Maximizing Throughput of Reliable Bulk Network Transmissions

by

John W. Byers

B.A. (Cornell University) 1991

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Christos Papadimitriou, Chair
Professor Michael Luby
Professor Richard M. Karp
Professor Dorit Hochbaum

Fall 1997

The dissertation of John W. Byers is approved:

Chair Date

Date

Date

Date

University of California at Berkeley

Fall 1997

Maximizing Throughput of Reliable Bulk Network Transmissions

by
John W. Byers

Abstract

Maximizing Throughput of Reliable Bulk Network Transmissions

by

John W. Byers
Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Christos Papadimitriou, Chair

We study combinatorial optimization and on-line scheduling problems which arise in the context of supporting applications which transmit bulk data over high-speed networks. One of our primary objectives in this thesis work is to formulate appropriate theoretical models in which to develop and analyze efficient algorithms for these problems – models which reflect both the experience of network architects, the design of network protocols, and contributions of theoretical research.

We first consider the optimization problem of maximizing the utilization of a shared resource, network bandwidth, across a set of point-to-point connections. A feasible solution to this allocation problem is an assignment of transmission rates to the connections which does not violate the capacity constraints of the network links. The connections and routers which are responsible for establishing this allocation must do so with incomplete information and limited communication capabilities. We develop a theoretical model which addresses these considerations and study the tradeoff between the quality of the solution we can obtain and the distributed running time. Our main theoretical result is a distributed algorithm for this problem which generates a feasible $(1+\epsilon)$-approximation to the optimal allocation in a polylogarithmic number of distributed rounds. A sequential implementation of our distributed algorithm gives a simple, efficient approximation algorithm for general positive linear programming. Subsequent experience with an implementation of the algorithm indicates that it is well suited to future deployment in high-speed networks.

The next problem we consider is the following on-line scheduling problem, which the sender of a point-to-point bulk transmission must address: Given an on-line sequence of transmission times, determine which data item to transmit at each transmission time, so as to maximize effective throughput to the receiver at all points in time. For this application, we measure effective throughput as the length of the intact prefix of the message at the receiver. This problem is made difficult in practice by factors beyond the sender's control, such as packet loss and wide variance in packet round-trip times. Using the method of competitive analysis, we compare the performance of our algorithm to that of an omniscient algorithm. We prove that while all deterministic policies perform poorly in this model, a simple randomized policy delivers near-optimal performance at any given point in time with high probability. Moreover, our theoretical result ensures that typical performance does not degrade significantly – a claim which our empirical studies bear out.

Using the models and tools developed for these problems, we then consider analogous problems which arise for *multicast* bulk transmissions, transmissions targeted to multiple destinations. We show how to tune our bandwidth allocation policy to still deliver a $(1+\epsilon)$ approximation to the optimal allocation in a polylogarithmic number of distributed rounds. For the scheduling problem, we prove that no on-line scheduling policy can deliver high performance which scales with the number of receivers without using encoding. We then show that by using forward error correction coding techniques, a simple multicast policy delivers effective throughput within a constant factor of optimal independent of the number of receivers.

Professor Christos Papadimitriou
Dissertation Committee Chair

To Lisa, who was there every step of the way.

Contents

List of Figures

Acknowledgements

This work was inspired and shaped by a great many individuals, to whom I am most grateful.

I had the fortuitous opportunity to work under the guidance of Dick Karp, Mike Luby and Christos Papadimitriou over the course of my graduate career at Berkeley. These three faculty members have been instrumental in teaching and mentoring me, and in setting the direction of my research.

Dick's insight into the heart of the matter both in teaching and in research were an inspiration early in my career. Preparing for research meetings with Dick was the best possible form of training for a beginning graduate student in theoretical computer science and I wish to specifically thank him for the time he put in to advising his students. Mike has had the greatest influence in directing and shaping the ideas that eventually formed this thesis. His guidance, experience, constructive criticism and friendship over the last several years have formed the basis for a strong advisor-student relationship without which this thesis would not have been written. Likewise, discussions with Christos were of great value and contributed substantially to the direction of several ideas presented in this thesis.

A number of others had substantial impact on this work. I would like to thank the three aforementioned individuals, along with Dorit Hochbaum, for serving on my dissertation committee and for making substantial improvements to the contents.

