXP achieves approximately 88% of the bound on the optimum performance, while the greedy algorithm achieves only 73%.

Notice that the reservation level is a relatively forgiving parameter. In particular, Figure 6-2 also includes the results for reservation levels of 8% ($\mu = 5.8e3$) and 2% ($\mu = 1.1e15$). (The fact that the reservation level is a forgiving parameter was previously observed in the context of symmetric loss networks [MGH93].)

6.3 Varying Virtual Circuit Bandwidth

A key factor in determining the correct reservation level and the correct value of μ is the number of virtual circuits that can be simultaneously accommodated on a single link. Figure 5-3 shows this effect analytically. We also illustrate this effect using simulations. The simulations are the same as in the base case except for the bandwidth of the virtual circuits. The graph in Figure 6-4 shows the results for 200 Kbps circuits and the graph in Figure 6-3 shows the results for 5 Mbps circuits. The optimal reservation levels and μ values for these cases are 10%, 989 and 3.3%, $1.6e9$, respectively. In each graph, we plot results for both values of μ. The simulations confirm that the reservation level should decrease as the number of simultaneous circuits that can be accommodated increases. Furthermore, the performance advantage of the EXP algorithm over the greedy algorithm increases with the number of circuits that can be simultaneously accommodated on a single link. In particular, in the case of 5 Mbps circuits, the EXP algorithm is 2% better than the greedy minimum-hop algorithm for a target maximum rejection rate of 2%, while for 200 Kbps circuits EXP is better by 9%. At a target maximum rejection rate of 4%, the improvements are 5% and 12% for the 5 Mbps and 200 Kbps cases, respectively. The following intuition helps explain why the performance advantage of the EXP algorithm over the greedy algorithm increases with the number of circuits that can be simultaneously accommodated on a single link. When the admission control mechanism rejects a virtual circuit request along a path that has sufficient bandwidth for the request, it does so in the expectation that multiple future requests can be accepted along that path, thus increasing the total number of accepted circuits. This expectation is partially based on the statistical assumptions made by the algorithm. The law of large numbers shows that expectations arising from a random process consisting of many events are more likely to be accurate predictions. Hence the accuracy of the predictions arising from the statistical assumptions made by the algorithm increases with the number of circuits that can be simultaneously accommodated on a single link.

The dependence of the correct reservation level on the number of circuits that can be simultaneously accommodated demonstrates the importance of incorporating stochastic properties into our analysis. An analysis based entirely on competitive analysis would not be able to predict this dependence. The dependence also illustrates the importance of knowing the number of circuits that can be simultaneously accommodated for each link. (Recall the discussion in Section 5.3 where the number of circuits that can be simultaneously accommodated for each link is denoted by b/r.) Thus, the simulations suggest that it will be difficult to find a mechanism for eliminating the need to know b/r in advance for each link.

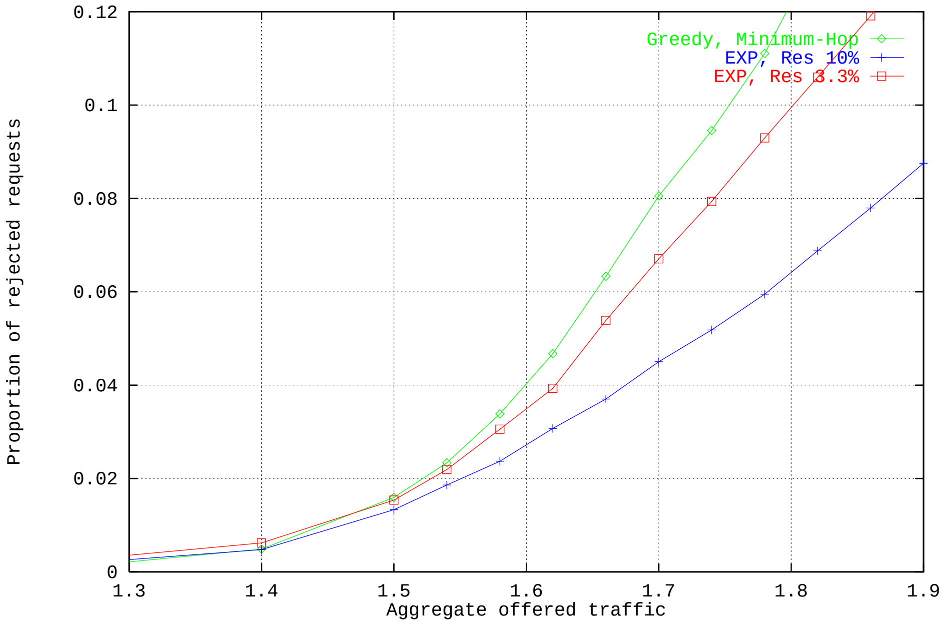

Figure 6-3: Simulations for virtual circuits with bandwidth of 5 Mbps.

6.4 Varying Duration

The simulation results shown in Figure 6-5 use a *bimodal* distribution on the durations. This distribution tests the relative performance when there is a mix of short duration and long duration circuits. The duration of each circuit comes either from an exponential distribution with mean 6 minutes or from an exponential distribution with mean 30 minutes. Circuits are split between these two mean durations to ensure that each mean duration contributes approximately half of the currently active circuits. Figure 6-5 shows that there is no observable change in the relative performance of our EXP algorithm and greedy minimum-hop algorithm.

6.5 Dynamic Traffic Patterns

This section investigates the robustness of our algorithm to environments with very dynamic traffic patterns. In particular, the simulations in Figure 6-6 randomly change the traffic matrix at time intervals of one mean circuit duration. Each change to the traffic matrix alters the traffic between any source-destination pair to a value picked uniformly at random between 0

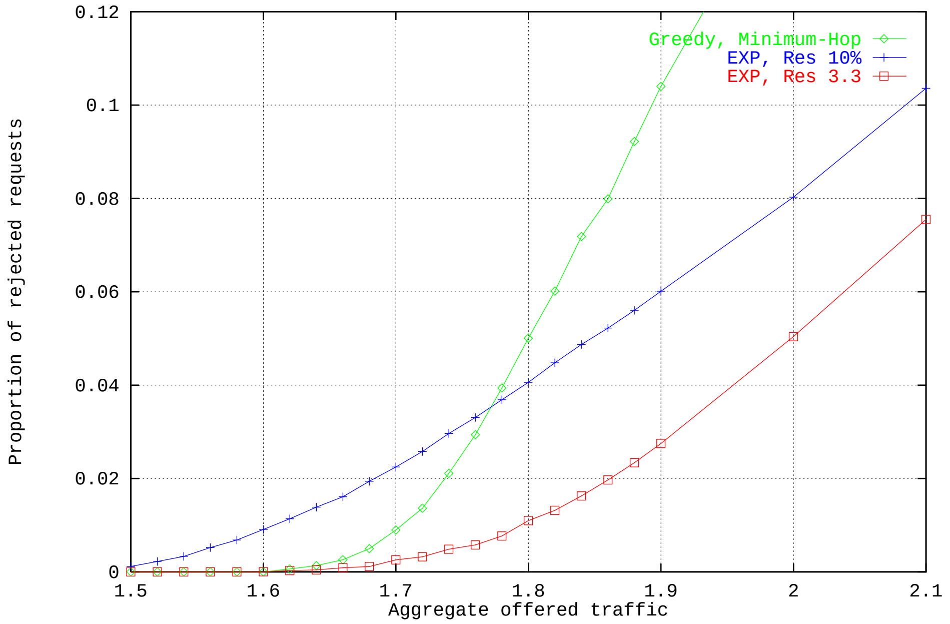

Figure 6-4: Simulations for virtual circuits with bandwidth of 200 Kbps.

and twice its value in the base simulation scenario. The results show that the EXP algorithm maintains its performance advantage over the greedy admission control strategy.

6.6 The Routing Effects of Admission Control

Even though our EXP algorithm uses a static minimum-hop criterion to decide among paths, the EXP algorithm includes an implicit state dependent routing component. The implicit state dependent routing results from the restrictions that the admission control component of the algorithm places on the set of paths from which the minimum-hop routing component of our algorithm can choose. In this section we seek to quantify the relative contributions made by the implicit state dependent routing component of the EXP algorithm and the admission control component of the EXP algorithm. Quantifying the relative contributions will also give simulation-based support to the justification in given Section 5.4 for our aggressive approach to choosing the reservation level.

To quantify the routing effect of our EXP algorithm we study the performance of a new greedy admission control algorithm that makes routing decisions that are similar to those of

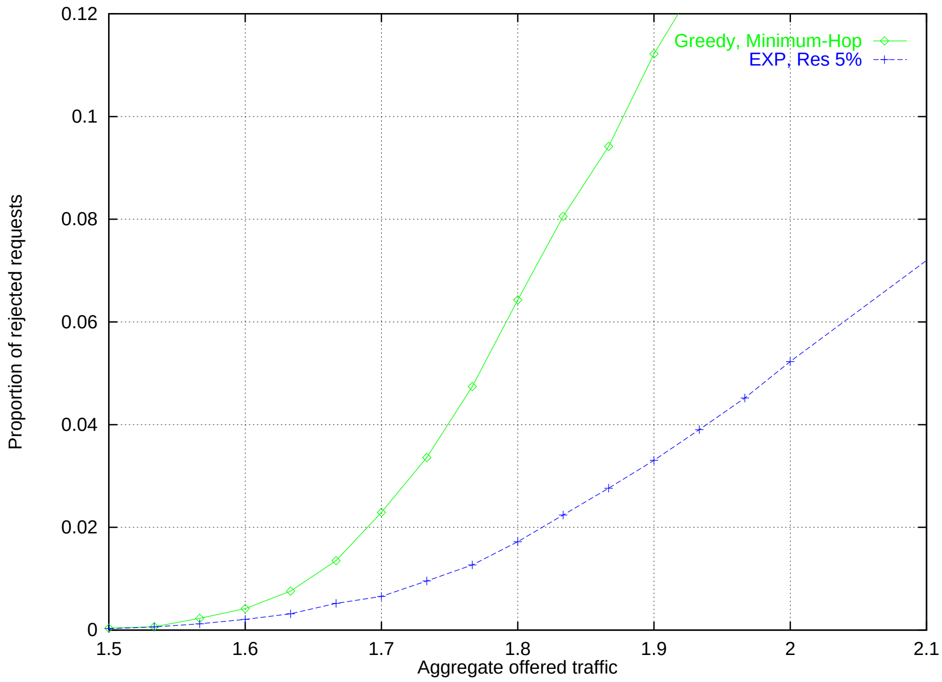

Figure 6-5: Simulations for virtual circuits with bimodal duration distribution.

the EXP algorithm. The new algorithm, called EXP-AC, is the same as the EXP algorithm except that it includes one additional step: If the EXP algorithm rejects the circuit, then the EXP-AC algorithm routes the circuit on the shortest path with respect to link costs given by $1 + \mu^{(u-1)}$, where u is the utilization of the link[1]. If, on the other hand, the EXP algorithm routes the circuit, then the EXP-AC algorithm uses the same path as the EXP algorithm.

The relative contributions made by the implicit routing and the admission control depend on the relationship between the topology and the traffic matrix. The simulations measure the percentage of the improvement over the greedy minimum-hop algorithm that is due to the implicit routing effects of the admission control (i.e., the percentage improvement achieved by the EXP-AC algorithm) as the degree to which the traffic matrix matches the topology changes.

Our simulations show that in the base case, the EXP-AC algorithm achieves 93% of the improvement that is achieved by the EXP algorithm. Thus, the implicit routing effects dominate. However, when the traffic matrix matches the topology perfectly, EXP-AC provides 0% of the

[1]Since no path meets the admission control criteria, it is not clear how to route in such a way as to capture routing effects of the admission control. The cost function seeks to find a path that comes closest to meeting the admission control requirements, without choosing a path that is too long.

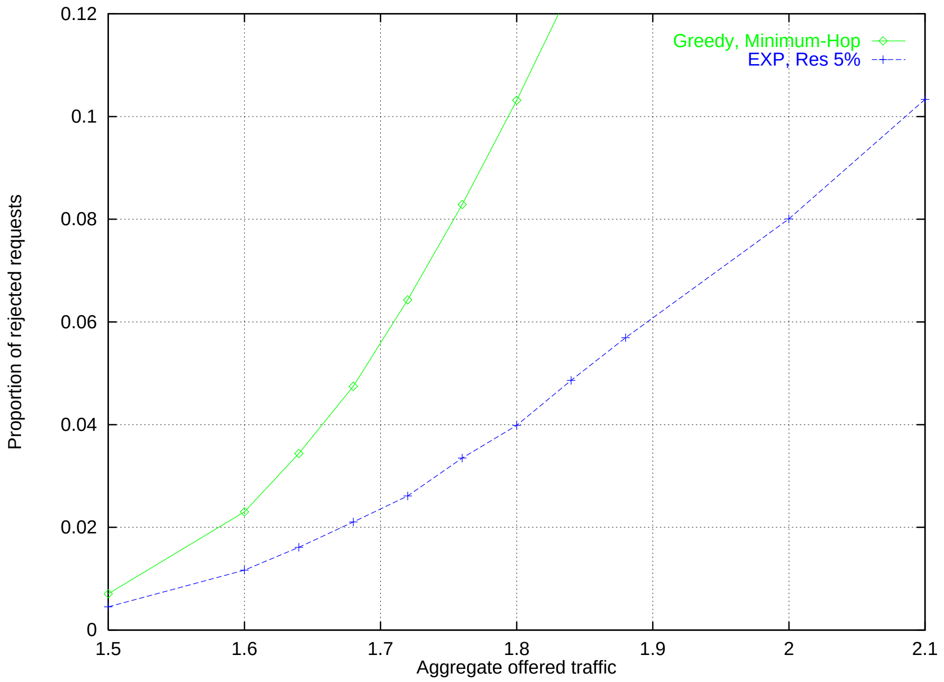

Figure 6-6: Dynamic changes to the traffic matrix.

improvement that is achieved by the EXP algorithm. Specifically, EXP-AC provides no improvement over the greedy minimum-hop algorithm while EXP provides over 10% improvement. In other words, the increase in throughput in this case is due to the admission control component of EXP. At the mid point of these extreme cases, (i.e., the traffic matrix is a 50%/50% linear combination of the base case and the traffic matrix that matches the topology perfectly) the EXP-AC algorithm achieves 46% of the improvement that is achieved by the EXP algorithm. In other words, in this case the implicit routing effects and the actual admission control contribute equally.

Now recall the discussion of Section 5.4. The following argument was used in Section 5.4 to justify the aggressive approach to choosing the reservation level: If the traffic matrix does not match the topology then the main effect of the EXP algorithm will be through its effect on routing instead of through actual admission control. If the traffic matrix closely matches the topology, then the links are utilized in a uniform manner, which immediately justifies the aggressive reservation. The simulations support this argument.

While we have only investigated the relative contribution made by the implicit routing of the admission control criteria and actual admission control in the context of the EXP algorithm, we expect to find similar results for other admission control algorithm proposed in the literature.

6.7 Cost Based Routing vs Minimum Hop Routing

In order to facilitate a distributed implementation, our algorithm attempts to minimize its use of dynamic state information, such as link utilization. In particular, the algorithm uses a static minimum-hop metric to decide among the paths that meet the admission control criteria. The obvious alternative to using the minimum-hop metric is a metric based on the link utilization. For example, one could choose the minimum cost path with respect to the exponential cost metric used for admission control. This section describes simulation results that support our claim that there are no performance penalties for using a static minimum-hop metric to decide among the paths that meet the admission control criteria. In particular, the simulations show that the inherent routing effects of the admission control provide sufficient state depended information to the routing decision.

The simulations compare the performance of the EXP algorithm to a modified algorithm that we will refer to as "EXP-MC". EXP-MC chooses the minimum cost path in the exponential cost metric used for admission control. If that path satisfies the admission control criteria, i.e., the cost of the path is sufficiently low, EXP-MC accepts the circuit. Otherwise, EXP-MC rejects the circuit. The essential difference between EXP and EXP-MC is that EXP uses a static minimum-hop metric to decide between the paths that meet the admission control requirements, while EXP-MC uses a minimum cost metric that is based on link utilizations.

A key parameter in determining the relative performance of the algorithms is the degree to which the traffic matrix and the topology match. When the traffic matrix and the topology are well matched, we would expect the EXP algorithm to outperform the EXP-MC algorithm. In this case, most virtual circuit paths should consist of one link and thus EXP-MC's greater tendency to use multi-link paths harms its performance relative to that of EXP. On the other hand, when the traffic matrix and the topology are not well matched, we would expect EXP-MC to outperform EXP. The simulations show that the performance differences between EXP and EXP-MC are not great. When the traffic matrix and the topology are perfectly matched, EXP enjoys a 2% performance advantage over EXP-MC. In other words, the arrival rate at which EXP reaches a 2% circuit rejection rate is 2% larger than the arrival rate at which EXP-MC reaches a 2% circuit rejection rate. In the base case, where the traffic matrix and the topology are not well matched, EXP-MC enjoys a 2% performance advantage over EXP.

6.8 Varying Maximum Loss Rates

Recall that the value of μ depends on the maximum loss rate. In particular, we use the maximum loss rate to set λ^*, which we use to set μ (see Chapter 5). Based on discussions with engineers charged with the operations of several commercial networks, we use a maximum loss rate of 2%. Since this 2% value is somewhat arbitrary, we need to considered the sensitivity of EXP to this value. To test the sensitivity, we consider some extreme values for the maximum loss rate. In particular, consider a low value of .1% and a high value of 4%. In the base simulation, the low value of .1% leads to a reservation level of 2.7% while the high value of 4% leads to a reservation level of 7%. Examining Figure 6-2, we note that there are only small performance differences for reservation levels determined based on maximum rejection rates that are in the interval [.1%, 4%]. Hence, the performance our EXP algorithm is not very sensitive to the choice of maximum loss rate.

Chapter 7

Competitive Multicast Admission Control and Routing Algorithms

7.1 Introduction

This chapter presents three admission control and routing algorithms for multicast communication. Requests to establish multicast communications come in two flavors called *batched* and *on-line*. For batched requests, all members join and terminate at the same time. (A teleconference call is a good example.) In contrast, for on-line requests, the members of the multicast group issue requests to join and leave the multicast group separately. (The viewers of CSPAN are an example of an on-line multicast group.)

Batched and on-line multicast groups have different service models. Batched multicast groups use *binary* admission control. With binary admission control either all of the potential multicast group members are accepted or none are accepted. We use binary admission control since batched multicast groups have coordinated arrivals, thus we expect that they will require coordinated admission control decisions. In contrast, on-line multicast groups use *non-binary* admission control. Non-binary admission control accepts some subset of the potential multicast group members. Since on-line multicast groups have independent arrivals, we expect that they will permit independent (i.e. non-binary) admission control decisions. As a consequence of using non-binary admission control, on-line multicast groups cannot, in general, rely on any particular potential member being admitted by the admission control algorithm. However, we expect that on-line multicast groups will typically have a "source" (e.g., the source of the video feed for CSPAN) that must be a member of the multicast group. Thus, our service model for on-line multicast groups includes a source, which is guaranteed to be part of the multicast group, and considers all other potential members as "destinations", whose actual membership is optional. We provide formal definitions for the admission control and routing problem for each type of multicast group in the section that presents the algorithm for that type of multicast group.

The algorithm in Section 7.3 considers batched multicast groups. In Section 7.4 we present an algorithm that can accommodate batched multicast groups as well as a restricted form of on-line multicast groups. Specifically, the on-line requests from different multicast groups may not be interleaved. In other words, all requests to join a specific multicast group occur without any intervening requests to join another multicast group. Finally, the algorithm of Section 7.5 removes the restriction that requests from different multicast groups may not be interleaved.

For each of the algorithms presented in this chapter, we make a set of simplifying assumptions. In particular, we assume that a node's membership in a multicast group is never terminated. Removal of this assumption for on-line multicast groups would be difficult without significant additional assumptions about the behavior of the potential multicast members. We discuss this issue in more detail in Chapter 8. We also assume that the bandwidth required by a multicast group is a small fraction of the link capacity for each link. This is similar to the bandwidth restriction in the AAP algorithm. (See Definition 4.2.1.) The details of this restriction differ slightly for each algorithm. Finally, we assume that the goal of the admission control and routing algorithm is to maximize the total accepted *benefit*. The benefit of a multicast group is defined as the bandwidth of the group multiplied by the maximum number of members of the group. This performance measure is formalized in Definition 7.3.2.

The algorithms described in this chapter will not perform well in practice if used in their present form. To perform well in practice, the multicast algorithms of this chapter need to be modified much in the same way that the unicast algorithm of [AAP93] is modified in Chapter 5. Section 7.6.1 discusses the types of modifications that are needed to make the multicast algorithms of this chapter perform well in practice. Section 7.6.1 also highlights the algorithmic principles demonstrated by the multicast algorithms in this chapter. Finally, Section 7.6.1

provides an informal explanation for the competitive ratios of our multicast algorithms and argues why we expect our multicast algorithms to perform much closer in practice to the optimal off-line algorithm than is suggested by the competitive ratios.

7.2 Preliminaries

7.2.1 Notation and Naming Conventions

We introduce several naming conventions for functions on graphs. Consider a graph $G = (V, E)$. A *cost function* for G is a function from E to the non-negative reals, $\Re^{\geq 0}$. We typically use cost functions to represents the cost of a link. We say that a cost function c for graph G is *polynomial* if it is polynomial in $|V|$. A *benefit function* for G is a function from V to $\Re^{\geq 0}$. We typically use benefit functions to represent the benefit associated with accepting a node into a multicast group. Based on our optimization goal (cf. Definition 7.3.1), the benefit associated with accepting a node into a multicast group will be the bandwidth requirement of that multicast group. We define a *bandwidth function* to be a function from E to $\Re^{\geq 0}$. We typically use bandwidth functions to specify the bandwidth that a multicast group adds to the bandwidth already carried by a link. Finally, a *capacity function* is a function from E to $\Re^{\geq 0}$. It represents the capacity of a link.

Consider a set of links E and any function $f : E \rightarrow \Re^{\geq 0}$. We denote $\sum_{e \in E} f(e)$ by $f(E)$. Similarly, for any set of nodes V and function $g : V \rightarrow \Re^{\geq 0}$, we denote $\sum_{v \in V} g(e)$ by $g(V)$. Finally, for graph $G = (V, E)$, let $f(G) = f(E)$ and $g(G) = g(V)$.

Consider graphs $G = (V, E)$ and $G' = (V', E')$. We say that $G \subseteq G'$ iff $V \subseteq V'$ and $E \subseteq E'$. Define $G - G'$ as follows: $(\{v \mid v \in V - V'\}, \{e \mid e \in E - E',\text{ if } e = (u, v) \text{ then } u, v \in V - V'\})$. We denote an *empty* graph $G = (\emptyset, \emptyset)$ simply by $G_\emptyset$. Consider graph $G = (V, E)$ and cost function c for G. For any two nodes $v, u \in V$ let $d(v, u)$ be the cost of the minimum cost path between v and u in G. Let $D(G)$ denote the *diameter* of G, i.e., $max_{v,u \in V}\{d(v, u)\}$. Let G_1 and G_2 be subgraphs of G. Then, the *distance* between G_1 and G_2 in G is $d(G_1, G_2) = min_{v_1 \in G_1, v_2 \in G_2}\{d(v_1, v_2)\}$.

7.2.2 Steiner Trees

The Steiner Tree problem [KMB81, RSC86, Ric92] is stated as follows.

Definition 7.2.1 (Steiner Tree) *Let $G = (V, E)$ be a graph, c a cost function for G, and $D \subseteq V$ a set of nodes. The Steiner Tree* $\textsc{st}(D)$ *for D is the minimum cost tree in G spanning the nodes in D.*

When $D = V$, the Steiner Tree is equivalent to the standard minimum cost spanning tree. In general, the problem of finding the Steiner Tree is NP-complete, however polynomial time approximate solutions exist [Wax88, NT94]. Specifically, there are algorithms (cf. [KMB81]) for which the spanning tree found by the algorithm has cost at most twice the cost of the Steiner Tree. We denote the algorithm that computes an approximate Steiner Tree by MCST. (For the purpose of this thesis it is not important which approximation algorithm is chosen.) We summarize our discussion with the following fact.

Fact 7.2.2 (ST algorithm) *The* MCST *algorithm takes as input a graph G=(V,E), a set of nodes $D \subseteq V$, and a cost function c. Its output is a tree T spanning the nodes in D such that $c(T) \leq 2\textsc{st}(D)$.*

7.2.3 Sparse Trees

Definition 7.2.3 (sparsity) *Consider a graph $G = (V, E)$, a cost function c, and a benefit function ϱ. Let $T = (V_T, E_T)$ be a subgraph of G, where $\varrho(V_T) \neq 0$. Define the* sparsity *of T to be ratio of the cost of T to the benefit of T, i.e., $c(T)/\varrho(T)$. T is said to be d-sparse if the sparsity is less than or equal to d.*

In general, we are interested in low sparsity subgraphs. Since eliminating links from a subgraph reduces its cost without changing its benefit or connectivity, all of our subgraphs will be trees.

Definition 7.2.4 (maximality) *Consider a graph $G = (V, E)$, a cost function c, and a benefit function ϱ. Let T be a subgraph of G. T is m-maximal, if for every subgraph T' of G, where $T \subseteq T'$, $c(T' - T)/\varrho(T' - T) > m$.*

Lemma 7.2.5 *For any graph G, cost function c, benefit function ϱ, and real numbers m, d such that $0 < m \leq d$, there exists a d-sparse m-maximal subgraph of G.*

Proof. We proceed by construction. (The construction takes an exponential number of steps.) Consider any subgraph T consisting of a single node. Since the sparsity of T is 0, T is d-sparse. Now consider all subgraph T' such that $T \subset T'$ and $c(T' - T)/\varrho(T' - T) \leq m$. If no such subgraph exists, T is d-sparse and m-maximal. Otherwise, pick one such subgraph T' and set $T = T'$. Since $m \leq d$, the new T is still d-sparse. Now repeat this procedure until either T is d-sparse and m-maximal or $T = G$. If $T = G$, T is also d-sparse and m-maximal. ∎

Lemma 7.2.5 proves the existence of 1-sparse 1-maximal subgraphs for any graph, cost function, and benefit function. However, we know of no polynomial time algorithm for constructing 1-sparse 1-maximal subgraphs. In fact, even verifying the 1-maximality condition seems computationally difficult. Fortunately, [AABV95] provide a polynomial time approximation algorithm. In particular, for any graph G, cost function c, and benefit function ϱ, their algorithm, which we call MAXSPARSE, can find a $O(\log^2 B)$-*sparse* 1-*maximal* subgraph where $B = \varrho(G)/min_{v \in V \text{ s.t. } \varrho(v) \neq 0}\{\varrho(v)\}$. The MAXSPARSE algorithm can guarantee the inclusion of one user specified node in the $O(\log^2 B)$-*sparse* 1-*maximal* subgraph. Furthermore, the subgraph returned by MAXSPARSE is a tree. We summarize the discussion with the following fact.

Fact 7.2.6 (MAXSPARSE algorithm) *The* MAXSPARSE *algorithm takes as input a graph G, a cost function c, a benefit function ϱ, and a node s. Let $B = \varrho(G)/min_{v \in V \text{ s.t. } \varrho(v) \neq 0}\{\varrho(v)\}$. Then the output of the* MAXSPARSE *algorithm is a $K_3 \log^2 B$*-sparse 1-maximal *tree containing s, where $K_3 \geq 1$ is a constant.*

7.2.4 Clustering

Our algorithms for on-line multicast groups require mechanisms that group nodes into connected subgraphs, called *clusters*. It is important that the size of the clusters and the overlap between the clusters is limited. Furthermore, we wish to insure that non-overlapping clusters are well separated.

Definition 7.2.7 (clustering) *Let $G = (V, E)$ be a graph, c be a polynomial cost function, and r be a real number. A* clustering *of G with respect to c and r is a set, $\mathcal{C}$, of pairs taken from the set $\{(G', v) \mid G' \subseteq G, v \in G'\}$. If $(C, v) \in \mathcal{C}$ then we call C a* cluster *and v a* center node*. Let $n = |V|$. The set $\mathcal{C}$ has the following properties:*

1. *For each* $(C, v) \in \mathcal{C}$, $r \le D(C) \le O(r \log n)$. *In other words, the diameter of any cluster is at least* r *and at most* $O(r \log n)$.

2. *For each node* $v \in V$, $1 \le |(C, u) \mid (C, u) \in \mathcal{C}, v \in C\}| \le O(\log n)$. *In other words, every node is an element of at least* 1 *and at most* $O(\log n)$ *clusters.*

3. $|\mathcal{C}| \le n$. *In other words, there are at most* n *clusters.*

4. *The clusters in* $\mathcal{C}$ *can be colored with* $O(\log n)$ *colors such that for any two clusters* C_1, C_2 *with the same color,* $d(C_1, C_2) \in \Omega(r)$. *In other words, any two clusters with the same color are not connected with any paths that have distance less than* $\Omega(r)$.

A mechanism for constructing a clustering is described in [AP90]. Specifically, [AP90] presents an algorithm, which we call CLUSTER which constructs a clustering with the properties in Definition 7.2.7.

Fact 7.2.8 (CLUSTER algorithm) *The* CLUSTER *algorithm takes as input a graph* $G = (V, E)$, *a polynomial cost function* c, *and a real number* r. *The* CLUSTER *algorithm returns a set of clusters,* $\mathcal{C}$, *with the following properties.*

1. *For each* $(C, v) \in \mathcal{C}$, $r \le D(C) \le K_1 r \log n$, *where* $K_1 \ge 1$ *is a constant.*

2. *For each node* $v \in V$, $1 \le |(C, u) \mid (C, u) \in \mathcal{C}, v \in C\}| \le K_2 \log n$, *where* $K_2 \ge 1$ *is a constant.*

3. $|\mathcal{C}| \le n$.

4. *The clusters in* $\mathcal{C}$ *can be colored with* $O(\log n)$ *colors such that for any two clusters* C_1, C_2 *with the same color,* $d(C_1, C_2) \in \Omega(r)$.

7.3 Batched Multicast Groups

This section presents an admission control and routing algorithm for batched multicast groups. We call the algorithm BMG (Batched Multicast Groups). The algorithm uses binary admission control. (A simplified version of the algorithm in Section 7.4 can be used to provide non-binary admission control for batched multicast groups.) Our algorithm has a competitive ratio of

$O(\log n)$, where n is the number of nodes in the network. Since unicast communication is a special case of multicast communication, the $\Omega(\log n)$ lower bound of [AAP93] proves our algorithm to have the optimal competitive ratio (cf. Lemma 7.3.3).

7.3.1 Problem Statement

We provide a formal definition for the admission control and routing problem for batched multicast groups. A discussion follows the definition.

Definition 7.3.1 (batched multicast admission control and routing for $\mathcal{G}$) *Let $\mathcal{G}$ be a set of graphs ranging over a node alphabet $\mathcal{V}$ If $G \in \mathcal{G}$, we describe $G = (V, E)$ by a set of nodes V and a set of undirected links E between the nodes. Furthermore, let $\mu = 2|V|(2|V|+1)$. Now define*

$$\begin{aligned}
Q_1 &= \{((V,E),b) \mid (V,E) \in \mathcal{G} \text{ and } b : E \to \Re^{\geq 0}\},\\
Q_2 &= \{(\varrho, r) \mid \varrho : \mathcal{V} \to \{0, r\}, r \in \Re^{>0}, |\{v \mid \varrho(v) \neq 0\}| \geq 2\}.
\end{aligned}$$

If $\sigma_i \in Q_2$ and $\sigma_i = (\varrho_i, r_i)$ then $\varrho(\sigma_i) = \varrho_i$ and $r(\sigma_i) = r_i$.

Let (σ, ρ) be a request sequence, response sequence pair such that $\sigma_i \in Q_2$ for all $i \in [1, |\sigma|)$. Then, for all $j \in [1, |\sigma|]$, $u_j(e) = \frac{1}{b(e)} \sum_{1 \leq i < j \,|\, \rho_i \neq \perp, e \in \rho_i} r(\sigma_i)$. Now define

$$\begin{aligned}
Q &= Q_1 \cup Q_2,\\
R &= \{T \mid T \text{ is a tree over } \mathcal{V}\} \cup \{\perp\},\\
S &= \{(\sigma, \rho) \mid\ 1.\ \sigma_0 \in Q_1, \text{ and } \rho_0 = \perp,\\
&\qquad 2.\ \sigma_i \in Q_2 \text{ for all } i \in [1, |\sigma|),\\
&\qquad 3.\ \text{if } \sigma_0 = ((V,E),b) \text{ then for all } i \in [1,|\sigma|), \{v \mid \varrho(\sigma_i)(v) \neq 0\} \subseteq V,\\
&\qquad 4.\ \text{if } \sigma_0 = ((V,E),b) \text{ then for all } i \in [1,|\sigma|), r(\sigma_i) \leq \min_{e \in E}\{\tfrac{b(e)}{\log \mu}\},\\
&\qquad 5.\ \text{if } \sigma_0 = (G,b) \text{ then for all } i \in [1,|\sigma|), \text{ if } \sigma_i \in Q_2 \text{ and } \rho_i \neq \perp \text{ then}\\
&\qquad\quad \rho_i \text{ is a tree in } G \text{ with leaves consisting of } \{v \mid \varrho(\sigma_i)(v) \neq 0\},\\
&\qquad 6.\ \text{if } \sigma_0 = ((V,E),b) \text{ then for all } e \in E, u_{|\sigma|}(e) \leq 1\}.
\end{aligned}$$

The differences between the definitions of the batched multicast problem and the (unicast) admission control and routing problem (Definition 2.3.1) result primarily from the fact that multicast requests can involve more that two nodes.

A multicast request to BMG, σ_i, is a tuple, (ϱ_i, r_i), consisting of a benefit function ϱ_i and a bandwidth requirement r_i. If node v is a *member* of the multicast group then $\varrho_i(v) = r_i$. Otherwise, $\varrho_i(v) = 0$. We require that each multicast request includes at least 2 members. The bandwidth of a multicast group is restricted to a $1/\mu$ fraction of the capacity of the lowest capacity link. (See fourth condition on S.) If a request, σ_i is accepted, the response ρ_i is a tree that spans the nodes v for which $\varrho_i(v) = r_i$. If a request, σ_i is rejected, the response $\rho_i = \perp$. With the exception of the fourth condition, the conditions on S are generalizations of the conditions in the definition of the (unicast) admission control and routing problem (cf. Definition 2.3.1) to the multicast setting. The fifth condition for S reflects the fact that we are using binary admission control. It states that the tree of an accepted multicast group must span *all* of its members.

The goal of our algorithm is to maximize the total amount of accepted benefit, where the benefit of a multicast group is defined to be its required bandwidth times the size of its membership.

Definition 7.3.2 (amount of accepted benefit) *Let $\mathcal{P} = (Q, R, S)$ be the batched multicast problem for general topology networks. Consider $(\sigma, \rho) \in \cup_i(Q^i \times R^i)$ for $i \in \mathbb{N}$. Then the* amount of accepted benefit *in ρ, $P(\sigma, \rho)$, is given by*

$$P(\sigma,\rho) = \begin{cases} \sum_{1 \le i < |\sigma| \,|\, \rho_i \ne \perp} r(\sigma_i)|\{v \mid \varrho(\sigma_i)(v) \neq 0\}| & \textit{if } (\sigma,\rho) \in S \textit{ and } |\{\rho_i \mid \rho_i \neq \perp\}| \geq 1 \\ \epsilon & \textit{otherwise} \end{cases}$$

for some fixed $\epsilon \in (0, 1)$.

We now show that lower bounds for unicast admission control and routing extend to admission control and routing for batched multicast groups. The value of μ for the batched multicast admission control and routing problem differs from the value of μ for the on-line multicast admission control and routing problem. In order to reuse the lower bounds presented below throughout this chapter, we use the most restrictive value of μ (cf. Section 7.5) in the lower bound lemmas.

Lemma 7.3.3 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the batched multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2 \log^3 |V| + 4)^{12}$. Let $\mathcal{P}'(\{\mathcal{G}\})$ be the unicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2 \log^3 |V| + 4)^{12}$.*

Let P be the performance function of Definition 7.3.2. Let P' be the performance function of Definition 2.3.7.

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P'}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P'}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Both statements also hold if P' is the the performance function in Definition 2.3.8.

Proof. Notice that the batched multicast admission control and routing problem restricted so that for each request $(\varrho, r) \in Q_2$, $|\{v \mid \varrho(v) \neq 0\}| = 2$ is the same as the unicast admission control and routing problem. Now the lemma follows from Lemma 2.4.4 and Lemma 2.5.2 and the fact that competitive ratios are invariant under constant factor changes in the performance function. ∎

7.3.2 Algorithm

Let the first request σ_0 be $((V, E), b)$. Let $n = |V|$. The admission control and routing decision is based on the current *utilization* of the network links. $u_i(e)$ denotes the utilization of link e just before the i^{th} request is handled. The utilization $u_i(e)$ is formally defined in Definition 7.3.1. Using the utilization, the algorithm computes the *exponential cost*. The cost of link e as seen by the algorithm when considering the i^{th} circuit is defined by $c_i(e) = r_i x_i(e)$, where $x_i(e) = (\mu^{u_i(e)} - 1)/n$ and μ is defined in Definition 7.3.1. Using the cost function $c_i(e)$ the algorithm now constructs an approximate Steiner tree, T_i, spanning the members of the multicast group, i.e., the nodes v for which $\varrho_i(v) = r_i$. If the cost of the approximate Steiner Tree, $c_i(T_i)$, is less than twice the benefit of the multicast group, $\varrho_i(T_i) = \sum_{v \in T_i} \varrho_i(v)$, then the multicast group is accepted. Otherwise, it is rejected. Figure 7-1 shows the code for the BMG algorithm.

7.3.3 Analysis

The analysis in this section is presented in a general form so that we can leverage some of the theorems in this section for the analysis of the algorithms in Sections 7.4 and 7.5. Our analysis consists of two parts: correctness and complexity. We consider the complexity first.

The complexity analysis uses the following simple lemma.

```
BMG(ϱ_i, r_i):
  for all e ∈ E : c(e) = r_i x(e);
  V' = {v ∈ V | ϱ_i(v) ≠ 0};
  T_i = MCST(G, V', c);
  if c(T_i) ≤ 2ϱ_i(T_i)
  then route the requested multicast group on T_i, and set:
    for all e ∈ T_i :  u(e) = u(e) + r_i/b(e);
  else reject the requested multicast group and set:
```

Figure 7-1: The BMG admission control and routing algorithm for batched multicast groups.

Lemma 7.3.4 *Let x be a real number such that $0 \leq x \leq 1$. Then $2^x - 1 \leq x$.*

Proof. Let $f(x) = 2^x - x$. We show that $f(x) \leq 1$ if $x \in [0, 1]$. The lemma follows from the facts that $f(0) = 1$, $f(1) = 1$, and $f''(x) = \frac{2^x}{\ln^2 2} > 0$ if $x \in [0, 1]$. ∎

The following lemma provides a lower bound on the total benefit accepted by the algorithm. In particular, it provides a lower bound on the benefit accepted by the algorithm in terms of the exponential cost function x and the capacity function b. To state the lemma we introduce the concept of an *acceptance sequence*.

Definition 7.3.5 (acceptance sequence) *A* acceptance sequence, *$\omega = \omega_1\omega_2\ldots$, for graph G is a sequence of triples, (T_i, R_i, ϱ_i), where T_i is a tree in G, R_i is a bandwidth function, and ϱ_i is a benefit function.*

In general, each element of an acceptance sequence will represent a multicast group request, where T_i is the subgraph along which the request allocated bandwidth, R_i gives the bandwidth the request allocated on link e, and ϱ_i is the benefit function of the request.

Lemma 7.3.6 *Consider an acceptance sequence $\omega = \omega_1 \ldots \omega_{k-1}$ for $G = (V, E)$, where $\omega_i = (T_i, R_i, \varrho_i)$.*

For each $e \in E$ and $i \in [1, k]$, let $x_i(e) = (\mu^{u_i(e)} - 1)/n$, where $u_i(e) = \frac{1}{b(e)} \sum_{1 \leq j < i | e \in T_j} R_j(e)$ and μ is a constant. Define $c_i(e) = R_i(e)x_i(e)$ to be a cost function. Assume that $R_i(e) \leq b(e)/\log\mu$ for each $i \in [0, k)$ and $e \in E$. Furthermore, assume that $max_{e \in E}\{R_i(e)\} \leq \varrho_i(T_i)$. Finally, for each $i \in [1, k)$, let $c_i(T_i)/\varrho_i(T_i) \leq S$, where $S \geq 1$. Then,

$$2S \log\mu \sum_{i=1}^{k-1} \varrho_i(T_i) \geq \sum_{e \in E} x_k(e)b(e).$$

Proof. First consider the change from $x_i(e)$ to $x_{i+1}(e)$ when $e \in T_i$.

$$\begin{aligned}
x_{i+1}(e) - x_i(e) &= \frac{1}{n}(\mu^{u_i(e)+R_i(e)/b(e)} - \mu^{u_i(e)}) \\
&= \frac{\mu^{u_i(e)}}{n}(\mu^{R_i(e)/b(e)} - 1) \\
&= \frac{\mu^{u_i(e)}}{n}(2^{\frac{R_i(e)}{b(e)}\log \mu} - 1)
\end{aligned}$$

We now make use of the fact that $2^x - 1 \leq x$ when $0 \leq x \leq 1$ (cf. Lemma 7.3.4) and the fact that the exponent of 2 in the above equality is at most 1 (since $R_i(e) \leq b(e)/\log \mu$). Therefore, when $e \in T_i$,

$$\begin{aligned}
b(e)(x_{i+1}(e) - x_i(e)) &\leq b(e)\frac{\mu^{u_i(e)}}{n}(\frac{R_i(e)}{b(e)}\log \mu) \\
&= \log \mu(c_i(e) + \frac{R_i(e)}{n}).
\end{aligned}$$

Now, summing over all links:

$$\begin{aligned}
\sum_{e \in T_i} b(e)(x_{i+1}(e) - x_i(e)) &\leq \sum_{e \in T_i}(c_i(e) + \frac{R_i(e)}{n})\log \mu \\
&\leq c_i(T_i)\log \mu + \sum_{e \in T_i}\frac{R_i(e)}{n}\log \mu
\end{aligned}$$

We now make use of the fact that $c_i(T_i)/\varrho_i(T_i) \leq S$, the fact that $max_e\{R_i(e)\} \leq \varrho_i(T_i)$, and the fact that there are at most n links in T_i.

$$\begin{aligned}
\sum_{e \in T_i} b(e)(x_{i+1}(e) - x_i(e)) &\leq \varrho_i(T_i)S\log \mu + \varrho_i(T_i)\log \mu \\
&\leq \varrho_i(T_i)2S\log \mu.
\end{aligned}$$

We complete the proof with an induction over i. Since $x_1(e) = 0$ for all $e \in E$,

$$\begin{aligned}
\sum_{e\in E} b(e)x_k(e) &= \sum_{e\in E}\sum_{i=1}^{k-1} b(e)(x_{i+1}(e) - x_i(e)) \\
&= \sum_{i=1}^{k-1}\sum_{e\in E} b(e)(x_{i+1}(e) - x_i(e)) \\
&\leq \sum_{i=1}^{k-1} \varrho_i(T_i) 2S \log \mu.
\end{aligned}$$

■

We now apply the lemma to the BMG algorithm.

Lemma 7.3.7 *Let $\sigma = \sigma_0\sigma_1 \ldots \sigma_{k-1}$ be a request sequence and let $\rho = \text{BMG}(\sigma)$ be the corresponding result sequence. For each σ_i let $T_i = \rho_i$ if $\rho_i \neq \perp$ and let $T_i = G_\emptyset$ if $\rho_i = \perp$. Then,*

$$4 \log \mu \sum_{i=1}^{k-1} \varrho_i(T_i) \geq \sum_{e\in E} x_k(e)b(e).$$

Proof. First we construct an acceptance sequence ω from request sequence σ and result sequence ρ. Consider a specific request σ_i. If $\sigma_i \neq \perp$, we define $\omega_i = (T_i, R_i, \varrho_i)$ as follows. $T_i = \rho_i$, $R_i(e) = r(\sigma_i)$ for all $e \in T_i$ and $R_i(e) = 0$ for all $e \notin T_i$, and $\varrho_i = \varrho(\sigma_i)$. If $\sigma_i = \perp$, $T_i = G_\emptyset$, $R_i(e) = 0$ for all $e \in E$, and $\varrho_i = \varrho(\sigma_i)$.