Lisa Camesano also devoted a substantial amount of time reading, reorganizing and crystallizing earlier versions of this thesis. Her contributions in this and other regards are immensely appreciated.

This work could not have come to fruition without the benefit of time well spent with Micah Adler. Our collaboration earlier in our graduate student careers was of immeasurable benefit to my development as a theoretician.

I am most fortunate that Yair Bartal and Danny Raz each spent a two year visit at the International Computer Science Institute which dovetailed with my time there. Yair's intensity and Danny's focus brought out some of my most productive research activity and our collaborations together form a key component of this thesis work.

My Berkeley graduate student experience would have been greatly diminished without the benefit of time spent with numerous members of the Berkeley and ICSI theory groups, especially Mike Mitzenmacher, David Blackston, Eric Vigoda, Deborah Weisser, Satish Rao, Alistair Sinclair, Umesh Vazirani, Ethan Bernstein, and Mor Harchol-Balter.

A host of others made every effort to assist me in making the time spent as a graduate student as enjoyable as possible. You all know who you are, but special recognition is accorded to Micah Adler, David Blackston, Mike Dahlin, Ketan Mayer-Patel, Ashu Rege, Eric Vigoda, and Alexis members.

Two other individuals deserving special mention are David Gries and Mark Rose, whose inspirational teaching techniques in part impacted my decision to start and finish my PhD, respectively.

Chapter 1

Introduction

Integrated-service networks currently under development will support applications with a diverse set of performance constraints and traffic characteristics. One emerging theme in the design of integrated-service networks is the need for multiple classes of service to support the diverse requirements of these applications effectively. At present, there are at least three distinct classes of applications which are likely to compete for shared resources such as link bandwidth in these networks. The first class, *interactive applications*, are characterized by transmission of short point-to-point messages and typically involve multiple traffic sources. The primary performance requirement of interactive applications, exemplified by Internet telephony, is to minimize the end-to-end latency of these brief transmissions. Second, *streaming applications* such as real-time video broadcast, are characterized by continuous injection of data into the network by a single source. The requirements of such applications can be complex; it is often necessary to deliver the data to the receiver or receivers at a smooth rate while keeping the end-to-end latency of transmissions low. Finally, *bulk applications*, such as point-to-point file transfer, involve reliable dissemination of data from a single source in the network to one or more receivers. These applications, which currently comprise a large fraction of Internet traffic, have a relatively simple performance requirement to satisfy: deliver the data to the receivers as quickly as possible.

In this thesis, we focus on developing efficient algorithms for optimization and scheduling problems which arise in the context of providing support for bulk applications in integrated-service networks. Some of the algorithmic techniques we present also provide support for streaming applications. The first problem we study in this context regards the allocation of bandwidth to bulk applications in the network. Since the goal of these applications is to deliver data to the receivers as quickly as possible, if left unchecked, they can inject data into the network at rates which could overrun the capacity of the underlying links. We assume that sources inject data into the network using *packets* with a bounded maximum payload size. When packets arrive at a network router faster than they can be processed, queue buildup results, followed eventually by packet loss when the queues overflow.

One emerging technique used to address this problem, *rate-based flow control*, regulates the rates at which point-to-point applications may inject data along the fixed paths they use in the network. If packet sizes are fixed, these rates can be allocated in packets per second, but they are more commonly specified in bits per second. A feasible end-to-end allocation of rates to a static set of connections is one in which each connection may subsequently transmit data at their allocated rate without violating the capacity constraints of the network. Designing rate-based flow control algorithms is made difficult by the *distributed* nature of the computational resources in the network and by the *dynamic* arrival and departure of connections over time. We place substantial emphasis on developing a distributed model which captures the limited communication and computational capabilities of these network resources accurately. While one widely studied objective of this approach by researchers in the networking community is distributing the bandwidth *fairly* across the connections, we will motivate the study of a different objective function for this problem: maximizing network throughput.

The second problem we address regards a scheduling problem that the sender of any network transmission must solve: selecting what data to place in each packet of the transmission to make the most effective use of the available bandwidth. Complicating the issue is the fact that packets in the network may be lost, they may experience widely varying end-to-end network latency, and the rate at which a source can inject packets into the network may fluctuate dramatically over time. Moreover, these complications are brought on by factors beyond the sender's control. Specific applications may have to deal with additional considerations. For example, streaming applications must decide whether it is worthwhile to retransmit a lost packet which may not arrive in time to be of value to the receiver. Likewise, multicast applications must decide whether to retransmit a packet (to all multicast recipients) which has been lost by a subset of the multicast group. We formulate these problems more precisely after outlining our model of the network.