By inspection, the u_i function, x_i function, and c_i function of Lemma 7.3.6 correspond to the u_i function, x_i function, and c_i function of BMG. Furthermore, R_i is defined such that $R_i(e) \leq b(e)/\log \mu$ for all $i \in [1, k)$ and $e \in E$ and $max_{e\in E}\{R_i(e)\} \leq \varrho_i(T_i)$. Finally, BMG insures that $c_i(T_i)/\varrho_i(T_i) \leq 2$. Now we can apply Lemma 7.3.6 to conclude that

$$4 \log \mu \sum_{i=1}^{k-1} \varrho_i(T_i) \geq \sum_{e\in E} x_k(e)b(e).$$

■

Having provided a lower bound for the total benefit of the accepted multicast groups in terms of $x_k(e)b(e)$, we now provide an upper bound, in terms of $x_k(e)b(e)$, for the total benefit of the multicast groups that are accepted by the optimal off-line algorithm, but rejected by the BMG algorithm. Again, we start with a general lemma. To state this lemma we define the concept of a *rejection sequence*.

Definition 7.3.8 (rejection sequence) *A* rejection sequence, $\gamma = \gamma_1\gamma_2\ldots$, *for graph* $G = (V, E)$ *is a sequence of triples,* (M_i, r_i, ϱ_i)*, where* $M_i \subseteq V$ *is a set of nodes,* r_i *is a real number, and* ϱ_i *is a benefit function.*

In general, each element of a rejection sequence will represent a multicast group where M_i is the set of members from the multicast group that are rejected by the on-line algorithm but accepted by the off-line algorithm, r_i gives the amount of bandwidth required by the multicast group, and ϱ_i is the benefit function of the multicast group.

Lemma 7.3.9 *Consider a rejection sequence* $\gamma = \gamma_1 \ldots \gamma_{k-1}$ *for* $G = (V, E)$*, where* $\gamma_i = (M_i, r_i, \varrho_i)$*. Consider further a cost function* x_k *and a bandwidth function* $b(e)$*.*

For each element $i \in [1, k)$ *let* T_i^M *be any tree spanning the nodes in* M_i*. Assume that for all* $e \in E$*,* $\frac{1}{b(e)} \sum_{1 \le i < k | e \in T_i^M} r_i \le 1$*. Assume further that for all* $i \in [1, k)$*,* $\varrho_i(T_i^M) \le r_i x_k(T_i^M) S$*. Then,*

$$\sum_{i=1}^{k-1} \varrho_i(T_i^M) \le S \sum_{e \in E} x_k(e) b(e).$$

Proof. Since $\varrho_i(T_i^M) \le r_i x_k(T_i^M) S$ for each $i \in [1, k)$,

$$\begin{aligned}
\sum_{i=1}^{k-1} \varrho_i(T_i^M) &\le \sum_{i=1}^{k-1} r_i x_k(T_i^M) S \\
&\le \sum_{i=1}^{k-1} \sum_{e \in T_i^M} r_i x_k(e) S \\
&\le \sum_{e \in E} x_k(e) b(e) S \frac{1}{b(e)} \sum_{1 \le i < k | e \in T_i^M} r_i.
\end{aligned}$$

Since we assume that $\frac{1}{b(e)} \sum_{i \le i < k | e \in T_i^M} r_i \le 1$ for all e, we can conclude that

$$\sum_{i=1}^{k-1} \varrho(T_i^M) \le S \sum_{e \in E} x_k(e) b(e).$$

■

We now apply the lemma to the BMG algorithm.

Lemma 7.3.10 *Let* $\sigma = \sigma_0\sigma_1 \ldots \sigma_{k-1}$ *be a request sequence and let* $\rho = \mathrm{BMG}(\sigma)$ *be the corresponding result sequence.*

For each request σ_i such that $\rho_i = \perp$ but σ_i is accepted by the optimal off-line algorithm, let T_i^M be the spanning tree used by the optimal off-line algorithm. For requests σ_i such that either $\rho_i \neq \perp$ or σ_i is rejected by the optimal off-line algorithm, let $T_i^M = G_\emptyset$. Then,

$$\sum_{e \in E} x_k(e)b(e) \geq \sum_{i=1}^{k-1} \varrho_i(T_i^M).$$

Proof. First we construct a rejection sequence γ from request sequence σ and result sequence ρ. Consider a specific request σ_i. If $\rho_i = \perp$ but σ_i is accepted by the optimal off-line algorithm, define $\gamma_i = (M_i, r_i, \varrho_i)$ as follows. $M_i = \{v | \varrho(\sigma_i)(v) \neq 0\}$, $r_i = r(\sigma_i)$, and $\varrho_i = \varrho(\sigma_i)$. If $\rho_i \neq \perp$ or σ_i is rejected by the optimal off-line algorithm, $M_i = \emptyset$, $r_i = r(\sigma_i)$, and $\varrho_i = \varrho(\sigma_i)$.

Let T_i^M be the spanning tree for M_i used by the off-line algorithm. Since the trees T_i^M are accepted by the off-line algorithm, the correctness of that algorithm implies that the capacity constraints are not violated, i.e., for all $e \in E$, $\frac{1}{b(e)} \sum_{1 \leq i < k | e \in T_i^M} r_i \leq 1$.

Now consider any γ_i where $M_i \neq \emptyset$. Since $\rho = \perp$, the approximate Steiner Tree found by the MCST algorithm, T_i, has cost greater than twice its benefit. Specifically, $c_i(T_i) > 2\varrho_i(T_i) = 2\varrho_i(T_i^M)$. (See code in Figure 7-1.) The MCST algorithm guarantees that the cost of T_i is at most twice that of Steiner Tree spanning M_i. Thus, $c_i(T_i^M) \geq \frac{1}{2}c_i(T_i) > \varrho_i(T_i^M)$. Since $c_i(e) = r_i x_i(e)$ and $x_i(e)$ is increasing in i, $c_i(e) \leq r_i x_k(e)$. Therefore, the fact that $c_i(T_i^M) > \varrho_i(T_i^M)$ implies that $r_i x_k(T_i^M) > \varrho_i(T_i^M)$.

By inspection, the x_k function of Lemma 7.3.9 corresponds to the x_k function of BMG. Since $\varrho_i(T_i^M) < r_i x_k(T_i^M)$, we can apply Lemma 7.3.9 with $S = 1$ to conclude that

$$\sum_{e \in E} x_k(e)b(e) \geq \sum_{i=1}^{k-1} \varrho_i(T_i^M).$$

∎

Using Lemmas 7.3.7 and 7.3.10 we can now prove the competitive ratio for the BMG algorithm.

Theorem 7.3.11 *Let $\mathcal{P}$ be the batched multicast admission control and routing problem for general topology networks. Let P measure the amount of accepted benefit. Then, the* BMG *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(\text{BMG})$, of $O(\log n)$.*

Proof. Let $\sigma = \sigma_0\sigma_1 \ldots \sigma_{k-1}$ be a request sequence let $\rho = \text{BMG}(\sigma)$ be the corresponding result sequence. For each σ_i let $T_i = \rho_i$ if $\rho_i \neq \perp$ and let $T_i = G_\emptyset$ if $\rho_i = \perp$. Then, Lemma 7.3.7 implies that

$$4 \log \mu \sum_{i=1}^{k-1} \varrho_i(T_i) \geq \sum_{e \in E} x_k(e) b(e).$$

Thus, $1/(4 \log \mu) \sum_{e \in E} x_k(e) b(e)$ is a lower bound on $P(\sigma, \rho)$, the benefit accepted by the BMG algorithm.

Now consider the requests that are rejected by the BMG algorithm but accepted by the optimal off-line algorithm. Specifically, if $\rho_i = \perp$ but σ_i is accepted by the optimal off-line algorithm, let T_i^M be the spanning tree used by the optimal off-line algorithm. For requests σ_i such that either $\rho_i \neq \perp$ or σ_i is rejected by the optimal off-line algorithm, let $T_i^M = G_\emptyset$. Then, Lemma 7.3.10 implies that

$$\sum_{e \in E} x_k(e) b(e) \geq \sum_{i=1}^{k-1} \varrho_i(T_i^M).$$

Thus, the benefit accepted by the optimal off-line algorithm, but rejected by the BMG algorithm is bounded from above by $\sum_{e \in E} x_k(e) b(e)$.

The total benefit accepted by the optimal off-line algorithm, $P_o(\sigma)$, is thus at most $P(\sigma, \rho) + \sum_{e \in E} x_k(e) b(e)$. Therefore, the competitive ratio of the BMG algorithm is less than:

$$\begin{aligned}
\frac{P_o(\sigma)}{P(\sigma, \rho)} &\leq \frac{P(\sigma, \rho) + \sum_{e \in E} x_k(e) b(e)}{P(\sigma, \rho)} \\
&\leq \frac{P(\sigma, \rho) + 4P(\sigma, \rho) \log \mu}{P(\sigma, \rho)} = O(\log \mu) = O(\log n),
\end{aligned}$$

where the last step follows from the fact that $\mu = 2n(2n+1)$. ■

We now prove the correctness of the BMG algorithm. In other words, we show that BMG solves the batched multicast problem. We first consider the capacity constraint.

Lemma 7.3.12 *Let $\sigma = \sigma_0 \sigma_1 \ldots \sigma_{k-1}$ be a request sequence and let $\rho = \text{BMG}(\sigma)$ be the corresponding result sequence. Then, for each link $e \in E$, $u_k(e) \leq 1$.*

Proof. Recall that $\mu = 2n(2n+1)$.

We proceed by contradiction. Let i be the first index such that $u_{i+1}(e) > 1$ for some link e. From the definition of $u_{i+1}(e)$ (cf. Definition 7.3.1), $u_i(e) > 1 - r_i/b(e)$. Since we assume that

$r_i \leq b(e)/(\log \mu)$, we conclude that $u_i(e) > 1 - 1/(\log \mu)$. Thus:

$$x_i(e) = (\mu^{u_i(e)} - 1)/n > (\mu^{1-1/(\log \mu)} - 1)/n = (\mu/2 - 1)/n = 2n.$$

Therefore, $c_i = r_i x_i(e) > r_i 2n$. Since request ρ_i includes link e, this implies that $\varrho_i(T_i) > r_i n$. This is a contradiction since the maximum benefit of σ_i is $r_i n$. ■

Theorem 7.3.13 *The* BMG *algorithm solves the batched multicast admission control and routing problem for general topology networks.*

Proof. The theorem follows directly from Lemma 7.3.12, the definition of the BMG algorithm (Figure 7-1), and the definition of the batched multicast admission control and routing problem for general topology networks (Definition 7.3.1). ■

7.4 Non-Interleaved On-line Multicast Groups

This section presents an admission control and routing algorithm for batched and on-line multicast groups. We call the algorithm NOMG (Non-interleaved On-line Multicast Groups). The algorithm uses binary admission control for the batched multicast groups and non-binary admission control for the on-line multicast groups. Our service model for on-line multicast groups includes a "source", which is a member that is guaranteed to be part of the multicast group, and considers all other potential members as "destinations", whose actual membership is optional.

Since the algorithm considers on-line multicast groups, multiple requests can be associated with any given multicast group. The NOMG algorithm restricts the on-line multicast groups to be *non-interleaved.* In other words, two requests associated with the same multicast group cannot be separated by a request from a different multicast group. We formally define the non-interleaved property in Definition 7.4.1. Non-interleaved on-line multicast groups exhibit most of the complications associated with on-line multicast groups. An extension of the NOMG algorithm, presented in Section 7.5, eliminates the restriction that the on-line multicast groups must be non-interleaved.

Our algorithm achieves a competitive ratio of $O(\log^6 n)$, where n is the number of nodes in the network. Since unicast communication is a special case of multicast communication, the

$\Omega(\log n)$ lower bound of [AAP93] applies to our algorithm (cf. Lemma 7.3.3).

7.4.1 Probabilistic Assumptions

On-line multicast groups introduce two sources of complications. Complications arise from the use of non-binary admission control and from the lack of knowledge about the future multicast group membership. We discuss the complications associated with non-binary admission control first.

Consider a batched multicast group with binary admission control. The BMG algorithm of Section 7.3 accepts the multicast group if the approximate Steiner Tree spanning the potential members of the multicast group has cost less than the benefit of the multicast group. Now consider a batched multicast group with *non-binary* admission control. Informally, an admission control and routing algorithm should admit any subset of the potential members as long as an approximate Steiner Tree spanning the subset has cost less than the benefit of the subset. However, an algorithm that simply picks *any* subset with cost greater than its benefit will, in general, not have a polylogarithmic competitive ratio. Rather, an admission control and routing algorithm for non-binary admission control must pick the subset of potential members with the largest benefit from among the subsets whose cost is less that their benefit. Such a subset is said to be *maximal*. Intuitively, choosing a non-maximal subset will cause unjustifiable rejections of potential multicast members (cf. proof of Lemma 7.4.19). The MAXSPARSE algorithm (cf. Section 7.2.3) will be used to construct maximal subsets of potential multicast members.

We now turn to the issue of lack of knowledge about future membership in on-line multicast groups. Consider an on-line multicast group with three potential members, s, v_1, v_2, where s is the source and thus guaranteed membership in the multicast group. Assume that the set s, v_1, v_2 is maximal, but that no subset is maximal. Let v_1 be the first to request membership in the multicast group. Let the cost of the minimum cost path from v_1 to s be greater than the benefit of s and v_1. The algorithm now has two choices. It can either accept v_1 or reject v_1. We consider each choice separately.

- The algorithm accepts v_1.

  In this case assume that v_2 does not request membership in the multicast group. Then the multicast group consisting of s and v_1 will be spanned by a tree (path) whose cost is greater than the benefit of s and v_1. However, the use of a tree whose cost is greater than its benefit will, in general, lead to a non-polylogarithmic competitive ratio.

- The algorithm rejects v_1.

  In this case assume that v_2 does request membership in the multicast group. Independently, of whether or not the algorithm accepts v_2, we are now left with a non-maximal subset of the potential members. (Recall that the only maximal subset includes s, v_2 and v_2.) However, in general, the use of non-maximal subsets leads to a non-polylogarithmic competitive ratio.

The example suggests that an algorithm with a good (polylogarithmic) competitive ratio must know in advance whether or not v_2 is going to request membership in the multicast group when deciding whether or not to accept v_1. In other words, it must know about future requests.

To circumvent the problems associated with not knowing about future membership requests, we make some *statistical assumptions* about the requests. Namely, we make assumptions about the likelihood that a certain node will request membership in a certain multicast group. Our statistical assumptions are very general in that we allow the probability that a node requests membership in a certain multicast group to be arbitrary. The statistical assumptions about different multicast groups are revealed in an on-line fashion, at any time before the first membership request for that multicast group. In effect, we can think of the multicast group as "declaring" its statistical assumptions prior to its first use. We comment that the statistical assumptions that we make do *not* eliminate the element of uncertainty; in particular, we allow probability distributions where the probability of any specific membership request is very small. We deal with small probabilities by aggregating statistical information using the clustering techniques of [AP90].

7.4.2 Problem Statement

We provide a formal definition for the admission control and routing problem for non-interleaved on-line multicast groups. An informal discussion follows the definition. The admission control and routing problem for non-interleaved on-line multicast groups is a probabilistic problem (cf. Definition 2.1.4).

Definition 7.4.1 (non-interleaved on-line multicast for a set of graphs $\mathcal{G}$) *Let $\mathcal{I}$ be a multicast group identifier alphabet. Let $\mathcal{G}$ be a set of graphs ranging over a node alphabet $\mathcal{V}$. If $G \in \mathcal{G}$, we describe $G = (V, E)$ by a set of nodes V and a set of undirected links E between the nodes. Furthermore, let $\mu = 2|V|(2K_1K_2K_3|V|\log^3|V| + 1)$. Now define*

$$
\begin{aligned}
Q_1 &= \{((V,E),b) \mid (V,E) \in \mathcal{G} \text{ and } b : E \to \Re^{\geq 0}\},\\
Q_2 &= \{(g,\varrho,r) \mid g \in \mathcal{I}, \varrho : \mathcal{V} \to \{0,r\}, r \in \Re^{>0}, |\{v \mid \varrho(v) \neq 0\}| \geq 2\},\\
Q_3 &= \{(g,\varrho,r,s) \mid g \in \mathcal{I}, \varrho : \mathcal{V} \to [0,r], r \in \Re^{>0}, s \in \mathcal{V}\},\\
Q_4 &= \{(g,\varrho,r) \mid g \in \mathcal{I}, \varrho : \mathcal{V} \to \{0,r\}, r \in \Re^{>0}, |\{v \mid \varrho(v) \neq 0\}| = 1\}.
\end{aligned}
$$

If $\sigma_i \in Q_2 \cup Q_4$ and $\sigma_i = (g_i, \varrho_i, r_i)$ then $g(\sigma_i) = g_i$, $\varrho(\sigma_i) = \varrho_i$, and $r(\sigma_i) = r_i$. The same definitions extend to $\sigma_i \in Q_3$. Furthermore, if $\sigma_i \in Q_3$ and $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ then $s(\sigma_i) = s_i$. Finally, if $\sigma_i \in Q_4$ then $v(\sigma_i)$ is the node v such that $\varrho(\sigma_i) \neq 0$.

Let (σ, ρ) be a request sequence, response sequence pair such that $\sigma_i \in Q_2 \cup Q_3 \cup Q_4$ for all $i \in [1, |\sigma|)$. Then, for all $j \in [1, |\sigma|]$, $u_j(e) = \frac{1}{b(e)} \sum_{1 \leq i < j \,|\, \rho_i \neq \perp, e \in \rho_i} r(\sigma_i)$. Furthermore, $T_g^j = \bigcup_{1 \leq i \leq j \,|\, g(\sigma_i) \neq g, \rho_i \neq \perp} \rho_i$. Now define:

$$
\begin{aligned}
Q &= Q_1 \cup Q_2 \cup Q_3 \cup Q_4,\\
R &= \{T \mid T \text{ is a tree over } \mathcal{V}\} \cup \{\perp\},
\end{aligned}
$$

$$
\begin{aligned}
S \quad = \quad \{(\sigma,\rho) \mid \ & 1.\ \sigma_0 \in Q_1,\ \textit{and}\ \rho_0 = \bot, \\
& 2.\ \sigma_i \in Q_2 \cup Q_3 \cup Q_4\ \textit{for all}\ i \in [1,|\sigma|), \\
& 3.\ \textit{if}\ \sigma_0 = ((V,E),b)\ \textit{then for all}\ i \in [1,|\sigma|), \{v \mid \varrho(\sigma_i)(v) \neq 0\} \subseteq V, \\
& 4.\ \textit{if}\ \sigma_0 = ((V,E),b)\ \textit{then for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_3\ \textit{then}\ s(\sigma_i) \in V, \\
& 5.\ \textit{if}\ \sigma_0 = ((V,E),b)\ \textit{then}\ |\{\sigma_i \mid \sigma_i \in Q_4\}| \leq |V|, \\
& 6.\ \textit{if}\ \sigma_0 = ((V,E),b)\ \textit{then for all}\ i \in [1,|\sigma|), r(\sigma_i) \leq \min_{e\in E}\{\tfrac{b(e)}{\log \mu}\}, \\
& 7.\ \textit{if}\ \sigma_0 = (G,b)\ \textit{then for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_2\ \textit{and}\ \rho_i \neq \bot\ \textit{then}\ \rho_i \\
& \quad \textit{is a tree in}\ G\ \textit{with leaves consisting of the set}\ \{v \mid \varrho(\sigma_i)(v) \neq 0\}, \\
& 8.\ \textit{for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_3\ \textit{then}\ \rho_i = \bot\ \textit{and there exist no}\ j < i \\
& \quad \textit{such that}\ \sigma_j \in Q_3\ \textit{and}\ g(\sigma_i) = g(\sigma_j), \\
& 9.\ \textit{for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_4\ \textit{then there exist}\ \sigma_j \in Q_3\ \textit{such that} \\
& \quad j < i, g(\sigma_j) = g(\sigma_i), r(\sigma_j) = r(\sigma_i)\ \textit{and}\ g(\sigma_k) = g(\sigma_i) \forall j < k < i, \\
& 10.\ \textit{for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_3\ \textit{and there exist}\ \sigma_j \in Q_4\ \textit{such that} \\
& \quad g(\sigma_j) = g(\sigma_i)\ \textit{and}\ \rho_j \neq \bot\ \textit{then}\ s(\sigma_i) \in T^j_{g_i}, \\
& 11.\ \textit{if}\ \sigma_0 = (G,b)\ \textit{then for all}\ i \in [1,|\sigma|),\ \textit{if}\ \sigma_i \in Q_3\ \textit{and}\ \rho_i \neq \bot\ \textit{then} \\
& \quad T^i_{g_i} \textit{is a tree in}\ G\ \textit{containing}\ v(\sigma_i), \\
& 12.\ \textit{if}\ \sigma_0 = ((V,E),b)\ \textit{then for all}\ e \in E, u_{|\sigma|}(e) \leq 1\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{D} \quad = \quad \{D \mid \forall g \in \mathcal{I}, \forall v \in \mathcal{V}, Pr_D[\{\sigma \mid \exists \sigma_i \in Q_4, g(\sigma_i) = g, v(\sigma_i) = v\} \mid \\
\{\sigma \mid \exists \sigma_j \in Q_3, g(\sigma_j) = g, \varrho(\sigma_j)(v) = a\}] = a/r(\sigma_j)\}.
\end{aligned}
$$

In Definition 7.4.1 there are three types of multicast requests: BATCH requests (Q_2), INIT requests (Q_3), and JOIN requests (Q_4). A BATCH request requests the acceptance of a batched multicast group. It is the same as the multicast requests in the batched multicast problem, except that it also includes a multicast group identification number. Thus, a BATCH request is specified by a triple, (g, ϱ, r), where g is a group identification number, ϱ is a benefit function, and r is a bandwidth requirement. An INIT request initializes an on-line multicast group by specifying the source and the *expected* benefit associated with each node in the network. The expected benefit encodes the statistical information used by the algorithm. More precisely, an INIT requests consists of a four tuple (g, ϱ, r, s), where g is a group identification number, ϱ is a benefit function, r is a bandwidth requirement, and s is a source node. The benefit function ϱ is used to specify the expected benefit associate with each of the nodes. Specifically, the probability that node v will request membership in multicast group g is given by $\varrho(v)/r$. Finally, a JOIN request requests membership in an on-line multicast group for some node. A JOIN request consist of a triple (g, ϱ, r), where g is a group identification number, ϱ is a benefit function, and r is a bandwidth requirement. The benefit function returns zero for all nodes except the one requesting membership. For the node requesting membership, the benefit function returns r.

There are twelve conditions listed for S. We discuss the conditions in order. The first condition states that the first request, σ_0, consists of the network topology and the capacity information for the network. The response to the first request is $\perp$. The second condition states that each subsequent request is either a BATCH, INIT, or JOIN request. The third condition ensures that all nodes for which the benefit is non-zero are actually nodes in the network. The fourth condition ensures that the source node of an INIT request is an actual node in the network. The fifth condition ensures that there are at most $|V|$ on-line multicast groups. The sixth condition enforces constraints on the bandwidth. In particular, it ensures that the bandwidth of a multicast group is restricted to a $1/\mu$ fraction of the capacity of the lowest capacity link. The seventh condition ensures that the tree constructed for an accepted BATCH request is in the network and spans all multicast members. The requirement that the tree must span all members reflects the fact that we are using binary admission control for BATCH requests. The eighth condition states that the response to an INIT request is $\perp$ and that only one INIT request can be made for each multicast group. The ninth condition ensures that a JOIN request for group g is always proceeded by an INIT request for that group. Furthermore, each JOIN request for group g must specify the same bandwidth as the INIT request for group g. Finally, condition nine encodes the requirement that the requests be non-interleaved. In other words, for every JOIN request σ_i for group g, every request between σ_i and the INIT request for group g is also a JOIN request for group g. Consider a multicast group g. In Definition 7.4.1 we define T_g^j to be the tree constructed for group g on the basis of the requests up to and including request σ_j. The tenth condition requires that the source specified for group g is included the tree T_g^j whenever the tree includes a non-source member of the multicast group. Consider a JOIN request σ_i. Definition 7.4.1 defines $v(\sigma_i)$ to be the node that request σ_i wishes to add to the multicast group $g(\sigma_i)$. Condition eleven ensures that for every accepted JOIN request σ_i, $T^i_{g(\sigma_i)}$ is a tree that includes $v(\sigma_i)$. Finally, condition twelve enforces the capacity constraints. In particular, $u_i(e)$ represents the percent of the capacity of link e that has been used by the requests up to but not including request σ_i. We call $u_i(e)$ the *utilization* of link e just before request σ_i is handled.

Finally, we define a set $\mathcal{D}$ of probability distributions D. In particular, the set of probability distributions must ensure that the probability that node v will request membership in multicast group g is given by $\varrho(\sigma_i)(v)/r(\sigma_i)$, when σ_i is the INIT request for group g. Thus, the probability that a request sequence contains a JOIN request for group g and node v, given that it contains a INIT request for group g with benefit a is a/r where r is the bandwidth requirement of group g.

We now show that lower bounds for admission control and routing for batched multicast groups extend to admission control and routing for non-interleaved on-line multicast groups.

Lemma 7.4.2 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the non-interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V|+4)^{12}$. Let $\mathcal{P}'(\{\mathcal{G}\})$ be the batched multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V|+4)^{12}$. Let P be the performance function of Definition 7.3.2.*

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P'}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P'}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Proof. Notice that the non-interleaved on-line multicast admission control and routing problem restricted to BATCH requests is the same as the batched multicast admission control and routing problem. Now the lemma follows from Lemma 2.4.4 and Lemma 2.5.2. ∎

As an immediate consequence of Lemma 7.3.3 and Lemma 7.4.2, we conclude that lower bounds for unicast admission control and routing extend to admission control and routing for non-interleaved on-line multicast groups.

Lemma 7.4.3 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the non-interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V|+4)^{12}$. Let $\mathcal{P}'(\{\mathcal{G}\})$*

be the unicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V| + 4)^{12}$. Let P be the performance function of Definition 7.3.2. Let P' be the performance function of Definition 2.3.7.

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P'}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P'}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Both statements also hold if P' is the the performance function in Definition 2.3.8.

Proof. The lemma follows immediately from Lemma 7.3.3 and Lemma 7.4.2. ∎

7.4.3 Algorithm

Overview. In response to an INIT request $\sigma_i = (g_i, \varrho_i.r_i, s_i)$, the algorithm groups the nodes into clusters (cf. Section 7.2.4). It then reserves bandwidth r_i on a tree that spans the source, s_i, and clusters whose expected benefit is sufficiently high. A node that is an element of a cluster to which bandwidth is reserved by the INIT request will be accepted if a JOIN request arrives for that node. On the other hand, a node that is not an element of such a cluster may be rejected when a JOIN request arrives for that node.

In response to a JOIN request $\sigma_i = (g_i, \varrho_i.r_i)$, the algorithm determines the minimum cost path from the requesting node to the existing spanning tree $T^i_{g_i}$ for that multicast group. If the cost of the path is sufficiently low, the node is accepted, otherwise, it is rejected. The cost of the path is guaranteed to be sufficiently low for a node that is an element of a cluster to which bandwidth is reserved by the INIT request for that multicast group.

In response to a BATCH request, the NOMG algorithm proceeds exactly like the BMG algorithm.

Details. Let the first request σ_0 be $((V, E), b)$. Let $n = |V|$. The admission control and routing decision is based on the current utilization of the network links. Using the utilization, the algorithm computes the *exponential cost.* The cost of link e, as seen by the algorithm when considering the i^{th} multicast request, σ_i, is defined by $c_i(e) = r_i x_i(e)$, where $x_i(e) = (\mu^{u_i(e)} - 1)/n$, and μ is the constant in Definition 7.4.1.

We first consider an INIT request $\sigma_i = (g_i, \varrho_i.r_i, s_i)$. The algorithm constructs a set of clusters $\mathcal{C}_i$ using the CLUSTER algorithm. The input to the cluster algorithm is the graph G, the cost function c_i, and the parameter $r_i \log n$. The elements of the cluster set $\mathcal{C}_i$ are pairs, (C, v), consisting of a cluster, C, and a center node, v, (cf. Section 7.2.4). The resulting set of tuples will have the following properties:

1. For each $(C, v) \in \mathcal{C}$, $r_i \log n \leq D(C) \leq K_1 r_i \log^2 n$.

2. For each node $v \in V$, $1 \leq |(C, u) \mid (C, u) \in \mathcal{C}, v \in C\}| \leq K_2 \log n$.

3. $|\mathcal{C}| \leq n$.

4. The clusters in $\mathcal{C}$ can be colored with $O(\log n)$ colors such that for any two clusters C_1, C_2 with the same color, $d(C_1, C_2) \in \Omega(r_i \log n)$.

Using the results of the CLUSTER algorithm the NOMG algorithm defines a new benefit function ϱ'_i. Informally, the goal of this benefit function is to identify clusters in which the expected benefit is high enough that Chernoff bounds can be used to estimate the actual benefit with high probability. Specifically, ϱ'_i assigns a benefit of zero to all nodes except center nodes for which the expected benefit of the associated cluster is greater than $12 r_i \log n$. For these center nodes ϱ'_i assigns the expected benefit of the nodes in the associated cluster. The graph G, the cost function c_i, the new benefit function ϱ'_i, and the source s_i now form the inputs to the MAXSPARSE algorithm. The algorithm returns a 1-maximal $O(\log^2 n)$-sparse tree, $T^{i+1}_{g_i}$. Finally, the algorithm reserves bandwidth r_i on the links of the tree. Subsequent JOIN requests will attach to $T^{i+1}_{g_i}$. Figure 7-2 shows the code for an INIT request.

Recall that $T^i_{g_i}$ represent the multicast tree of group g_i just before request σ_i is handled (cf. Definition 7.4.1). Now consider a JOIN request σ_i. First the algorithm constructs the cost function c_i. Then, it identifies the node v_i which is requesting membership in the multicast group. Using the cost function c_i, it finds the shortest path p with which to attach v_i to the multicast tree $T^i_{g_i}$. If the cost of that path, $c_i(p)$, is less than $K_1 \log^2 n$ times the benefit, $\varrho_i(v_i) = r_i$, of the requesting node, then v_i is accepted. Otherwise, it is rejected. Finally, the algorithm reserves bandwidth r_i on the links of p. Figure 7-3 shows the code for a JOIN request.

Now consider a BATCH request σ_i. The algorithm proceeds exactly like the BMG algorithm, except for the fact that the ratio of $c_i(T_{g_i})$ to $\varrho_i(T_{g_i})$ can now be $2K_1 \log^2 n$ rather than 2.

NOMG-INIT$(g_i, \varrho_i, r_i, s_i)$:
 for all $e \in E$: $c(e) = r_i x(e)$; (where $x(e) = (\mu^{u(e)} - 1)/n$)
 $\mathcal{C} = \text{CLUSTER}(G, c, r_i \log n)$;
 for all $v \in V$:
  $\varrho_i'(v) = \sum_{v' \in C} \varrho_i(v')$ **if** $(C, v) \in \mathcal{C}$ and $\sum_{v' \in C} \varrho_i(v') \geq 12 r_i \log n$;
  $\varrho_i'(v) = 0$ otherwise;
 $T_{g_i} = \text{MAXSPARSE}(G, c, \varrho_i', s_i)$;
 for all $e \in T_{g_i}$: $u(e) = u(e) + \frac{r_i}{b(e)}$;

Figure 7-2: The NOMG admission control and routing algorithm an INIT request.

NOMG-JOIN(g_i, ϱ_i, r_i):
 for all $e \in E$: $c(e) = r_i x(e)$; (where $x(e) = (\mu^{u(e)} - 1)/n$)
 $v_i = v \in G$ s.t. $\varrho_i(v) \neq 0$;
 if there exists a path p in $G = (V, E)$ from v_i to T_{g_i} s.t. $\sum_{e \in p} c(e) \leq K_1 r_i \log^2 n$;
 then <u>route</u> the requesting member on p, and set:
  for all $e \in p$: $u(e) = u(e) + \frac{r_i}{b(e)}$;
  $T_{g_i} = T_{g_i} \cup p$;
 else <u>reject</u> the requesting member.

Figure 7-3: The NOMG admission control and routing algorithm for a JOIN request.

NOMG-BATCH(g_i, ϱ_i, r_i):
 for all $e \in E$: $c(e) = r_i x(e)$; (where $x(e) = (\mu^{u(e)} - 1)/n$)
 $V' = \{v \in V \mid \varrho_i(v) \neq 0\}$;
 $T_{g_i} = \text{MCST}(G, V', c)$;
 if $c(T_{g_i}) \leq 2K_1 \log^2 n \varrho_i(T_{g_i})$
 then <u>route</u> the requested multicast group on T_{g_i}, and set:
  for all $e \in T_{g_i}$: $u(e) = u(e) + \frac{r_i}{b(e)}$;
 else <u>reject</u> the requested multicast group.

Figure 7-4: The NOMG admission control and routing algorithm for a BATCH request.

7.4.4 Analysis

The analysis consists of two parts: correctness and complexity. We consider complexity first. The complexity analysis will prove that the NOMG algorithm achieves its competitive ratio with high probability (cf. Definition 2.4.3). Thus, we limit the number of multicast groups to n (cf. Section 2.4).

We introduce some notation for the analysis. Consider request sequence $\sigma = \sigma_0 \ldots \sigma_{k-1}$ and multicast group g_j. In Definition 7.4.1 we denote the tree spanning the accepted members of group g_j after request σ_{k-1} by $T^k_{g_j}$. Let G' be a subgraph of G. Then, $T^k_{g_j}|G'$ represents the nodes and links of $T^k_{g_j}$ that are also an element of G'. We define two more quantities.

Definition 7.4.4 ($\sigma(v, g_j)$) *For each node v and group g_j, let $\sigma(v, g_j)$ be the index of the highest index* JOIN *or* BATCH *request in σ that requests membership for node v in g_j. (Note, $\sigma(v, g_j)$ is undefined when no such* JOIN *or* BATCH *request exists.)*

Definition 7.4.5 (ϱ_{g_j}) *For each group g_j, let ϱ_{g_j} be the benefit function for group g_j. Specifically, $\varrho_{g_j}(v) = \varrho_{\sigma(v,g)}(v)$ when $\sigma(v, g_j)$ is defined, otherwise $\varrho_{g_j}(v) = 0$.*

The following lemma shows that certain JOIN requests are guaranteed to be successful. This lemma is based on the fact that the requests from different multicast groups are not interleaved.

Lemma 7.4.6 *Let $\sigma = \sigma_0 \ldots \sigma_{k-1}$ be a request sequence and $\rho =$ NOMG(σ) be the corresponding result sequence. Let $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ be an* INIT *request. Consider a cluster that either contains the source or whose center node is included in $T^{i+1}_{g_i}$. In other words, consider C such that $(C, v) \in \mathcal{C}_i$ and either $s_i \in C$ or $v \in T^{i+1}_{g_i}$.*

Now consider a JOIN *request for group g_i from a node in C. In other words, consider* JOIN *request $\sigma_j = (g_j, \varrho_j, r_j)$ such that $g_j = g_i$ and $v_j = v(\sigma_j) \in C$.*

Then, the NOMG *algorithm will accept v_j, i.e., $v_j \in T^k_{g_i}$ and $\rho_j \neq \perp$.*

Proof. The proof takes advantage of the fact that the request sequence σ is non-interleaved. Consider the state just before request σ_i is handled. Recall that the diameter of each cluster in $\mathcal{C}_i$ is less than $K_1 r_i \log^2 n$. Since $v_j \in C$, either there exists a path p from v_j to s_i such that $c_i(p) \leq K_1 r_i \log^2 n$ or there exists a path p from v_j to the cluster center v such that $c_i(p) \leq K_1 r_i \log^2 n$. Now compare the cost for path p seen by the JOIN request, $c_j(p)$, to the cost seen by the INIT request, $c_i(p)$. Since the request sequence is not interleaved, any increase in $c_j(p)$ over $c_i(p)$ is due to the addition of a link in p to the multicast tree for group g_i. Thus, there exists some prefix p' of p such that p' connects v_j to the multicast tree of group g_i and $c_j(p') \leq K_1 r_i \log^2 n$. The lemma now follows from the definition of the NOMG algorithm (cf. Figure 7-3) and the fact that $\varrho_j(v_j) = r_j = r_i$. ∎

Consider the INIT request $(g_i, \varrho_i, r_i, s_i)$. Consider an element $(C, v) \in \mathcal{C}_i$ of the cluster set found by the NOMG algorithm. The benefit function $\varrho_i(C)$ provides the expected benefit from the group g_i JOIN requests for all of the nodes in cluster C. Using Chernoff bounds, the following lemmas show that, with high probability, the actual benefit from the group g_i JOIN requests for all of the nodes in cluster C is close to the expected benefit.

Lemma 7.4.7 *Let A be a sum of indicator variables. Furthermore, let E be the expectation of A. If $E \geq 12 \log n$, then $A = \Theta(E)$ probability at least $1 - O(1/n^3)$.*

Proof. We proceed by providing an upper bound and a lower bound on A that holds with probability at least $1 - 1/n^3$. Based on the Chernoff bounds in [Rag86], we can conclude that

$$Pr\{A > eE\} \leq e^{-E} < 1/n^3.$$

It follows that $A = O(E)$ with probability at least $1 - 1/n^3$. The Chernoff bounds in [Rag86] also lead to the conclusion that

$$Pr\{A - E < -\gamma E\} \leq e^{\frac{-\gamma^2 E}{2}}.$$

By choosing $\gamma = .8$, we show that $Pr\{A < .2E\} < 1/n^3$. Therefore, it follows that $A = \Omega(E)$ with probability at least $1 - 1/n^3$. ∎

Lemma 7.4.8 *Let A be a sum of indicator variables. Furthermore, let E be the expectation of A. If $E \leq 12 \log n$, then $A = O(\log n)$ with probability at least $1 - O(1/n^3)$.*

Proof. Based on the Chernoff bounds in [Rag86], we can conclude that

$$Pr\{A > (1 + \delta)E\} \leq \left(\frac{e^{\delta}}{(1 + \delta)^{(1+\delta)}} \right)^E.$$

Now choose $\delta = (24e \log n)/E$ and substitute $12 \log n$ for E. Then, $Pr\{A > 24e \log n\} \leq (1/2)^{24e \log n}$. Since $(1/2)^{24e \log n} < 1/n^3$, it follows that $A = O(\log n)$ with probability at least $1 - 1/n^3$. ∎

Lemma 7.4.9 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$ (cf. Definition 7.4.1). Let $(C, v) \in \mathcal{C}_i$, $\varrho_i(C) \geq 12 r_i \log n$, and $v \in T_{g_i}^{i+1}$. Define σ_{k-1} to be the final request of any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Then,*

$$Pr_D \left[O(1) \varrho_{g_i}(T_{g_i}^k | C) \geq \varrho_i(C) \right] \geq 1 - O(1/n^3).$$

Proof. Consider any join request $\sigma_j = (g_j, \varrho_j, r_j)$ such that $i < j < k$, $g_j = g_i$, and $v_j \in C$. Then, by Lemma 7.4.6, $v_j \in T_{g_i}^k | C$. Thus, for any extension $\sigma_{i+1} \ldots \sigma_{k-1}$ of the prefix $\sigma_0 \ldots \sigma_i$, $\overline{\varrho}_{g_i}(T_{g_i}^k | C) = \varrho_{g_i}(T_{g_i}^k | C)/r_i$ is just equal to the number of nodes in C that issue a JOIN request. Furthermore, $\overline{\varrho}_i(C) = \varrho_i(C)/r_i$ is the expected number of nodes in C that issue a JOIN request. In other words, $\overline{\varrho}_{g_i}(T_{g_i}^k | C)$ is just a sum of indicator variables and $\overline{\varrho}_i(C)$ is the expectation of that sum. Furthermore, $\overline{\varrho}_i(C) \geq 12 \log n$. The lemma now follows from Lemma 7.4.7. ■

Lemma 7.4.10 *Let $D \in \mathcal{D}$ be a probability distribution over request sequences. Define σ_{k-1} to be the final request of any request sequence from distribution D. Then,*

$$Pr_D \left[O(\log^4 \mu) \sum_j \varrho_{g_j}(T_{g_j}^k) \geq \sum_{e \in E} x_k(e) b(e). \right] \geq 1 - O(1/n^3),$$

where j ranges over the indices of all BATCH *and* JOIN *requests.*

Proof. Consider a specific request sequence σ and the corresponding result sequence $\rho =$ NOMG(σ). First we construct an acceptance sequence ω from σ and ρ. Consider a specific request σ_i. We define $\omega_i = (T_i, R_i, \varrho_i)$ as follows. T_i is the tree along which the NOMG algorithm reserves bandwidth in response to request σ_i. In other words, T_i includes the links e such that $u_i(e) \neq u_{i+1}(e)$. $R_i(e) = r_i$ for all $e \in T_i$ and $R_i(e) = 0$ for all $e \notin T_i$. Finally, if σ_i is JOIN and BATCH request, then $\varrho_i = \varrho(\sigma_i)$, and if σ_i is INIT request, then ϱ_i is the constructed benefit function ϱ'_i (cf. Figure 7-2). If the NOMG algorithm did not reserve any bandwidth in response to the request then $T_i = G_\emptyset$.

By inspection, the u_i function, x_i function, and c_i function of Lemma 7.3.6 correspond to the u_i function, x_i function, and c_i function of NOMG. Furthermore, R_i is defined such that $R_i(e) \leq b(e)/\log \mu$ for all i and e and $max_e\{R_i(e)\} \leq \varrho_i(T_i)$.

For JOIN and BATCH requests NOMG insures that $c_i(T_i)/\varrho_i(T_i) \leq O(\log^2 n)$. Consider an INIT request next. Since the tree generated by the MAXSPARSE algorithm is $O(\log^2 n)$-sparse, the NOMG algorithm also guarantees that $c_i(T_i)/\varrho_i(T_i) \leq O(\log^2 n)$ for INIT requests. Now we can apply Lemma 7.3.6 and the fact that $\log n \leq \log \mu$ to conclude that

$$O(\log^3 \mu) \sum_{i=0}^{k-1} \varrho_i(T_i) \geq \sum_{e \in E} x_k(e) b(e). \tag{7.1}$$

Now we need to relate $\sum_{i=0}^{k-1} \varrho_i(T_i)$ to $\sum_j \varrho_{g_j}(T^k_{g_j})$. By definition, the tree $T^k_{g_j}$ is equal to the tree created by the union of the T_i in the subsequence of ω_i consisting of the triples created from requests for group g_j. However, the function $\varrho_{g_j}(T^k_{g_j})$ only counts the benefit from the JOIN and BATCH requests, not the INIT requests.

Consider any prefix $\sigma_0 \ldots \sigma_i$ of a request sequence from distribution D such that $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an INIT request. We need to account for the benefit of the tree generated by the NOMG algorithm, $\varrho'_i(T^{i+1}_{g_i})$, which is counted in $\sum_{i=0}^{k-1} \varrho_i(T_i)$ but not in $\sum_j \varrho_{g_j}(T^k_{g_j})$. By definition of $T^{i+1}_{g_i}$,

$$\varrho'_i(T^{i+1}_{g_i}) = \sum_{\substack{(C,v) \in \mathcal{C}_i \\ v \in T^i_{g_i}}} \varrho'_i(v) = \sum_{\substack{(C,v) \in \mathcal{C}_i \\ v \in T^i_{g_i}}} \varrho_i(C).$$

By Lemma 7.4.9, for each cluster C such that its cluster center is in $T^{i+1}_{g_i}$, it is the case that $Pr_D\left[O(1)\varrho_{g_i}(T^k_{g_i}|C) \geq \varrho_i(C)\right] \geq 1 - O(1/n^3)$. Since there are at most n clusters, $O(1)\varrho_{g_i}(T^k_{g_i}|C) \geq \varrho_i(C)$ for all clusters $(C,v) \in \mathcal{C}_i$ with probability at least $1 - O(1/n^2)$.

Since every node is an element of at most $O(\log n)$ clusters, we can conclude that

$$Pr_D\left[\sum_{\substack{(C,v) \in \mathcal{C}_i \\ v \in T^i_{g_i}}} \varrho_i(C) \leq O(\log n)\varrho_{g_i}(T^k_{g_i})\right] \geq 1 - O(1/n^2).$$

Since there are at most n multicast groups, $Pr_D\left[O(\log n)\sum_j \varrho_{g_j}(T^k_{g_j}) \geq \sum_{i=0}^{k-1} \varrho_i(T_i)\right] \geq 1 - O(1/n)$. This fact, combined with equation 7.1 allows us to conclude that

$$Pr_D\left[O(\log^4 \mu)\sum_j \varrho_{g_j}(T^k_{g_j}) \geq \sum_{e \in E} x_k(e)b(e)\right] \geq 1 - O(1/n).$$

■

Having shown a lower bound for the total benefit of the accepted multicast groups in terms of $x_k(e)b(e)$, we now provide an upper bound, in terms of $x_k(e)b(e)$, for the total benefit of the multicast groups that are accepted by the optimal off-line algorithm, but rejected by the NOMG algorithm.

Consider request sequence $\sigma = \sigma_0 \ldots \sigma_{k-1}$ and the corresponding result sequence $\rho =$ NOMG(σ). Let $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ be an INIT request. We now define several quantities.