1.1 Networking Terminology

Throughout this thesis, we model the network as a capacitated, directed graph with nodes representing connection endpoints and network routers, edges representing links, and capacities representing link bandwidth. Since the router at the tail of an edge makes admission control decisions for that edge, we will frequently think of the routers, rather than the edges, as being capacitated. Often, a single physical router may reside at the tail of a constant number of network links; we model this phenomenon by using a separate virtual router for each outgoing edge. Each router may maintain a constant amount of state about each of the connections which utilize it. A source communicates by injecting *packets* into the network, each of which contains header information and a payload of bounded size. In many protocols, a packet arrival at a receiver triggers an *acknowledgment*, in the form of a packet initiated by the receiver and targeted to the sender. When acknowledgments are employed, we use the standard term *round-trip time* of a packet to denote the elapsed time between the sender's transmission of a packet and the sender's receipt of an acknowledgment for that packet. Throughout this thesis, we model the network as *connection-oriented*, in that for every application, all packets traveling from a source to a given destination use the same fixed route in the network. In this spirit, we often refer to the communication channel between a source and destination as a *connection* in the network, even though the connection is a virtual one. Our model is intended to reflect the reality of a trend in high-speed networks towards connection-oriented architectures with per-connection state inside the network [MSZ 96].

We separately consider unicast and multicast applications, applications which transmit data to a single destination and multiple destinations, respectively. In a connection-oriented network, a unicast application establishes a connection along a source-destination *path* in the network. Therefore, packets injected by a unicast source traverse the routers and edges along the fixed path to the destination. A multicast application communicates along a set of edges and routers that form a multicast *tree* in the network rooted at the sender. The leaves of the multicast tree correspond to the destinations of the transmission. In standard multicast implementations, such as IP Multicast, when a packet reaches a router corresponding to an internal node in this multicast tree, that router forwards a copy of the packet on *all* of its outgoing links which are edges of the multicast tree. We employ this model in our work as well, and describe it completely in Chapter 5.

1.2 Bandwidth Allocation

The most widely studied objective of rate-based flow control in the networking community is *max-min fairness* [Jaf 81]. A max-min fair equilibrium state has the following characterization: no connection can increase their rate without decreasing the rate of some other connection with equal or lower rate [BG 87]. Algorithmic techniques for efficiently converging on a max-min fair allocation for a set of bulk connections is described in the section on related work.

In many settings, fairness is not the most important consideration, and in fact, much of the theoretical work on rate-based allocation focuses on maximizing network *throughput*. There is extensive motivation for studying this objective function. In a pay-per-use network, network administrators are interested in generating allocations which maximize the throughput of the network, or maximize the revenue derived from the set of connections. In intranets where bulk connections can be prioritized, higher priority connections warrant receiving higher rate allocations. The objective in these scenarios can be expressed as maximizing the weighted throughput of the connections. A drawback of throughput maximization is that some low-profit connections may receive an allocated rate of zero. However, it is possible to combine approaches, first by providing a minimal rate, or fairness, guarantee to all connections, then maximizing the throughput on the residual network capacity.

We begin with a simplified formulation of the static bandwidth allocation problem where our objective is to maximize network throughput, measured as the sum of the allocated rates across n point-to-point connections. This problem is most easily expressed as the following simple linear program.

$$\begin{aligned} \max \sum_{j=1}^{n} y_j \text{ s.t.} \\ \forall i, \sum_{j} a_{ij} y_j \leq C_i \\ \forall j,\ y_j \geq 0 \end{aligned}$$

In the program we solve for the variables y_j, which correspond to the flow rates assigned to the connections. The 0/1 constants a_{ij} are set to 1 if the route of connection j utilizes router i and 0 otherwise. The constants C_i hold the value of the capacity of router

i. A feasible solution to this program corresponds to an end-to-end allocation to the set of connections.