Definition 7.4.11 (M_{g_i}) *M_{g_i} is a set nodes consisting of s_i and the set of nodes v such that there exists a* JOIN *request σ_j for which $v = v(\sigma_j)$, $g_i = g(\sigma_j)$, $\rho_j = \perp$ and σ_j is accepted by the optimal off-line algorithm.*

In other words, M_{g_i} consist of the source for group g_i, and the set of nodes which issue JOIN requests for group g_i but are rejected by the NOMG algorithm and accepted by the optimal off-line algorithm. We divide the set M_{g_i} into two sets: $M^{>}_{g_i}$ and $M^{<}_{g_i}$.

Definition 7.4.12 ($M^{>}_{g_i}$) *The set $M^{>}_{g_i}$ is a subset of M_{g_i}:*

$$M^{>}_{g_i} = \{v | v = s_i \text{ or } (v \in M_{g_i} \text{ and } v \in C \text{ where}$$
$$((C, v') \in \mathcal{C} \text{ and } \varrho_i(C) \geq 12 r_i \log n \text{ and } s_i \notin C))\}.$$

In other words, the set $M^{>}_{g_i}$ includes the source and the nodes in M_{g_i} that are elements of clusters in $\mathcal{C}_i$ that have benefit greater than $12 r_i \log n$ and do not include the source node.

Definition 7.4.13 ($M^{<}_{g_i}$) *The set $M^{<}_{g_i}$ is a subset of M_{g_i}:*

$$M^{<}_{g_i} = \{v | v = s_i \text{ or } (v \in M_{g_i} \text{ and } v \in C \text{ where}$$
$$((C, v') \in \mathcal{C} \text{ and } \varrho_i(C) < 12 r_i \log n \text{ and } s_i \notin C))\}.$$

In other words, the set $M^{<}_{g_i}$ includes the source and the nodes in M_{g_i} that are elements of clusters in $\mathcal{C}_i$ that have benefit less than $12 r_i \log n$ and do not include the source node. Note that Lemma 7.4.6 implies that $M_{g_i} = M^{<}_{g_i} \cup M^{>}_{g_i}$.

Consider the following Chernoff bound lemma.

Lemma 7.4.14 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$. Furthermore, let $(C, v) \in \mathcal{C}_i$, $\varrho_i(C) \geq 12 r_i \log n$ and $s_i \notin C$.*

Let $M^{>}_{g_i}$ be defined as in Definition 7.4.12 for any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Define $M^{>}_{g_i}|C$ be the nodes of $M^{>}_{g_i}$ that are also in cluster C. Then,

$$Pr_D\left[\varrho_{g_i}(T^{M>}_{g_i}|C) \leq O(1)\varrho'_i(v)\right] \geq 1 - O(1/n^3).$$

Proof. For any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$, define A to be the number of nodes in cluster C that issue a join request. Thus, A is a random variable based on the distribution D over request sequences. Let E be the expected number of nodes in cluster C that issue a JOIN request. $E = \varrho'_i(v)/r_i \geq 12\log n$. Hence, by Lemma 7.4.7, $Pr_D[A = \Theta(E)] \geq 1 - O(1/n^3)$. The lemma now follows from the observation that $\varrho_{g_i}(T^{M>}_{g_i}|C)/r_i \leq A$ and that $E = \varrho'_i(v)/r_i$. ■

Lemma 7.4.15 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$.*

Let $M^{>}_{g_i}$ be defined as in Definition 7.4.12 for any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Let $T^{M>}_{g_i}$ be any tree spanning $M^{>}_{g_i}$. Then,

$$Pr_D\left[\varrho_{g_i}(T^{M>}_{g_i}) \leq O(\log^2 n) r_i x_k(T^{M>}_{g_i})\right] \geq 1 - O(1/n^2).$$

Proof. Recall that the clusters in $\mathcal{C}_i$ can be colored with $O(\log n)$ colors so that the minimum cost path between any two clusters of the same color is $\Omega(r_i \log n)$, where the cost of a link is given by c_i. Consider any particular extension of request sequence prefix $\sigma_0 \ldots \sigma_i$. Consider $M^{>}_{g_i}$ for that extension. Clearly, there exists some color, say red, such that a $\Omega(\log n)$ fraction of the nodes in $M^{>}_{g_i}$ are elements of clusters with the color red. Denote by $R^{>}_{g_i}$ the nodes in $M^{>}_{g_i}$ that are elements of red clusters. Furthermore, define $R^{>c}_{g_i}$ to be the center nodes of the red clusters which have elements in $R^{>}_{g_i}$. Let $T^{R>}_{g_i}$ be the subtree of $T^{M>}_{g_i}$ spanning the nodes in $R^{>}_{g_i}$.

Let $T^{R>c}_{g_i}$ be the minimum cost (using cost function c_i) tree spanning s_i and the nodes in $R^{>c}_{g_i}$. By Lemma 7.4.6 and the definition of $R^{>c}_{g_i}$, the tree $T^{i+1}_{g_i}$ constructed by the NOMG algorithm does not include the center nodes in $R^{>c}_{g_i}$. Since $T^{i+1}_{g_i}$ is 1-maximal and does not include the center nodes in $R^{>c}_{g_i}$, it must be the case that

$$c_i(T^{R>c}_{g_i}) \geq \varrho'_i(T^{R>c}_{g_i}). \tag{7.2}$$

To complete the proof, we relate the terms in this inequality back to $T_{g_i}^{M>}$. First consider the cost. Since the minimum cost path between any two clusters of the same color is $\Omega(r_i \log n)$ and the diameter of each cluster is $O(r_i \log^2 n)$, $c_i(T_{g_i}^{R>^c}) \leq O(\log n) c_i(T_{g_i}^{R>})$. Furthermore, since $T_{g_i}^{R>}$ is a subtree of $T_{g_i}^{M>}$, $c_i(T_{g_i}^{R>}) \leq c_i(T_{g_i}^{M>})$. Thus,

$$O(\log n) c_i(T_{g_i}^{M>}) \geq c_i(T_{g_i}^{R>^c}). \tag{7.3}$$

Now consider the benefit. The argument about the benefit is probabilistic. Thus, we no longer consider a specific request sequence. Rather, we consider random variables determined by the distribution D over request sequences. By definition of ϱ'_i, Lemma 7.4.14, and the fact that there are at most n clusters, $Pr_D\left[O(1)\varrho'_i(T_{g_i}^{R>^c}) \geq \varrho_{g_i}(T_{g_i}^{R>})\right] \geq 1 - O(1/n^2)$. Furthermore, by construction, for any extension of request sequence prefix $\sigma_0 \ldots \sigma_i$, $\varrho_{g_i}(T_{g_i}^{R>}) O(\log n) \geq \varrho_{g_i}(T_{g_i}^{M>})$. Thus,

$$Pr_D\left[O(\log n)\varrho'_i(T_{g_i}^{R>^c}) \geq \varrho_{g_i}(T_{g_i}^{M>})\right] \geq 1 - O(1/n^2). \tag{7.4}$$

Combining Equations 7.2, 7.4, and 7.3, we have

$$Pr_D\left[O(\log^2 n) c_i(T_{g_i}^{M>}) \geq \varrho_{g_i}(T_{g_i}^{M>})\right] \geq 1 - O(1/n^2).$$

Since $c_i(e) = r_i x_i(e)$ and $x_i(e)$ is increasing in i, $r_i x_k(e) \geq c_i(e)$. Therefore,

$$Pr_D\left[O(\log^2 n) r_i x_k(T_{g_i}^{M>}) \geq \varrho_{g_i}(T_{g_i}^{M>})\right] \geq 1 - O(1/n^2).$$

■

To provide the same bounds for $M_{g_i}^{<}$ we need the following Chernoff bound lemma.

Lemma 7.4.16 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$. Furthermore, let $(C, v) \in \mathcal{C}_i$, $\varrho_i(C) < 12 r_i \log n$ and $s_i \notin C$.*

Let $M_{g_i}^{<}$ be defined as in Definition 7.4.13 for any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Define $M_{g_i}^{<}|C$ be the nodes of $M_{g_i}^{<}$ that are also in cluster C. Then,

$$Pr_D\left[\varrho_{g_i}(T_{g_i}^{M<}|C) = r_i O(\log n)\right] \geq 1 - O(1/n^3).$$

Proof. For any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$, define A to be the number of nodes in cluster C that issue a join request. Thus, A is a random variable based on the distribution D over request sequences. Let E be the expected number that issue a JOIN request. $E = \varrho_i(C)/r_i < 12 \log n$. Hence, by Lemma 7.4.8, $A \leq O(\log n)$ with probability at least $1 - O(1/n^3)$. The lemma now follows from the observation that $\varrho_{g_i}(T^{M<}_{g_i}|C)/r_i \leq A$. ∎

Lemma 7.4.17 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$.*

Let $M^<_{g_i}$ be defined as in Definition 7.4.13 for any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Let $T^{M<}_{g_i}$ be any tree spanning $M^<_{g_i}$. Then,

$$Pr_D\left[\varrho_{g_i}(T^{M<}_{g_i}) \leq O(\log^2 n) r_i x_k(T^{M<}_{g_i})\right] \geq 1 - O(1/n^2).$$

Proof. Recall that the clusters in $\mathcal{C}_i$ can be colored with $O(\log n)$ colors so that the minimum cost path between any two clusters of the same color is $\Omega(r_i \log n)$, where the cost of a link is given by c_i. Consider any extension of request sequence prefix $\sigma_0 \ldots \sigma_i$. Consider $M^<_{g_i}$ for that extension. Clearly, there exists some color, say red, such that a $\Omega(\log n)$ fraction of the nodes in $M^<_{g_i}$ are elements of clusters with the color red. Denote by $R^<_{g_i}$ the nodes in $M^<_{g_i}$ that are elements of red clusters. Furthermore, define $R^{<c}_{g_i}$ to be the center nodes of the red clusters which have elements in $R^<_{g_i}$. Define $T^{R<}_{g_i}$ to be the subtree of $T^{M<}_{g_i}$ spanning the nodes in $R^<_{g_i}$.

Let $T^{R<c}_{g_i}$ be the minimum cost (using cost function c_i) tree spanning s_i and the nodes in $R^{<c}_{g_i}$. Since no element of $R^{<c}_{g_i}$ is an element of a cluster containing s_i and since the minimum cost path between any two clusters of the same color is $\Omega(r_i \log n)$, we conclude that $c_i(T^{R<c}_{g_i}) \geq |R^{<c}_{g_i}|\Omega(r_i \log n)$. To consider the benefit associated with $T^{R<}_{g_i}$ we can no longer consider any extension of request sequence prefix $\sigma_0 \ldots \sigma_i$. Rather, a probabilistic argument is needed. By definition of $R^<_{g_i}$, Lemma 7.4.16, and the fact that there are at most n clusters, $Pr_D\left[|R^{<c}_{g_i}|O(r_i \log n) \geq \varrho_{g_i}(T^{R<}_{g_i})\right] \geq 1 - O(1/n^2)$. Thus,

$$Pr_D\left[O(1)c_i(T^{R<c}_{g_i}) \geq \varrho_{g_i}(T^{R<}_{g_i}).\right] \geq 1 - O(1/n^2). \tag{7.5}$$

To complete the proof, we relate the terms in this inequality back to $T^{M<}_{g_i}$. The following discussion holds for any extension of request sequence prefix $\sigma_0 \ldots \sigma_i$. First consider the benefit.

By construction,

$$O(\log n)\varrho_{g_i}(T_{g_i}^{R<}) \geq \varrho_{g_i}(T_{g_i}^{M<}) \tag{7.6}$$

Now consider the cost. Since the minimum cost path between any two clusters of the same color is $\Omega(r_i \log n)$ and the diameter of each cluster is $O(r_i \log^2 n)$, $c_i(T_{g_i}^{R<c}) \leq O(\log n) c_i(T_{g_i}^{R<})$. Furthermore, since $T_{g_i}^{R<}$ is a subtree of $T_{g_i}^{M<}$, $c_i(T_{g_i}^{R<}) \leq c_i(T_{g_i}^{M<})$. Thus,

$$O(\log n) c_i(T_{g_i}^{M<}) \geq c_i(T_{g_i}^{R<c}). \tag{7.7}$$

Combining Equations 7.5, 7.6, and 7.7, we have

$$Pr_D\left[O(\log^2 n) c_i(T_{g_i}^{M>}) \geq \varrho_{g_i}(T_{g_i}^{M>})\right] \geq 1 - O(1/n^2).$$

Since $c_i(e) = r_i x_i(e)$ and $x_i(e)$ is increasing in i, $r_i x_k(e) \geq c_i(e)$. Therefore,

$$Pr_D\left[O(\log^2 n) r_i x_k(T_{g_i}^{M<}) \geq \varrho_{g_i}(T_{g_i}^{M<})\right] \geq 1 - O(1/n^2).$$

∎

Lemma 7.4.18 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$.*

Let M_{g_i} be defined as in Definition 7.4.11 for any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Let $T_{g_i}^M$ be any tree spanning M_{g_i}. Then,

$$Pr_D\left[\varrho_{g_i}(T_{g_i}^M) \leq O(\log^2 n) r_i x_k(T_{g_i}^M))\right] \geq 1 - O(1/n^2).$$

Proof. Consider any extension of request sequence prefix $\sigma_0 \ldots \sigma_i$. Consider M_{g_i}, $M_{g_i}^<$ and $M_{g_i}^>$ for that extension. Since the cluster set $\mathcal{C}_i$ covers the graph G and Lemma 7.4.6 shows that all nodes that are in the same cluster with the source node, s_i, are accepted, we conclude that $M_{g_i} = M_{g_i}^< \cup M_{g_i}^>$.

Let $T_{g_i}^{M<}$ be the subtree of $T_{g_i}^M$ that spans the nodes in $M_{g_i}^<$. Let $T_{g_i}^{M>}$ be the subtree of $T_{g_i}^M$ spans the nodes in $M_{g_i}^>$. Then,

$$2 r_i x_k(T_{g_i}^M) \geq r_i x_k(T_{g_i}^{M>}) + r_i x_k(T_{g_i}^{M<}).$$

The lemma now follows from Lemmas 7.4.15 and 7.4.17 and the fact that

$$\varrho_{g_i}(T_{g_i}^{M>}) + \varrho_{g_i}(T_{g_i}^{M<}) \geq \varrho_{g_i}(T_{g_i}^{M}).$$

∎

Now we can provide an upper bound on the total rejected benefit in terms of cost function $x_k(e)b(e)$.

Lemma 7.4.19 *Let $D \in \mathcal{D}$ be a probability distribution over request sequences.*

For any request sequence from distribution D let $\mathcal{I}$ be the set of group identifier used in that request sequence. For any on-line group $g_i \in \mathcal{I}$, define M_{g_i} as in Definition 7.4.11. For any batched group $g_i \in \mathcal{I}$, let M_{g_i} be the empty set unless, NOMG *rejected the group g_i and the optimal off-line algorithm accepted the group g_i. In that case, M_{g_i} consists of the members of the multicast group, i.e., the set $\{v \mid \varrho_j(v) \neq 0$ where $j = \sigma(v, g_i)\}$. Then,*

$$Pr_D\left[O(\log^2 n)\sum_{e\in E} x_k(e)b(e) \geq \sum_{i\in\mathcal{I}} \varrho_{g_i}(M_{g_i}).\right] \geq 1 - O(1/n).$$

Proof. Consider a specific request sequence σ and the corresponding result sequence $\rho =$ NOMG(σ). First we construct a rejection sequence γ from σ and ρ. Consider multicast group g_i. Define $\gamma_{g_i} = (M_{g_i}, r_{g_i}, \varrho_{g_i})$ as follows. M_{g_i} is defined as in the statement of the lemma. Furthermore, r_{g_i} gives the bandwidth requirement of the multicast group. Finally, the benefit function ϱ_{g_i} is just ϱ_{g_i} (cf. Definition 7.4.5).

Let $T_{g_i}^M$ be the spanning tree for M_{g_i} use by the off-line algorithm. Since the trees $T_{g_i}^M$ are accepted by the off-line algorithm, the correctness of that algorithm implies that the capacity constraints are not violated, i.e., for all $e \in E$, $\frac{1}{b(e)}\sum_{i\in\mathcal{I}|e\in T_{g_i}^M} r_{g_i} \leq 1$.

Now consider any γ_{g_i} for a batched multicast group. Let j be the index of the request for the multicast group, i.e., $j = \sigma(v, g_i)$ for some $v \in T_{g_i}^M$. Since NOMG rejected the request σ_j, the Steiner Tree found by the MCST algorithm $T_{g_j}^j$ has cost greater than $2K_1 \log^2 n$ times its benefit. Specifically, $c_j(T_{g_j}^j) > 2K_1 \log^2 n\varrho_j(T_{g_j}^j) = 2K_1 \log^2 n\varrho_j(T_{g_i}^M)$. (Recall, by construction, $g_i = g_j$ and $T_{g_i}^M$ is the tree used by the optimal off-line algorithm to span the multicast members spanned by $T_{g_j}^j$.) The MCST algorithm guarantees that the cost of $T_{g_j}^j$ is at most twice that of the minimum cost spanning tree. Thus,

$$c_j(T_{g_i}^M) \geq 1/2c_j(T_{g_j}^j) > K_1 \log^2 n\varrho_j(T_{g_i}^M).$$

Since $c_j(e) = r_j x_j(e)$ and $x_j(e)$ is increasing in j, $r_j x_k(e) \geq c_j(e)$. Therefore, $r_j x_k(T_{g_i}^M) > K_1 \log^2 n \varrho_j(T_{g_i}^M)$. By definition of ϱ_{g_j}, $\varrho_j = \varrho_{g_j}$. Thus, $g_i = g_j$ implies that $\varrho_j = \varrho_{g_i}$. Furthermore, let r_{g_i} be the bandwidth required by multicast group g_i. Thus, $r_j = r_{g_i}$. Therefore,

$$r_{g_i} x_k(T_{g_j}^M) > K_1 \log^2 n \varrho_{g_i}(T_{g_i}^M).$$

Now consider any γ_{g_i} for an on-line multicast group. In this case, the discussion no longer holds for every request sequence. Rather, the discussion is probabilistic. Let r_{g_i} be the required bandwidth of broadcast group g_i. Lemma 7.4.18 states that

$$Pr_D\left[\varrho_{g_i}(T_{g_i}^M) \leq O(\log^2 n) r_{g_i} x_k(T_{g_i}^M)\right] \geq 1 - O(1/n^2).$$

Furthermore, since there are at most n on-line multicast groups in any request sequence, $\varrho_{g_i}(T_{g_i}^M) \leq O(\log^2 n) r_{g_i} x_k(T_{g_i}^M)$ for *all* multicast groups with probability at least $1 - O(1/n)$.

By inspection, the x_k function of Lemma 7.3.9 corresponds to the x_k function of the NOMG algorithm. Since $\varrho_i(T_{g_i}^M) < O(\log^2 n) r_{g_i} x_k(T_{g_i}^M)$ for all g_i with probability at least $1 - O(1/n)$, we can apply Lemma 7.3.9 to conclude that

$$Pr_D\left[O(\log^2 n) \sum_{e \in E} x_k(e) b(e) \geq \sum_{i \in \mathcal{I}} \varrho_{g_i}(T_{g_i}^M)\right] \geq 1 - O(1/n).$$

The lemma now follows from the observation that $\varrho_{g_i}(T_{g_i}^M) \geq \varrho_{g_i}(M_{g_i})$. ■

Using Lemmas 7.4.10 and 7.4.19 we can now prove the competitive ratio for the NOMG algorithm. Recall that the goal of the algorithm is to maximize the amount of accepted benefit (cf. Definition 7.3.2).

Theorem 7.4.20 *Let $\mathcal{P}$ be the non-interleaved on-line multicast admission control and routing problem for general topology networks. Let P measure the amount of accepted benefit. Then, the* NOMG *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(\text{NOMG})$, of $O(\log^6 n)$ with probability at least $1 - O(1/n)$.*

Proof. Define σ_{k-1} to be the final request of any request sequence from any distribution $D \in \mathcal{D}$. Lemma 7.4.10 states that

$$Pr_D\left[O(\log^4 \mu) \sum_{j} \varrho_{g_j}(T_{g_j}^k) \geq \sum_{e \in E} x_k(e) b(e)\right] \geq 1 - O(1/n).$$

where j ranges over the indices of all BATCH and JOIN requests. Thus, for any distribution $D \in \mathcal{D}$, the distribution D picks, with probability at least $1 - O(1/n)$, a request sequence σ such that the benefit accepted by the NOMG algorithm, $P(\sigma, \text{NOMG}(\sigma))$, is bounded from below by $1/O(\log^4 \mu) \sum_{e \in E} x_k(e)b(e)$.

Now consider the requests that were rejected by the NOMG algorithm but accepted by the optimal off-line algorithm. Specifically, Define $\mathcal{I}$ and M_{g_i} for each $g_i \in \mathcal{I}$ as in Lemma 7.4.18. Lemma 7.4.18 states that

$$Pr_D \left[\sum_{i \in \mathcal{I}} \varrho_{g_i}(T_{g_i}^M) \leq O(\log^2 n) \sum_{e \in E} x_k(e)b(e) \right] \geq 1 - O(1/n).$$

for any distribution $D \in \mathcal{D}$, the distribution D picks, with probability at least $1 - O(1/n)$, a request sequence σ such that the benefit accepted by the optimal off-line algorithm, but rejected by the NOMG algorithm is bounded from above by $O(\log^2 n) \sum_{e \in E} x_k(e)b(e)$.

With probability at least $1 - O(1/n)$, $P(\sigma, \text{NOMG}(\sigma)) + O(\log^2 n) \sum_{e \in E} x_k(e)b(e)$ is an upper bound on the total benefit accepted by the optimal off-line algorithm. Therefore, with probability at least $1 - O(1/n)$ the competitive ratio of the NOMG algorithm is less than:

$$\begin{aligned} \frac{P_o(\sigma)}{P(\sigma, \text{NOMG}(\sigma))} &\leq \frac{P(\sigma, \text{NOMG}(\sigma)) + O(\log^2 n) \sum_{e \in E} x_k(e)b(e)}{P(\sigma, \text{NOMG}(\sigma))} \\ &\leq \frac{P(\sigma, \text{NOMG}(\sigma)) + O(\log^2 n) O(\log^4 \mu) P(\sigma, \text{NOMG}(\sigma))}{P(\sigma, \text{NOMG}(\sigma))} = O(\log^6 \mu). \end{aligned}$$

The last step follows from the fact that $\mu = 2n(2K_1K_2K_3 n \log^3 n + 1)$. ■

We now prove the correctness of the NOMG algorithm. In other words, we show that NOMG solves the non-interleaved on-line multicast problem. We first consider the capacity constraint.

Lemma 7.4.21 *Let $\sigma = \sigma_0\sigma_1 \ldots \sigma_{k-1}$ be the request sequence and $\rho = \text{NOMG}(\sigma)$ be the corresponding result sequence. For each $i \leq k$ and each link $e \in E$, $u_i(e) \leq 1$.*

Proof. Recall that $\mu = 2n(2K_1K_2K_3 n \log^3 n + 1)$.

We proceed by contradiction. Let i be the first index such that $u_{i+1}(e) > 1$ for some link e. From the definition of $u_{i+1}(e)$ (cf. Figures 7-2, 7-3, and 7-4), $u_i(e) > 1 - r_i/b(e)$. Since we assume that $r_i \leq b(e)/(\log \mu)$, we conclude that $u_i(e) > 1 - 1/(\log \mu)$. Thus:

$$x_i(e) = (\mu^{u_i(e)} - 1)/n > (\mu^{1-1/(\log \mu)} - 1)/n = (\mu/2 - 1)/n = 2K_1K_2K_3 n \log^3 n.$$

Therefore, $c_i = r_i x_i(e) > r_i 2K_1K_2K_3 n \log^3 n$. This leads to a contradiction for each type of request.

First consider an INIT request. Since each node is an element of at most $K_2 \log n$ clusters and there are at most n nodes in the network, $\varrho_i'(T_{g_i}^{i+1}) \leq K_2 r_i n \log n$. Furthermore, the $K_3 \log^2 n$ sparsity of the tree $T_{g_i}^{i+1}$ implies that $K_3 \log^2 n \varrho_i'(T_{g_i}^{i+1}) \geq c_i(T_{g_i}^{i+1})$. Thus,

$$r_i K_2 K_3 n \log^3 n > r_i 2K_1K_2K_3 n \log^3 n$$

which is a contradiction.

Next consider a JOIN request. For a JOIN request, if $\rho \neq \perp$, then $c_i(p) \leq r_i K_1 \log^2 n$ (cf. Figure 7-3). This contradicts the fact that $c_i(e) > r_i 2K_1K_2K_3 n \log^3 n$.

Finally consider a BATCH request. The benefit of a batch request, $\varrho_i(T_{g_i}^i)$, is bounded by $r_i n$. The fact that the multicast group was accepted implies that $2K_1 r_i n \log^2 n \geq c_i(T_{g_i}^i)$. This contradicts the fact that $c_i(e) > r_i 2K_1K_2K_3 n \log^3 n$. ■

Theorem 7.4.22 *The* NOMG *algorithm solves the non-interleaved on-line multicast admission control and routing problem for general topology networks.*

Proof. The theorem follows directly from Lemma 7.4.21, the definition of the NOMG algorithm (Figures 7-2, 7-3, and 7-4), and the definition non-interleaved on-line multicast admission control and routing problem for general topology networks (Definition 7.4.1). ■

7.5 Interleaved On-line Multicast Groups

This section presents an admission control and routing algorithm for batched and on-line multicast groups. It extends the NOMG algorithm by removing the restriction that request sequences must be non-interleaved. We call the algorithm IOMG (Interleaved On-line Multicast Groups). The service model for the IOMG algorithm is the same as that for the NOMG algorithm. In particular, the algorithm uses binary admission control for the batched multicast groups and non-binary admission control for the on-line multicast groups. On-line multicast groups also specify a "source" node that is guaranteed membership in the multicast group. (Definition 7.5.1 formally defines the admission control and routing problem for interleaved on-line multicast groups.)

Our algorithm has a competitive ratio of $O(\log^6 n)$, where n is the number of nodes in the network. Since unicast communication is a special case of multicast communication, the $\Omega(\log n)$ lower bound of [AAP93] applies to our algorithm (cf. Lemma 7.5.4).

7.5.1 Interleaved Requests

The following example illustrates why the NOMG algorithm cannot handle interleaved request sequences. Consider a multicast group g_i and a cluster C such that the NOMG algorithm reserves a path to the cluster C center node v in the tree T_{g_i} constructed as a result of the INIT request for group g_i. The path to the center node v is constructed in reliance on two facts. First, we rely on the fact that enough JOIN requests are likely to arrive in the cluster. Second, we rely on the fact that the nodes issuing the JOIN requests will be able to find a path that connects them to the multicast tree. The first fact is guaranteed by the probabilistic assumptions (cf. Lemma 7.4.9 for the NOMG algorithm). For the NOMG algorithm the second fact is guaranteed by Lemma 7.4.6. (The proof of this lemma rests on the fact that the request sequences for the NOMG algorithm are non-interleaved.) However, for the IOMG algorithm, the second fact cannot be relied upon since the proof of Lemma 7.4.6 is no longer valid. In particular, requests from other multicast groups can cause the links in cluster C to be 100% utilized before the JOIN requests in cluster C for group g_i arrive. Thus, the nodes issuing the JOIN requests in cluster C for group g_i are prevented from connecting to the multicast tree for group g_i. In particular, there may no longer be a path with sufficient bandwidth from a node issuing a JOIN request to the center node v of cluster C. In this event, the path reserved to the center node v of cluster C in tree T_{g_i} may no longer be justifiable. (I.e. Lemma 7.4.10 which relies on Lemma 7.4.6 would fail to hold.) To address this problem, the IOMG algorithm reserves some bandwidth on paths in cluster C to ensure that the nodes issuing JOIN requests in cluster C for group g_i will be able to connect to the center node v of cluster C. The amount of bandwidth reserved by the IOMG algorithm for any particular node is proportional to the probability that the node will actually issue a JOIN request for the multicast group. In particular, for each multicast group, the IOMG algorithm reserves for each node the excepted amount of bandwidth that the node will need in order to connect to the multicast group.

7.5.2 Problem Statement

We provide a formal definition for the admission control and routing problem for interleaved on-line multicast groups. The problem is the same as the admission control and routing problem for non-interleaved on-line multicast groups (cf. Definition 7.4.1) except that the restriction to non-interleaved request sequences is removed and the constant μ is different. The admission control and routing problem for interleaved on-line multicast groups is a probabilistic problem.

Definition 7.5.1 (interleaved on-line multicast for a set of graphs $\mathcal{G}$) *Let $\mathcal{G}$ be set of graphs. The* interleaved on-line multicast admission control and routing problem for $\mathcal{G}$ *is the same as the non-interleaved on-line multicast admission control and routing problem for $\mathcal{G}$ with the following modifications. The constant $\mu = (2K_1K_2K_3|V|^2 \log^3 |V| + 4)^{12}$. Furthermore, the ninth condition on S now is: for all $i \in [1, |\sigma|)$, if $\sigma_i \in Q_4$ then there exist $\sigma_j \in Q_3$ such that $j < i$, $g(\sigma_j) = g(\sigma_i)$, and $r(\sigma_j) = r(\sigma_i)$. (In other words, the condition that $g(\sigma_k) = g(\sigma_i)$ for all $k \in (j, i)$ is eliminated.)*

We use the performance function in Definition 7.3.2 that measures the accepted value.

As with admission control and routing for non-interleaved on-line multicast groups, the lower bounds for admission control and routing for batched multicast groups and the lower bounds for unicast admission control and routing extend to admission control and routing for interleaved on-line multicast groups. We first show that the lower bounds for non-interleaved multicast groups extend to interleaved multicast groups.

Lemma 7.5.2 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$. Let $\mathcal{P}'(\{\mathcal{G}\})$ be the non-interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2 \log^3 |V| + 4)^{12}$. Let P be the performance function of Definition 7.3.2.*

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve problem $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Proof. Notice that the non-interleaved on-line multicast admission control and routing problem is a restricted form of the interleaved on-line multicast admission control and routing problem.

Now the lemma follows from Lemma 2.4.4 and Lemma 2.5.2. ■

As an immediate consequence of Lemma 7.4.2 and Lemma 7.5.2, we conclude that lower bounds for batched multicast admission control and routing extend to admission control and routing for interleaved on-line multicast groups.

Lemma 7.5.3 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$. Let $\mathcal{P}'(\{\mathcal{G}\})$ be the batched multicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V|+4)^{12}$. Let P be the performance function of Definition 7.3.2.*

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve problem $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Proof. The lemma follows immediately from Lemma 7.4.2 and Lemma 7.5.2. ■

As an immediate consequence of Lemma 7.4.3 and Lemma 7.5.2, we conclude that lower bounds for unicast admission control and routing extend to admission control and routing for interleaved on-line multicast groups.

Lemma 7.5.4 *Let $\mathcal{P}(\{\mathcal{G}\})$ be the interleaved on-line multicast admission control and routing problem for the set of graphs $\mathcal{G}$. Let $\mathcal{P}'(\{\mathcal{G}\})$ be the unicast admission control and routing problem for the set of graphs $\mathcal{G}$ where $\mu = (2K_1K_2K_3|V|^2\log^3|V|+4)^{12}$. Let P be the performance function of Definition 7.3.2. Let P' be the performance function of Definition 2.3.7.*

If $\mathcal{C}_{\mathcal{P}'(\{G\}),P'}(A') \geq K$ for all deterministic on-line algorithms A' that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}_{\mathcal{P}(\{G\}),P}(A) \geq K$ for all deterministic on-line algorithms A that solve $\mathcal{P}(\{G\})$. Similarly, if $\mathcal{C}^b_{\mathcal{P}'(\{G\}),P'}(A'_r) \geq K$ for all randomized on-line algorithms A'_r that solve $\mathcal{P}'(\{G\})$, then $\mathcal{C}^b_{\mathcal{P}(\{G\}),P}(A_r) \geq K$ for all randomized on-line algorithms A_r that solve $\mathcal{P}(\{G\})$.

Both statements also hold if P' is the the performance function in Definition 2.3.8.

Proof. The lemma follows immediately from Lemma 7.4.3 and Lemma 7.5.2. ■

7.5.3 Algorithm

Recall that the utilization function u_i represents the utilization of the network just before request σ_i is handled. The NOMG algorithm uses the u_i function in the cost function c_i used to handle request σ_i. In contrast, the IOMG algorithm uses a slightly different utilization function, u_i^r, for the cost function c_i used to handle request σ_i. Informally, the difference between function u_i and function u_i^r is that u_i^r tracks the *expected* utilization due to some of the requests rather than the actual utilization due to those requests.

Description. We consider each type of request separately. We start with an INIT request $\sigma_i = (g_i, \varrho_i, r_i, s_i)$. There are three additional steps performed by the IOMG algorithm for an INIT request. (Compare Figures 7-2 and 7-5.) First, the IOMG algorithm updates both the u_i and the u_i^r functions for the links that are included in T_{g_i}. Second, for each cluster $(C, v) \in \mathcal{C}_i$, where the center node v is included in T_{g_i}, the algorithm constructs a shortest path tree, T_i^C, that is rooted at v and spans the nodes $u \in C$ with $\varrho_i(u) \neq 0$. The shortest path tree computation is done by the SPTREE procedure. (The SPTREE procedure can use any standard shortest path tree algorithm.) The inputs to the SPTREE algorithm are the cluster C, the root v, the set of nodes that must be spanned (specified by ϱ_i), and the cost function c. Consider some node u in the shortest path tree T_i^C. When node u issues a JOIN request for group g_i it will use the path that connects u to the center node v in order to connect to the group g_i multicast tree (see the discussion of JOIN request for IOMG algorithm). The shortest path trees for each of the clusters are combined into a forest F_{g_i}. The third additional step performed by the IOMG algorithm for an INIT request considers clusters $(C, v) \in \mathcal{C}_i$ where v is not included in T_{g_i} and $s_i \in C$. These clusters are merged into a subgraph C'. Then the algorithm constructs a shortest path tree, $T_i^{C'}$, that is rooted at s_i and spans the nodes $u \in C'$ with $\varrho_i(u) \neq 0$. The shortest path tree is added to the forest F_{g_i}. For each link in F_{g_i}, the IOMG algorithm updates the value of u_i^r as follows. Define the quantity $\varrho_i(e, F_{g_i})$ as the minimum of r_i and the sum of $\varrho_i(u)$ for all nodes u that use link e on a path to the root node of a tree in F_{g_i}. Then the IOMG algorithm adds $\frac{\varrho_i(e,F_{g_i})}{b(e)}$ to $u^r(e)$. Informally, $\frac{\varrho_i(e,F_{g_i})}{b(e)}$ represents the expected change in utilization of link e due to future JOIN requests from nodes in cluster C that use link e to reach a root node in F_{g_i}. Since the multicast group does not transmit along links in F_{g_i} until an actual JOIN request requires such links, the utilization function u is not updated. However, the utilization function u will be updated as part of a JOIN request.

Now consider a JOIN request. The IOMG algorithm breaks the nodes issuing JOIN requests into two groups: nodes that are an element of the forest F_{g_i} and those that are not. Nodes that are not an element of F_{g_i} are handled as in the NOMG algorithm except that the utilization function u^r must also be updated. (Compare Figures 7-3 and 7-6.) Now consider a JOIN request $\sigma_i = (g_i, \varrho_i, r_i)$ where $v_i = v(\sigma_i)$ and v_i is an element of $F^i_{g_i}$. IOMG use the FPATH procedure to determine a path for v_i. Specifically, the FPATH procedure takes as input the forest $F^i_{g_i}$ and the multicast tree $T^i_{g_i}$. It determines a path p from v_i to a node in the current multicast tree $T^i_{g_i}$ such that the path p is in one of the trees of $F^i_{g_i}$. By construction of $F^i_{g_i}$ and $T^i_{g_i}$, such a path is guaranteed to exist. Since the node v_i can be a member of more than one tree in the forest $F^i_{g_i}$, there may be multiple paths. The FPATH procedure picks one of the paths arbitrarily. (The implementation of the FPATH procedure is trivial. We do not provide the details.) Finally, the utilization functions u and u^r is updated for path p.

Now consider a BATCH request. For a BATCH request, the IOMG algorithm works exactly like the NOMG algorithm except that the utilization function u^r is also updated. (Compare Figures 7-4 and 7-7.)

7.5.4 Analysis

The analysis consists of two parts: correctness and complexity. We consider complexity first. The complexity analysis parallels that of the NOMG algorithm. In fact, most of the lemmas in Section 7.4.4 apply to the IOMG algorithm with no modification except that the utilization function u^r should be used in place of the utilization function u. In this section, we will only restate the lemmas that actually change for the IOMG algorithm. However, even the lemmas that change tend to be very similar to the version in Section 7.4.4. Therefore, the proofs will only mention the points at which the lemmas in this section differ from the versions in Section 7.4.4.

The proof of the following lemma differs entirely from the proof of the corresponding lemma, Lemma 7.4.6, for the NOMG algorithm. Recall that the proof of Lemma 7.4.6 is based on the fact that the request sequences for the NOMG algorithm are non-interleaved.

IOMG-INIT$(g_i, \varrho_i, r_i, s_i)$:
  for all $e \in E:\ c(e) = r_i x(e)$; (where $x(e) = (\mu^{u^r(e)} - 1)/n$)
  $\mathcal{C}_i = \text{CLUSTER}(G, c, r_i \log n)$;
  for all $v \in V$:
    $\varrho'_i(v) = \sum_{v' \in C} \varrho_i(v')$ **if** $(C, v) \in \mathcal{C}_i$ and $\sum_{v' \in C} \varrho_i(v') \geq 12 r_i \log n$;
    $\varrho'_i(v) = 0$ otherwise;
  $T_{g_i} = \text{MAXSPARSE}(G, c, \varrho', s_i)$;
  for all $(C, v) \in \mathcal{C}_i$ s.t. $v \in T_{g_i}$
    $T^C_{g_i} = \text{SPTREE}(C, v, \varrho_i, c)$;
    $F_{g_i} = F_{g_i} \cup T^C_{g_i}$;
  $C' = \cup_{(C,v) \in \mathcal{C}_i | v \notin T_{g_i}, s_i \in C} C$;
    $T^{C'}_{g_i} = \text{SPTREE}(C', s_i, \varrho_i, c)$;
    $F_{g_i} = F_{g_i} \cup T^{C'}_{g_i}$;
  for all $e \in E$:
    $u^r(e) = u^r(e) + \frac{r_i}{b(e)}$ **if** $e \in T_{g_i}$;
    $u^r(e) = u^r(e) + \frac{\varrho_i(e, F_{g_i})}{b(e)}$ **if** $e \notin T_{g_i}$ and $e \in T_{g_i}$; (*)
    $u(e) = u(e) + \frac{r_i}{b(e)}$ **if** $e \in T_{g_i}$;

Figure 7-5: The IOMG admission control and routing algorithm an INIT request.

IOMG-JOIN(g_i, ϱ_i, r_i):
  for all $e \in E:\ c(e) = r_i x(e)$; (where $x(e) = (\mu^{u^r(e)} - 1)/n$)
  $v_i = v \in G$ s.t. $\varrho_i(v) \neq 0$;
  if $v \in F_g$ **then** path $p = \text{FPATH}(F_{g_i}, T_{g_i})$;
    for all $e \in p:\ \ u(e) = u(e) + \frac{r_i}{b(e)}$;
    $T_{g_i} = T_{g_i} \cup p$;
  else
    if there exists a path p in $G = (V, E)$ from v_i to T_{g_i} s.t. $\sum_{e \in p} c(e) \leq K_1 r_i \log^2 n$;
    then route the requesting member on p, and set:
      for all $e \in p$:
        $u^r(e) = u^r(e) + \frac{r_i}{b(e)}$;
        $u(e) = u(e) + \frac{r_i}{b(e)}$;
      $T_{g_i} = T_{g_i} \cup p$;
    else reject the requesting member and set:

Figure 7-6: The IOMG admission control and routing algorithm for a JOIN request.

```
IOMG-BATCH(g_i, ϱ_i, r_i):
  for all e ∈ E : c(e) = r_i x(e); (where x(e) = (μ^{u^r(e)} − 1)/n)
  V′ = {v ∈ V | ϱ_i(v) ≠ 0};
  T_{g_i} = MCST(G, V′, c);
  if c(T_{g_i}) ≤ 2K_1 log^2 n ϱ_i(T_{g_i})
  then route the requested multicast group on T_{g_i}, and set:
    for all e ∈ T_{g_i} :
      u^r(e) = u^r(e) + r_i/b(e);
      u(e) = u(e) + r_i/b(e);
  else reject the requested multicast group and set:
```

Figure 7-7: The IOMG admission control and routing algorithm for a BATCH request.

Lemma 7.5.5 *Let $\sigma = \sigma_0 \ldots \sigma_{k-1}$ be a request sequence and $\rho = $ NOMG(σ) be the corresponding result sequence. Let $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ be an* INIT *request. Consider a cluster that either contains the source or whose center node is included in $T_{g_i}^{i+1}$. In other words, consider C such that $(C, v) \in \mathcal{C}_i$ and either $s_i \in C$ or $v \in T_{g_i}^{i+1}$.*

Now consider a JOIN *request for group g_i from a node in C. In other words, consider* JOIN *request $\sigma_j = (g_j, \varrho_j, r_j)$ such that $g_j = g_i$ and $v_j = v(\sigma_j) \in C$.*

Then, the NOMG *algorithm will accept v_j, i.e., $v_j \in T_{g_i}^k$ and $\rho_j \neq \perp$.*

Proof. Since v_j is an element of a cluster C such that $(C, v) \in \mathcal{C}_i$ and either $v \in T_{g_i}^{i+1}$ or C contains the source, we can conclude that $v_j \in F_{g_i}^{i+1}$. Now the definition of IOMG for the JOIN request implies that $v_j \in T_{g_i}^k$ and $\rho_j \neq \perp$. ∎

Lemma 7.5.6 *Let $\sigma_0 \ldots \sigma_i$ be a request sequence prefix where $\sigma_i = (g_i, \varrho_i, r_i, s_i)$ is an* INIT *request. Let $D \in \mathcal{D}$ be a probability distribution over request sequences consistent with $\sigma_0 \ldots \sigma_i$ (cf. Definition 7.4.1). Let $(C, v) \in \mathcal{C}_i$, $\varrho_i(C) \geq 12 r_i \log n$, and $v \in T_{g_i}^{i+1}$. Define σ_{k-1} to be the final request of any extension of the request sequence prefix $\sigma_0 \ldots \sigma_i$. Then,*

$$Pr_D\left[O(1)\varrho_{g_i}(T_{g_i}^k|C) \geq \varrho_i(C)\right] \geq 1 - O(1/n^3).$$

Proof. The proof is exactly the same as the proof of the corresponding NOMG lemma, Lemma 7.4.9, except that Lemma 7.5.5 is referenced place of Lemma 7.4.6. ∎

Lemma 7.5.7 *Let $D \in \mathcal{D}$ be a probability distribution over request sequences. Define σ_{k-1} to* *be the final request of any request sequence from distribution D. Then,*

$$Pr_D\left[O(\log^4\mu)\sum_j \varrho_{g_j}(T^k_{g_j}) \geq \sum_{e\in E} x_k(e)b(e).\right] \geq 1 - O(1/n^3),$$

where j ranges over the indices of all BATCH *and* JOIN *requests.*

Proof. The differences with the corresponding NOMG lemma, Lemma 7.4.10, arise only in the construction of the acceptance sequence. Specifically, $\omega_i = (T_i, R_i, \varrho_i)$ is defined differently for INIT requests and some JOIN requests.

Consider a JOIN request, $\sigma_i = (g_i, \varrho_i, r_i)$, first. The element ω_i is constructed differently than in Lemma 7.4.10 when the requesting node $v_i = v(\sigma_i)$ is an element of the forest $F^i_{g_i}$. In this case, the IOMG algorithm does not change the utilization function u^r_i (cf. Figure 7-6). As a consequence, we set $T_i = G_\emptyset$ and let $R_i(e) = 0$ for all $e \in E$. Furthermore, $\varrho_i = \varrho(\sigma_i)$.

Now consider an INIT request, $\sigma_i = (g_i, \varrho_i.r_i, s_i)$. For the IOMG algorithm, T_i is the union of $T^{i+1}_{g_i}$ and $F^{i+1}_{g_i}$. Furthermore, R_i is defined as follows. $R_i(e) = r_i$ for all $e \in T^{i+1}_{g_i}$, $R_i(e) = \varrho_i(e, F^{i+1}_{g_i})$ for all e such that $e \in F^{i+1}_{g_i}$ and $e \notin T^{i+1}_{g_i}$, and $R_i(e) = 0$ for all $e \notin T_i$. Furthermore, $\varrho_i(v) = \varrho'_i(v)$ when $v \neq s_i$ and $\varrho_i(v) = \sum_{u\in C|s_i\in C,(C,v)\in\mathcal{C}_i} \varrho(\sigma_i)(u)$ when $v = s_i$.

By inspection, the u_i function and x_i function of Lemma 7.3.6 correspond to the u^r_i function and x_i function of IOMG. Furthermore, R_i is defined such that $R_i(e) \leq b(e)/\log\mu$ for all $i \leq k$ and $e \in E$. In the case of JOIN requests, the c_i function of Lemma 7.3.6 correspond to the c_i function of IOMG. Thus, for the types of JOIN requests considered here, $c_i(T_i)/\varrho_i(T_i) \leq O(\log^2 n)$. (For the other types of JOIN requests, i.e., requests where $v(\sigma_i) \notin F^i_{g_i}$, see Lemma 7.4.10.)

For INIT requests, the c_i function of Lemma 7.3.6 differs slightly from the c_i function of IOMG. To simplify the discussion refer to the cost function of IOMG by c^{IOMG}_i and the benefit function of IOMG by ϱ^{IOMG}_i. Our goal is to conclude that $c_i(T_i)/\varrho_i(T_i) \leq O(\log^2 n)$ on the basis our information from the c^{IOMG}_i and ϱ^{IOMG}_i.

The difference between the cost functions depends on the link. The cost functions correspond for all links except those where $R_i(e) \neq r_i$. From the definition of $R_i(e)$, the links e for which $R_i(e) \neq r_i$ are the links that are in $F^{i+1}_{g_i}$ but not in $T^{i+1}_{g_i}$. For such links $c^{\text{IOMG}}_i(e) = r_i x_i(e)$ and $c_i(e) = \varrho^{\text{IOMG}}_i(e, F^{i+1}_{g_i})x_i(e)$. (Recall that $\varrho^{\text{IOMG}}_i(e, F^{i+1}_{g_i})$ represents the sum of $\varrho^{\text{IOMG}}_i(u)$ for all nodes u that use link e on their path to a root in the forest $F^{i+1}_{g_i}$.) Thus, $c_i(e) = \frac{\varrho^{\text{IOMG}}_i(e, F^{i+1}_{g_i})}{r_i} c^{\text{IOMG}}_i(e)$.

Physically, $\frac{\varrho_i^{\text{IOMG}}(e, F_{g_i}^{i+1})}{r_i}$ represents the expected number of nodes that both will issue a JOIN request and have e on their path to a root the forest $F_{g_i}^{i+1}$.

Consider cluster $(C, v) \in \mathcal{C}_i$ such that $v \in T_{g_i}^{i+1}$ and $T_i^C \subseteq F_{g_i}^{i+1}$ (cf. Figure 7-5). Let $p(u)$ be the path in tree T_i^C from u to v, the root of T_i^C. Now the definition of $\frac{\varrho_i^{\text{IOMG}}(e, F_{g_i}^{i+1})}{r_i}$ and the fact that T_i^C is a shortest path tree implies that

$$c_i(T_i^C) = \sum_{u \in T_i^C} \frac{\varrho_i^{\text{IOMG}}(u)}{r_i} c_i^{\text{IOMG}}(p(u)).$$

Since cluster C has diameter $O(r_i \log^2 n)$ based on cost function c_i^{IOMG}

$$c_i(T_i^C) \leq \sum_{u \in T_i^C} \varrho_i^{\text{IOMG}}(u) O(\log^2 n).$$

By definition, $\varrho_i'(v) = \sum_{u \in T_i^C} \varrho_i^{\text{IOMG}}(u)$, where v is the center node of cluster C. Thus, $c_i(T_i^C) \leq \varrho_i'(v) O(\log^2 n)$.

Now consider cluster $(C, v) \in \mathcal{C}_i$ such that $v \notin T_{g_i}^{i+1}$ but $s_i \in C$. Let C' be the union of all of these clusters (cf. Figure 7-5). Based on the code in Figure 7-5, $T_i^{C'} \subseteq F_{g_i}^{i+1}$. Let $p(u)$ be the path in tree $T_i^{C'}$ from u to s_i, the root of $T_i^{C'}$. Now the definition of $\frac{\varrho_i^{\text{IOMG}}(e, F_{g_i}^{i+1})}{r_i}$ and the fact that $T_i^{C'}$ is a shortest path tree implies that

$$c_i(T_i^{C'}) = \sum_{u \in T_i^C} \frac{\varrho_i^{\text{IOMG}}(u)}{r_i} c_i^{\text{IOMG}}(p(u)).$$

Since C' has diameter $O(r_i \log^2 n)$ based on cost function c_i^{IOMG}

$$c_i(T_i^{C'}) \leq \sum_{u \in T_i^{C'}} \varrho_i^{\text{IOMG}}(u) O(\log^2 n).$$

Now sum over all trees in the forest $F_{g_i}^{i+1}$. The construction of ϱ_i for the acceptance sequence and the definition of $T_{g_i}^{i+1}$ and $F_{g_i}^{i+1}$ imply that

$$c_i(F_{g_i}^{i+1}) \leq \varrho_i(T_{g_i}^{i+1}) O(\log^2 n). \tag{7.8}$$

For the links in $T_{g_i}^{i+1}$ the cost function c_i of Lemma 7.3.6 corresponds to the cost function c_i^{IOMG} of IOMG. Since the tree $T_{g_i}^{i+1}$ generated by the MAXSPARSE algorithm is $O(\log^2 n)$-sparse, the IOMG algorithm also guarantees that $c_i(T_{g_i}^{i+1})/\varrho_i'(T_{g_i}^{i+1}) \leq O(\log^2 n)$ for INIT requests. By construction of ϱ_i for the acceptance sequence, $\varrho_i'(T_{g_i}^{i+1}) = \varrho_i(T_{g_i}^{i+1})$. So,

$$c_i(T_{g_i}^{i+1}) \leq \varrho_i(T_{g_i}^{i+1}) O(\log^2 n). \tag{7.9}$$

Now Equations 7.9 and 7.8, along with the fact that $c_i(T_i) \leq c_i(F_{g_i}^{i+1}) + c_i(T_{g_i}^{i+1})$ imply that $c_i(T_i)/\varrho_i(T_i) \leq O(\log^2 n)$. Finally, we note that $max_e\{R_i(e)\} \leq \varrho_i(T_i)$.

The remainder of the proof proceeds exactly as in Lemma 7.4.10 except that references to Lemma 7.4.9 are replaced with references to Lemma 7.5.6. ∎

Theorem 7.5.8 *Let $\mathcal{P}$ be the interleaved on-line multicast admission control and routing problem for general topology networks. Let P measure the amount of accepted benefit. Then, the* IOMG *algorithm has a competitive ratio, $\mathcal{C}_{\mathcal{P},P}(\text{IOMG})$, of $O(\log^6 n)$ with probability at least $1 - O(1/n)$.*

Proof. The proof is exactly the same as the proof of the corresponding NOMG lemma, Lemma 7.4.20, except that Lemma 7.5.7 is referenced place of Lemma 7.4.10. ∎

We now prove the correctness of the IOMG algorithm. In other words, we show that IOMG solves the interleaved on-line multicast problem with high probability (cf. Definition 2.1.6). We first consider the capacity constraint.

Lemma 7.5.9 *Let $\sigma = \sigma_0\sigma_1 \ldots \sigma_{k-1}$ be the request sequence and $\rho = \text{NOMG}(\sigma)$ be the corresponding result sequence. For each $i \leq k$ and each link $e \in E$, $u_i(e) \leq 1/8$.*

Proof. Recall that $\mu = (2K_1K_2K_3n^2\log^3 n + 4)^{12}$.

Consider any request σ_i. For all request types, $u_{i+1}(e) - u_i(e) \leq r_i/b(e)$. Since $r_i \leq b(e)/\log\mu$, $u_{i+1}(e) - u_i(e) \leq 1/\log\mu$. Therefore, $u_{i+1}(e) - u_i(e) \leq 1/48$.

For all request types, the IOMG algorithm guarantees that $c_i(e) \leq 2K_1K_2K_3r_in\log^3 n$ (cf. proof of Lemma 7.4.21). Since $c_i(e) = r_ix_i(e) = r_i(\mu^{u_i(e)} - 1)/n$,

$$\begin{aligned}
(\mu^{u_i(e)} - 1)/n &\leq 2K_1K_2K_3n\log^3 n \\
\Rightarrow \quad u_i(e) &\leq \frac{\log(2K_1K_2K_3n^2\log^3 n + 1)}{\log\mu} \\
\Rightarrow \quad u_i(e) &\leq \frac{1}{12}
\end{aligned}$$

The lemma now follows from the fact that $u_{1+1} - u_i(e) \leq 1/48$ ■

Lemma 7.5.10 *Let $D \in \mathcal{D}$ be a probability distribution over request sequences. Define σ_{k-1} to be the final request of any request sequence from distribution D. Then, $Pr_D\left[u_k(e) \leq 1\right] \geq 1 - O(1/\mu)$.*

Proof.

Consider $u_k^r(e)$ for a particular request sequence σ and response sequence $\rho = \textsc{iomg}(\sigma)$. We divide $u_k^r(e)$ into two parts: $u_k^R(e)$ and $u_k^E(e)$ such that $u_k^r(e) = u_k^R(e) + u_k^E(e)$. In particular, $u_k^E(e)$ represents the additions to $u_k^r(e)$ due to the starred line of the IOMG algorithm in Figure 7-5. Thus, it represents the additions to $u_k^r(e)$ due to the shortest path forests F_{g_i}. Physically, $u_k^E(e)$ represents the expected amount of bandwidth added to link e due to the forests F_{g_i}. Define $u_k^R(e) = u_k^r(e) - u_k^E(e)$. Furthermore, define $u_k^A(e) = u_k(e) - u_k^R(e)$. Physically, $u_k^A(e)$ represents the actual amount of bandwidth added to link e (using JOIN requests) due to the forests F_{g_i}.

Thus, $u_k^A(e)$ is a random variable which is the weighted sum of indicator variables where the weight factor for each variable is at most $1/\log \mu$ (since $r_i \leq 1/\log \mu$). The expectation of that sum is given by $u_k^E(e)$. Furthermore, by Lemma 7.5.9, $u_k^E(e) \leq 1/8$. Now, define $A = u_k^A(e) \log \mu$ and $E = u_k^E(e) \log \mu$. Then, A is a random variable which is the weighted sum of indicator variables where the weight factor for each variable is less than 1. The expectation of that sum is given by E. Furthermore, $E \leq \frac{1}{8} \log \mu$. Based on the Chernoff bounds in [Rag86], we can conclude that

$$Pr\{A > 2eE\} \leq e^{-2E} \leq e^{-(2/8)\cdot \log \mu} \leq 1/\mu$$

Hence, $Pr_D[u_k^A(e) \leq 6u_k^E(e)] \geq 1 - O(1/\mu)$. Since $6u_k^E(e) \leq 6/8 \leq 3/4$, we conclude that $Pr_D[u_k^A(e) \leq 3/4] \geq 1 - O(1/\mu)$.

Based on Lemma 7.5.9, $u_k^R \leq 1/8$. The lemma now follows from the fact that $u_k^r(e) \leq u_k^R + u_k^A(e)$ and the fact that $Pr_D[u_k^A(e) \leq 3/4] \geq 1 - O(1/\mu)$. ■

Theorem 7.5.11 *The* IOMG *algorithm solves the interleaved on-line multicast admission control and routing problem for general topology networks with probability greater than $1 - O(1/\mu)$.*

Proof. The theorem follows directly from Lemma 7.5.10, the definition of the IOMG algorithm (Figures 7-5, 7-6, and 7-7), and the definition interleaved on-line multicast admission control and routing problem for general topology networks (Definition 7.5.1). ■

7.6 Towards Practical Multicast Algorithms

7.6.1 Introduction

In the introduction to this chapter we suggest that the multicast algorithms in this chapter can serve as the basis for practical multicast admission control and routing algorithms much in the same way that the AAP algorithm serves as the basis for the EXP algorithm, our practical unicast admission control and routing algorithm.

The BMG algorithm can serve as the basis for a practical admission control and routing algorithm for batched multicast groups. In particular, we believe that the BMG algorithm needs exactly the same modifications that are made to the AAP algorithm in constructing the EXP algorithm. Specifically, the constant μ used in the cost function c and the relationship between μ and the benefit function ϱ need to be modified. (See Chapter 5.)

The NOMG and IOMG algorithms can serve as the basis for practical admission control and routing algorithms for on-line multicast groups. However, the modifications needed for the NOMG and IOMG algorithms are significantly more extensive than those needed for the BMG algorithm. This section focuses on the issues related to using the NOMG and IOMG algorithms as the basis for practical multicast admission control and routing algorithms for on-line multicast groups.

The remainder of this section is organized as follows. Section 7.6.2 discusses the $O(\log^6 n)$ competitive ratio achieved by the NOMG and IOMG algorithms. The section explains why we expect the ratio of the performance of an optimal off-line algorithm to the performance of the NOMG and IOMG algorithms to be much better than $O(\log^6 n)$ in practice. Section 7.6.3 highlights some of the key concepts that used by the NOMG and IOMG algorithms and that we also expect to use in practical versions of the algorithms. Finally, Section 7.6.4 discusses areas in which the NOMG and IOMG algorithms must be modified to improve their performance in practice. We do not provide detailed descriptions of practical admission control and routing algorithms for on-line multicast groups since we do not believe that a detailed description can be asserted with confidence without an extensive simulation study like that given for unicast communication in Chapter 6.

7.6.2 Source of Competitive Ratio

The NOMG and IOMG algorithms achieve a competitive ratio of $O(\log^6 n)$. There are several sources for the $\log n$ factors in the competitive ratio. We will discuss the source of each of the $\log n$ factors and argue why we do not expect to see the factors when comparing practical versions of the algorithms to an optimal off-line algorithm.

One $\log n$ factor in the competitive ratio is due to the fact that the requests arrive on-line. Hence the admission control and routing decisions must be made without knowledge of how the decisions will affect future requests. The $O(\log n)$ competitive AAP algorithm for unicast admission control and routing and the $O(\log n)$ competitive BMG algorithm for batched multicast admission control and routing have a $\log n$ factor in their competitive ratios for the same reason. We do not expect practical versions of the NOMG and IOMG algorithms to suffer from this $\log n$ factor. In particular, consider the EXP algorithm, our practical unicast admission control and routing algorithm developed based on the AAP algorithm. The simulation results for the EXP algorithm suggest that the EXP algorithm performs close to the optimal off-line algorithm in practice. We expect that practical versions of the NOMG and IOMG algorithms will include the same types of modifications that lead from the AAP algorithm to the EXP algorithm. As a consequence, we do not expect the $\log n$ factor due to the on-line arrival of the requests to appear in practice when practical versions of the NOMG and IOMG algorithms are compared to an optimal off-line algorithm.

A $\log^2 n$ factor in the competitive ratio is due to the fact that we can only construct a 1-maximal $O(\log^2 n)$-sparse tree, rather than a 1-maximal 1-sparse tree. (See the definitions of the INIT request code, Figure 7-2 and Figure 7-5, and the discussion of the MAXSPARSE algorithm in Section 7.2.3.) Thus, in the worst case, the MAXSPARSE algorithm, used by the NOMG and IOMG algorithms, will construct a tree where the cost of the tree is $O(\log^2 n)$ greater than the benefit of the tree. At present we have no practical experience with the MAXSPARSE algorithm. However, the structure of the algorithm suggests that the tree constructed by the

MAXSPARSE algorithm will, in practice, have much lower sparsity. Furthermore, the effect of a high sparsity tree is an admission control policy that tends to be more greedy. In practice we may be able to compensate for this effect by adjusting the constants in the cost function. (See the discussion in Chapter 5 of the constants used for the EXP algorithm.)

A $\log n$ factor in the competitive ratio is due to the fact that a given node can be an element of $\log n$ clusters. (See the definitions of the INIT request code, Figure 7-2 and Figure 7-5, for information on how clusters are used.) As a result, a node can be counted up to $\log n$ times when determining the benefit of the tree that spans the cluster centers and that is constructed in response to the INIT requests. In practice, we have never observed the CLUSTER algorithm to produce a set of clusters where a node is a member of $\log n$ clusters. In fact, our experience with the cluster algorithm suggests that, in practice, most nodes are elements of just one cluster, while a few nodes are elements of two clusters. Nodes that are elements of more than two clusters are extremely rare. In other words, in practice, the overlap between clusters is minimal. As a result, we do not expect to see this $\log n$ factor in practice when the NOMG and IOMG algorithms are compared to an optimal off-line algorithm.

Another $\log n$ factor in the competitive ratio results from the fact that the $\log n$ separation between clusters can only be guaranteed for a $\log n$ fraction of the clusters. Recall that the separation between clusters is needed to provide a lower bound on the cost of a tree spanning the nodes whose JOIN requests are rejected. However, since the overlap between clusters is minimal, the CLUSTER algorithm will typically achieve a $\log n$ separation for a constant fraction of the clusters. The effect of the $\log n$ factor in the competitive ratio is further mitigated in practice by the fact that the $\log n$ separation between clusters is only important if all the nodes whose JOIN requests are rejected are concentrated at the boundary of the clusters. If the nodes whose JOIN requests are rejected are not concentrated at the boundary of the clusters and the clusters have little overlap, a strong lower bound on the cost of a tree spanning the nodes whose JOIN requests are rejected can be established for nodes from all clusters, not just for nodes from a $\log n$ fraction of the clusters.

The final $\log n$ factor in the competitive ratio for the NOMG and IOMG algorithms arises from the fact that a cluster can have a diameter that is as much as $\log n$ larger than the desired diameter. Recall that the NOMG and IOMG algorithms connect nodes to the multicast tree using the cluster centers. By using the cluster centers we overestimate by a $\log n$ factor the lower bound on the cost of a tree spanning the nodes whose JOIN requests are rejected. Overestimating the lower bound on the cost of a tree spanning the nodes whose JOIN requests are rejected may cause certain JOIN requests to be unnecessarily rejected. Fortunately, experience with the CLUSTER algorithm suggests that the clusters will have a diameter that is at most a small constant factor larger than the desired diameter. Furthermore, the over estimation of the cost of a tree spanning the nodes whose JOIN requests are rejected only becomes significant when most of the nodes whose JOIN requests are rejected are concentrated at the boundary of the clusters. In practice, we do not expect this to be the case. Thus, we expect our estimate of the lower bound on the cost of a tree spanning the nodes whose JOIN requests are rejected to be fairly accurate.

7.6.3 Key Concepts from Competitive Algorithms

The polylogarithmic competitive ratio that we achieve for the NOMG and IOMG algorithms suggests that probabilistic information can be exploited to provide effective admission control and routing algorithms for on-line multicast groups. There are three key concepts that are used by the NOMG and IOMG algorithms to exploit the probabilistic information and that we believe will continue to play a key role in practical versions of the NOMG and IOMG algorithms. We discuss these concepts in this section.

Aggregation. The initial admission control and routing decisions for on-line multicast groups, made in response to INIT requests, are not made for individual nodes, rather the decisions are made for groups of nodes. The CLUSTER algorithm determines the groups of nodes for which the initial admission control and routing decisions are made. Making admission control and routing decisions for groups of nodes, rather than individual nodes, has two key advantages. First, the number of nodes that will actually issue a JOIN request can be predicted with much greater accuracy for a group of nodes than for an individual node. Our competitive analysis proof exploits this fact. In particular, we group nodes such that we can predict with high probability, i.e., with probability greater than $1 - O(1/n^3)$, the number of nodes that will actually issue a JOIN request in any given cluster. Making accurate predictions about the number of nodes that will issue a JOIN request will clearly be useful in practice as well. The second advantage of grouping nodes is the fact that the probabilistic information used by the algorithm will be much easier to determine for groups of nodes than for individual nodes. We discuss this issue further in Section 7.6.4.

Pre-reservation. Our algorithms pre-reserves resources for on-line multicast groups. In particular, both the NOMG and the IOMG algorithm pre-reserve bandwidth on a tree spanning the cluster centers of clusters from which the algorithms have decided they will admit JOIN requests. This pre-reservation is important. Let v be a node issuing a JOIN request such that the cost of the path to v can only be justified if other nodes near v also join the multicast group. The pre-reservation ensures that the nodes, whose successful JOIN requests node v needs in order to justify the cost of its path, will, when they issue their join requests, find enough network resources available to join the multicast group. In practical versions of the NOMG and IOMG algorithms we expect to use a much less aggressive form of pre-reservation. We discuss this issue further in Section 7.6.4.

Partial pre-reservation. The NOMG algorithm only pre-reserves resources to cluster centers. Thus, it only pre-reserves resources for groups of nodes. Since the request sequences for the NOMG algorithm are non-interleaved, the NOMG algorithm can assume that any node issuing a JOIN request will be able to join the multicast group using its local cluster center. For interleaved request sequences, this assumption no longer holds. Thus, to ensure that individual nodes will be able to connect to the multicast tree using their local center node, the IOMG algorithm must pre-reserve resources for *individual* nodes. In particular, it must pre-reserve resources along the paths from the individual nodes to the local center node. However, the probability that any individual node will join a particular multicast group can be very low. Thus, it is not feasible to pre-reserve all the resource that each node might need to connect to the multicast tree. The IOMG algorithm addresses this problem by for pre-reserving for each node an amount of resources that is proportional to the probability that the node will join the multicast group.

7.6.4 Key Changes for Practical Algorithms

In this section, we suggest how practical multicast algorithms might be constructed from the NOMG and IOMG algorithms. We discuss five areas where the NOMG and IOMG algorithms need to be modified.

Constants. As with the AAP algorithm, the constants used in the NOMG and IOMG algorithms are chosen for the worst case situation. Thus, we do not expect them to perform well in practice. However, the same techniques that were used to set the constants for the unicast algorithm can be used for the NOMG and IOMG algorithms. (See Chapter 5.) In fact, we believe that exactly the same constants used for the EXP algorithm can be used for practical versions of the NOMG and IOMG algorithms. (A review of the discussion in Chapter 5 shows that the discussion applies without modification to multicast communication.)

Pre-reservation. In Section 7.6.3 we mention that the pre-reservation done by the NOMG and IOMG algorithms can be less aggressive in practice. Recall that the NOMG and IOMG algorithms reserve resources along the tree spanning the center nodes of the clusters. This pre-reservation can be done for subsections of the tree instead of the entire tree. Consider for example a cluster where the expected number of multicast members is high enough to support a connection to the source without taking into account the expected multicast members from nearby clusters. Furthermore, assume that the expected multicast members of the cluster are not needed to support connections to nearby clusters. In this case, no pre-reservation is needed for the cluster. Rather, resources along a path to the cluster need only be reserved when the first JOIN request arrives from a node in the cluster.

Reoptimization. The NOMG and IOMG algorithms use minimum spanning trees rather than the shortest path trees that have typically been used by the Internet community [WDP88, DC90, Dee91, Moy92]. The advantage of minimum spanning trees is the fact that they are typically much more efficient. It is easy to construct situations where a shortest path tree requires many more resources to span the members of a multicast group than are required by a minimum spanning tree. Moreover, a shortest path tree will never require fewer resources than a minimum spanning tree. In fact, the use of a shortest path tree in our multicast algorithms would lead to a linear (instead of the current polylogarithmic) competitive ratio.

However, the disadvantage of minimum spanning trees is that the optimality of any particular tree can be seriously eroded by a dynamic membership. In contrast, shortest path trees remain optimal in the face of dynamic membership. Thus, unlike shortest path trees, our minimum spanning trees will require some periodic reoptimizations when the membership in the multicast groups has changed substantially.

Gathering probabilistic information. The NOMG and IOMG algorithms rely on probabilistic information that may not be available in practice. Furthermore, if the information is available, it could be inaccurate. These inaccuracies could stem both from honest mistakes on the part of users and from misinformation, the goal of which is to affect the admission control decisions of the network.

We suggest several ways of dealing with these issues. First, the lack of information can be dealt with by using default probability information and then learning the correct probability information over time. This technique is likely to be successful for multicast groups whose behavior reaches a steady state. However, many multicast groups may not have steady state behavior. Consider, for example, a multicast group that distributes CSPAN. For some debates there may be considerable interested. However, for other debates, CSPAN may be lucky to get a single viewer. There are several possible approaches to such multicast groups. One approach is to use probabilistic information that is content dependent. In this case, CSPAN might warn the network whenever it expects a particular debate to be popular. Another possibility is to have two sets of probabilistic assumptions: a high use set and a low use set. Whenever many JOIN requests are detected over a short period of time, the network starts using the high use set of assumptions. After a prolonged period with few arrivals the network switches to the low use set. The advantage of this approach is that it does not require constant user intervention. The problems with getting accurate information can be countered by aggregating members for which probability information is collected. For example, the probabilistic information could be collected on a per local access network basis rather than a per user basis. (Recall from our discussion in Chapter 4 that local access networks should use greedy admission control. Thus, our multicast admission control and routing algorithms will only be concerned with connecting access networks to multicast groups rather than connecting individual users to multicast groups.) Finally, we consider the issue of misinformation. There are two complementary approaches. First, the network can learn about the true probabilistic information much in the same way it has to learn about the information for situations where no probabilistic information is available. Second, the network can impose financial or service penalties on users that provide probabilistic information that turns out to be significantly incorrect.

Dynamic membership. Dynamic multicast membership poses problems in addition to those associated with the optimality of the minimum spanning tree. The following example illustrates one of the additional problems associated with dynamic multicast membership. Consider two nodes that are close to each other, but that are so far from the source that the cost of connecting them to the multicast tree of the source can only be justified if both of them join the tree. If the probability of each of them joining the multicast group is high, then the NOMG and IOMG algorithms will accept the first JOIN request that comes from one of them in anticipation that the other node will also join eventually. Unfortunately, if the membership is dynamic, the first of the two nodes may have left the multicast group by the time the second node issues a JOIN request. In this case, the expensive connection from the source is, at any given time, only supporting one node. However, the assumption for our example was that the connection could not be justified unless it is supporting two nodes. This argument can be formalized to show that the competitive ratio would become linear in the size of the network if the multicast group membership were dynamic. To address this problem, we propose collecting slightly different probabilistic information. In particular, instead of knowing the probability that a give node will ever join a multicast group, the algorithm should know the probability that it will join the multicast group over some small time interval. This way, the algorithm can determine the likelihood that two nodes, which are close to each other, but which are so far from the source that the cost of connecting them to the multicast tree can only be justified if both of them join the tree, will actually want to join the multicast group in overlapping time intervals.

CHAPTER 8

Discussion and Future Work

This chapter consists of three sections. Some of the algorithms discussed in this thesis are not practical in their present form. However, the algorithms illustrate several principles important to admission control and routing algorithms. These principles are summarized in Section 8.1. The admission control and routing problem can be seen as an instance of a more general resource allocation problem. Section 8.2 discusses the applicability of our results to the more general resource allocation problem. The section also mentions that we expect resource allocation problems that require admission control to become increasingly important in the near future. Finally, Section 8.3 mentions several open problems.

8.1 Lessons From Theory

Non-greedy admission control. The theoretical analysis presented in this thesis provides strong support for the use of non-greedy admission control strategies. In particular, the lower bounds on the oblivious competitive ratio provided for lines, trees, meshes, trees of meshes, fat-trees and hypercubes are all strictly lower than the lower bounds on the competitive ratio for greedy admission control and routing algorithms for the same topologies. Furthermore, for the tree and the fat-tree we provide randomized algorithms that match the oblivious lower bounds and thus beat the greedy lower bounds. For general topology networks we prove an $\Omega(n)$ lower bound on the competitive ratio for greedy admission control algorithms. Furthermore, the AAP algorithm of [AAP93] and the multicast algorithms of Chapter 7 beat this $\Omega(n)$ lower bound by achieving polylogarithmic competitive ratios. The lower bounds in [AAP93] show these algorithms to be close to optimal. The simulation results for the EXP algorithm in Chapter 6 confirm the theoretical support for non-greedy admission control.

The following example, constructed for a hypothetical telephone network, illustrates the intuition supporting the use of non-greedy admission control. Assume a user is attempting to place a call from Boston to New York. However, the links on the direct path between Boston and New York are currently 100% utilized. Assume further that there is a path through San Francisco that has enough capacity to carry the call from Boston to New York. If the links along the path through San Francisco have low utilization, the call from Boston to New York should clearly use the path. However, if the links along the path through San Francisco have high utilization, the call from Boston to New York should not use the path, because this might block two future calls, e.g., one from Boston to San Francisco and one from San Francisco to New York. Essentially, the non-greedy admission control should reject the call from Boston to New York in the hope/assumption that it will be able to carry two calls along the same path. This hope/assumption is based on observing the high utilization on the links of the path through San Francisco. This high utilization signals both that the links are likely to run out of capacity soon and that the links carry many current (and future) calls, some of which may use the links more efficiently than a call from Boston to New York that is routed through San Francisco.

Randomization. For trees and meshes the lower bounds for deterministic algorithms are higher that the lower bounds we provide for the oblivious competitive ratio of randomized algorithms. Furthermore, for trees we provide a randomized algorithm, TREE, that beats the deterministic lower bound and matches the oblivious lower bound. (See Chapter 4.) The randomization in that algorithm is used for several purposes, including for the purpose of preventing pathological request sequences from causing a poor competitive ratio. (If the roadblocks were not placed randomly, a request sequence that always requests circuits whose paths go through the roadblocks would cause a poor competitive ratio.) Thus, the TREE algorithm suggests that randomization can be an important tool for avoiding poor performance from pathological request sequences.

Hierarchical backbone networks. Many modern networks, like the Internet and the telephone system, are structured into many small low capacity regions (access networks) and a large high capacity region (backbone network) that connects the access networks to each other. Thus, these networks can be seen as hierarchical backbone networks (cf. Chapter 4). The theoretical results in this thesis suggest that greedy admission control should be used for the access networks while non-greedy admission control should be used for the backbone network. Our mechanism for constructing admission control algorithms for hierarchical backbone networks suggests that we should use greedy admission control in the access networks. Greedy admission control has a good competitive ratio in the access networks since each of these networks only has short paths. Furthermore, our fat-tree algorithm (cf. Chapter 4) suggests that we should use non-greedy admission control in the backbone network. The simulation results of Chapter 6 further support this approach. In particular, they show that non-greedy admission control is most effective for the high bandwidth links typically used in backbone networks, while it is not very effective for the low bandwidth links often used in access networks. Recall, from the discussion in Section 6.3 that the increased performance advantage of the non-greedy algorithms on high capacity networks is based on the law of large numbers. In particular, high capacity backbone networks typically serve more circuits. Thus, the expectations arising from the statistical assumptions are more likely to be accurate predictions.

Multicast. There are several important implications for practical multicast algorithms that arise out of the competitive multicast algorithms presented in Chapter 7. We consider four implications. First, the probabilistic information about which nodes may join a multicast group can be successfully exploited. In fact, using the clustering techniques, the information can even be exploited when the likelihood that any particular node will join a particular multicast group is low. The need to use the clustering to aggregate the probability information of multiple nodes represents the second important implication of Chapter 7. The third observation is that pre-reservation of resources is important. In particular, our algorithms reserve resources along the tree connecting the center nodes of the clusters. This pre-reservation is needed for closely situated clusters of the network that individually cannot justify the use of some expensive path to connect to the multicast group, but together can justify the use of the path. For such multicast groups the pre-reservation guarantees that each of the clusters will in fact be able to join the multicast group once JOIN requests from nodes in that cluster arrive. (In practice, this pre-reservation can likely be done less aggressively. See Section 7.6.4.) Finally, our multicast algorithm for interleaved on-line multicast groups suggests that resource reservation that is proportional to the probability that a node will join a multicast group is needed.

8.2 Future Applications of Admission Control

The admission control and routing problem is an instance of a more general resource allocation problem. Specifically, consider a system which manages some kind of resource. In the case of circuit networks, this resource is bandwidth. However, one can imagine systems where the key resource might be cpu time, disk space, access time to some shared data structure, etc. Users send requests to this system. In response to these requests, the system first decides whether or not it will accept the requests and then decides which specific resources it wants to use to service the requests. There are four key characteristics that make such resource allocation systems candidates for the techniques presented in this thesis. First, they must have unrelated users. In particular, if the system rejects one user, this should not significantly impact the desire of other users to join the system. Our admission control strategies are predicated on the idea that users requiring too many resources be rejected in favor of those that require fewer resources. If rejecting the requests that require many resources causes those that require few resources not to arrive, our strategy would clearly not be successful. The second characteristic is that many users are needed. Having many users ensures that the predictions made by the algorithm about future events on the basis of statistical assumptions are more likely to be accurate. (See the discussion in Sections 8.1 and 6.3.) The third characteristic is that multiple resources must be needed to satisfy a request. If not, the entire notion of comparing requests that require many resources to requests that require few resources makes no sense. The fourth characteristic is that resources must actually be reserved for users.

Today there are not many systems that meet the four above described characteristics. (The long distance telephone networks are examples of such systems. In fact, these networks use some of the ideas that came from the symmetric loss network literature [ACF$^+$92].) However, we expect that there will be many systems like this in the near future. Some immediate examples are the public Lotus Notes system that AT&T is developing and the Internet if it moves towards circuit routing and a usage based fee system. Furthermore, the proliferation of home computers, the growth in on-line services, and the substantial infrastructure investments that phone and cable companies are making, suggest that large scale systems with the above described characteristics will become increasingly important.

For many systems we actually expect the gains that can be achieved by effective admission control to be much greater than the gains we are able to achieve in unicast communication. (See Chapter 6.) For unicast communication the effect of a poor admission control decision is limited by the length (in the number of links) of the longest allowable path. (This is the maximum number of short requests that may have to be rejected as a result of admitting the long request.) However, for multicast communication, for example, the effect of a poor admission control decision is only limited by the size of the largest multicast group. For example, consider a region of the network to which capacity constraint imply that the nodes in the region can only connect to one more multicast group outside the region. Now a poor admission control decision could admit a multicast group with one member in that region instead of a multicast group with thousands of members in that region.

8.3 Open Problems

General topology networks. The AAP algorithm of [AAP93] provides the optimal competitive ratio for admission control and routing algorithms when every virtual circuit requests at most $1/\log n$ fraction of the capacity of the lowest capacity link. Our general topology multicast algorithms of Chapter 7 make similar restrictions. Our algorithms for trees and fat-trees overcome these restrictions. In particular, they can accommodate circuits that require the entire capacity of the links. An open question is whether or not algorithms with good (polylogarithmic) competitive ratio that do not restrict circuits to a small fraction of the link capacity exist for general topology networks.

Exploiting special topologies. The competitive ratios achieved by the AAP algorithm of [AAP93] and our multicast algorithms of Chapter 7 for general topology networks are polylogarithmic in the size of the network. However, for certain special topologies, better competitive ratios are possible. For example, a constant competitive ratio can be achieved on symmetric loss networks. An open question is whether or not certain network characteristics, such as for example connectivity, can lead to competitive ratios that are better than polylogarithmic.

Multicast with dynamic membership. In Section 7.6.4 we discuss some of the complications associated with dynamic membership. In that section we propose using information about the probability that a node will join a multicast group over some small time interval. An open question is how to incorporate this type of information into the analysis of Chapter 7.

Practical multicast algorithms. In Section 7.6.4 we describe some possible approaches to designing a practical multicast admission control and routing algorithm. We intend to explore those approaches. At present though, the question of how to design a practical multicast admission control and routing algorithm remains open.

Bibliography

[AABV95] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guaratees for minimum-weight k-trees and prize-collecting salesmen. In *Proceedings of the* 27^{th} *Annual ACM Symposium on Theory of Computing, Las Vegas, Nevada*, 1995.

[AAF$^+$93] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proceedings of the* 23^{rd} *Annual ACM Symposium on Theory of Computing, San Diego, California*, May 1993.

[AAG94] B. Awerbuch, Y. Azar, and R. Gawlick. Dense trees and competitive selective multicast. unpublished manuscript, December 1994.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In 34^{th} *Annual Symposium on Foundations of Computer Science,* Palo Alto, California, November 1993.

[ABFR94] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive non-preemptive call control. In *Procedings of 5'th ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[ACF+92] G.R Ash, J. Chen, A.E. Frey, B.D. Huang, C. Lee, and G.L. McDonald. Real-time network routing in the AT&T network-improved service quality at lower cost. In *Proceedings of the IEEE Global Telecommunications Conference, Orlando, Florida*, pages 115–136, December 1992.

[ACG91] H. Ahmadi, J.S Chen, and R. Guérin. Dynamic routing and call control in high-speed integrated networks. In *Procedings Workshop Systems Engngineering and Traffic Engineering, ITC'13, Copenhagen, Denmark*, 1991.

[AG90] H. Ahmadi and R. Guérin. Bandwidth allocation in high speed networks based on concept of equivalent capacity. In *Procedings 7th ITC Seminar, Morristown, NJ*, October 1990.

[AGLR94] B. Awerbuch, R. Gawlick, F.T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high performance computation and communication. In *Proceedings of* 35*th Conference on Foundations of Computer Science, Santa Fe, NM*, November 1994.

[Aki84] J. Akinpelu. The overload performance of engineered networks with nonhierarchical and hierarchical routing. *AT&T Bell Laboratories Technical Journal*, 63(7):1261–1281, September 1984.

[AKP+93] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. On-line load balancing of temporary tasks. In *Procedings of Workshop on Algorithms and Data Structures*, pages 119–130, August 1993.

[AP90] B. Awerbuch and D. Peleg. Sparse partitions. In 31^{st} *Annual Symposium on Foundations of Computer Science*, pages 503–513, 1990.

[BDBK+90] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. In *Proceedings of the* 22^{nd} *Annual ACM Symposium on Theory of Computing*, pages 379–386, May 1990.

[BFKR93] A. Blum, A. Fiat, H. Karloff, and Y. Rabani. Private communication, 1993.

[BG92] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, NJ, second edition, 1992.

[BLS87] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. In *Proceedings of the* 19^{th} *Annual ACM Symposium on Theory of Computing,* New York City, pages 373–382, may 1987.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. *Journal of the ACM*, 39(4):745–763, 1992.

[Bou92] J.Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[BTC92] Anthony Ballardie, Paul Tsuchiya, and Jon Crowcroft. Core based trees (CBT). Internet Draft I-D, University College London, November 1992.

[Buc94] R. Buck. The oracle media server for ncube massively parallel systems. In *Proceding of the 8th International Parallel Processing Symposium*, pages 670–673, April 1994.

[Cas94] S. Casner. Frequently asked questions on the multicast backbone (MBONE). From `ftp.isi.edu:mbone/faq.txt`, 1994.

[CCS94] I. Castineyra, J. N. Chiappa, and M. Steenstrup. The nimrod routing architecture. Technical report, Internet Draft, July 1994.

[CG88] I. Cidon and I. S. Gopal. PARIS: An approach to integrated high-speed private networks. *International Journal of Digital & Analog Cabled Systems*, 1(2):77–86, April-June 1988.

[CGG91] I. Cidon, I. Gopal, and R. Guérin. Bandwidth management and congestion control in plaNET. *IEEE Communications*, 29(10):54–63, October 1991.

[DC90] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANS. *ACM Transactions on Computer Systems*, pages 85–111, May 1990.

[Dee91] S. Deering. Multicast routing in a datagram internetwork. PhD thesis, Stanford University, 1991.

[DEF+94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. Protocol independent multicast (pim). Internet draft, Stanford University, March 1994.

[deP91] M. dePrycker. *Asynchronous Transfer Mode, Solutions for Broadband ISDN*. Ellis Horwood, England, 1991.

[EM93] A. Elwalid and D. Mitra. Effective bandwidth of general Markovian traffic sources and admission control of high speed networks. *IEEE Transactions on Networking*, 1:329–343, 1993.

[EM95] A. Elwalid and D. Mitra. Analysis, approximations and admission control of a multi-service multiplexing system with priorities. In *Proceedings of IEEE INFOCOM*, pages 463–472, 1995.

[GAN90] Guérin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high speed networks. Technical Report RC 16317, IBM Watson Research Division, October 1990.

[GG92] J.A. Garay and I.S. Gopal. Call preemption in communication networks. In *Proceedings of INFOCOM '92*, volume 44, pages 1043–1050, Florence, Italy, 1992.

[GIK+92] J. Garay, I.Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. In *Proceedings. of 2nd Annual Israel Conference on Theory of Computing and Systems*, Israel, 1992.

[GK90] R.J. Gibbon and F.P. Kelly. Dynamic routing in fully connected networks. *IMA journal of Mathematical Control and Information*, 7:77–111, 1990.

[GKPR95a] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Distributed routing and admission control. In preparation, 1995.

[GKPR95b] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and admission control of virtual circuits in general topology networks. In preparation, 1995.

[GKPR95c] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and admission control using known distributions on holding times. In preparation, 1995.

[GKR95] R. Gawlick, C. Kalmanek, and K. Ramakrishnan. On-line routing for permanent virtual circuits. In *Proceedings of IEEE Infocom*, April 1995.

[GS95] A. Greeberg and R. Srikant. Computational techniques for accurate performance evaluation of multirate, multihop communication networks. Unpublished Manuscript, 1995.

[HSE95] S. Herzog, S. Shenker, and D. Estrin. Sharing the "cost" of multicast trees: An axiomatic analysis. In *Proceedings of SIGCOMM '92*, Boston, Massachusetts, August 1995.

[Kat67] S. Katz. Statistical performance analysis of a switched communication network. In *ITC 5*, pages 566–575, 1967.

[Kel88] F.P. Kelly. Routing in circuit-switched networks: optimization, shadow prices and decentralization.formal verification of parallel programs. *Advances in Applied Probability*, 20:112–144, 1988.

[KMB81] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[KMRS88] A. Karlin, M. Manasse, L. Rudolpoh, and D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[Kru82] R. Krupp. Stabilization of alternate routing networks. In *Proceedings of IEEE Internation Conference on Communications*, page 3I.2.1, 1982.

[LAD+92] C. Leiserson, Z. Abuhamdeh, D. Douglas, M. Ganmukhi C. Feynman, J. Hill, W. Hillis, B. Kuszmaul, M. St. Pierre, D. Wells, M. Wong, S. Yang, and R. Zak. The network architecture of the connection machine cm-5. In *the proceedings of SPAA92*, 1992.

[Lei93] C. Leisersen. Private communication, 1993.

[Mar83] V. Marbukh. Study of a fully connected channel communication network with many nodes and circumventing routes. *Automatika i Telemakhanika*, 12:86–94, 1983.

[MG92] D. Mitra and R. Gibbons. State dependent routing on symmetric loss networks with trunk reservations ii: Asymptotics, optimal design. *Annals of Operations Research*, 35(3):3–30, 1992.

[MGH93] D. Mitra, R. Gibbens, and B. Huang. State-dependent routing on symmetric loss netowkrs with trunk reservations – I. *IEEE Transactions on Communications*, 42(2), February 1993.

[Moy92] J. Moy. Multicast extension to OSPF. Technical report, Internet Draft, September 1992.

[MS91] D. Mitra and J. Seery. Comparative evaluations of randomized and dynamic routing strategies for circuit switched networks. *Annals of Operations Research*, 39(1):102–116, 1991.

[NT94] C. Noronha Jr. and F. Tobagi. Evaluation of multicast routing algorithms for multimedia streams. In *Proceedings of IEEE ITS, Rio de Janeiro, Brazil*, August 1994.

[OK85] T. Ott and K. Krishnan. State dependent routing of telephone traffic and the use of separable routing schemes. In *Proceedings of the 11th international teletraffic congress, Kyoto, Japan*, 1985.

[Plo95] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE Journal on Selected Areas in Communications*, 1995. Invited paper.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[Rab76] M.O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Results*, pages 21–39. Academic Press, 1976.

[Rag86] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. In 27^{th} *Annual Symposium on Foundations of Computer Science,* Toronto, Ontario, Canada, pages 10–18, May 1986.

[Ric92] F. Hwang D. Richards. Steiner tree problems. *Networks*, 22:55–89, 1992.

[RS89] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *Proceedings 16th ICALP*, July 1989. To appear in Springer-Verlag Lecture Notes in Computer Science 372.

[RSC86] V. Rayward-Smith and A. Clare. On finding steiner vertices. *Networks*, 16:283–294, 1986.

[SD94] S. Sibal and A. DeSimone. Controlling alternate routing in general-mesh packet flow networks. *Proceedings of SIGCOMM 94*, 1994.

[Sha94] N. Shacham. Real-time traffic distribution in heterogeneous environment–phase 1 technical report. Technical report, SRI International, April 1994.

[SM90] F. Shahrokhi and D. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37:318 – 334, 1990.

[ST85] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[VG93] D. C. Verma and P. M. Gopal. Routing reserved bandwidth multi-point connections. In *Proceedings of the Annual ACM SIGCOMM Symposium*, pages 96–105, San Fransisco, CA, September 1993.

[Wax88] B. Waxman. Routing of multipoint connections. *IEEE Journal on Slected Areas in Communications*, 6(9):1617–1622, December 1988.

[WDP88] D. Waitzman, S. Deering, and C. Partridge. Distance vector multicast routing protocol. Technical report, RFC1075, November 1988.

[Yao77] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of 18th IEEE Symposium on Foundations of Computer Science*, pages 222–227, October 1977.

[ZDE+93] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: a new resource reservation protocol. In *Proceedings of the International Networking Conference (INET)*, pages BCB–1, San Francisco, California, August 1993. Internet Society.