There are numerous obstacles to developing an efficient algorithmic solution to this optimization problem which can be successfully deployed in a high-speed network. One fundamental obstacle is the *distributed* nature of the computational resources in our network. Although a centralized coordinator with complete information about the system, and thus the linear program, could compute an optimal allocation in polynomial time, no such coordination exists in a distributed network. In the system we study, connections are autonomous, meaning that they have no prior knowledge of other connections in the system, nor does the network have any prior knowledge of which connections will be present. We model this behavior by viewing connections and routers as distributed agents. Although these agents initially have very limited information, they are able to communicate in a natural, but restricted way. In particular, connections may communicate with routers through which they pass, and routers with connections which pass through them. Then with the incomplete information they obtain about the system, they must then decide upon a feasible allocation. We formalize this model, which derives from work of Papadimitriou and Yannakakis [PY 93] in Chapter 3.

In the setting of networking, the distributed running time is a crucial consideration, as algorithms with which run in polynomial time are often too inefficient to warrant practical implementation. It can therefore be important to sacrifice optimality for the sake of more efficient running time. We consider the trade-off between the distributed running time and the quality of the solution we obtain, measured in terms of the *approximation ratio*, the ratio between the value of the optimal solution and the value of the solution our algorithm obtains. Also, we strive to keep our algorithms as simple as possible, since they will be implemented at the connection endpoints and routers of the network, where processing time is a critical resource.

Another consideration is that of *dynamic* arrivals and departures of connections in the system. A solution to the static allocation problem is useful only while the set of connections involved in the allocation remain in the system. We consider how to smoothly and quickly adjust the allocation as connections arrive and depart. We address the following questions about this approach in this thesis:

Question 1 *What communication and computational model for the network is appropriate* *for designing distributed bandwidth allocation policies.*

Question 2 *What quality of solution and running time bounds can we obtain in this model?*

Question 3 *Are the algorithms we design sufficiently simple and efficient to warrant practical implementation?*

The next step is to extend these models and results to the setting in which multicast connections are also present in the network. We formalize a model in which to study allocations to multicast connections in Chapter 5. There, we explore the tradeoff between the optimality of the allocation and the distributed running time in this model for both multicast bulk applications and multicast streaming applications.

1.3 Data Selection Policies

The next problem we consider is the on-line scheduling problem which the sender of a unicast (single destination) bulk transmission faces: Given a set of fixed-size words $w_1, w_2, \ldots, w_M$ comprising a message and a sequence of events $\rho_1, \rho_2, \ldots, \rho_N$ which either request the sender to send a word, acknowledge the receipt of a word, or acknowledge that a word has been discarded, the sender must decide what data to transmit at each send request. In this on-line setting, the sender must determine which data to place into each packet of the transmission with imperfect information about the future. A primary objective of the sender is to minimize the completion time; we are interested in the more stringent requirement of making the most effective use of the available bandwidth at intermediate points of the transmission as well. We measure the performance of our algorithms at these intermediate points in time based on the length of the *prefix* of the message successfully delivered to the receiver.

There are at least three factors which contribute to the sender's uncertainty about the future, all of which the sender *cannot control* when best-effort protocols are employed, and network service is not guaranteed: Any packet which the sender transmits might be discarded. The round-trip time of packets in the network can vary widely over time. The rate at which a sender is allowed to inject packets into the network can increase or decrease based on external factors such as network congestion. Moreover, recent experience in modeling these phenomena indicate that they are extremely difficult to predict. Traces of packet loss on the Internet indicate the presence of frequent, long-lived bursts of packet loss [Pax 96, YKT 96]. Variance in packet latencies and packet interarrival times on the Internet is large [Jac 88, Pax 97]. These factors motivate studying the performance of transmission protocols on arbitrary transcripts of event sequences. In this worst-case model, the algorithm is asked to deliver high performance on any possible sequence of events relative to an optimal algorithm, and is graded based on its worst-case performance.

Although this worst-case specification may seem overly pessimistic, it provides a useful measure of assessment for data selection policies. Our main objective is to develop an algorithm which performs well with respect to the worst-case measure, but a secondary objective is to assess the performance of widely used data selection policies. If they fare poorly in the model, we are interested in understanding why, and consider how to improve them. A common complaint regarding competitive analysis of on-line algorithms is that optimizing worst-case behavior is an ill-advised strategy for practical scenarios which rarely exhibit worst-case behavior. Our response to this complaint is that for the problem we describe, the cost of tolerating worst-case behavior is minimal in terms of typical performance degradation. Moreover, existing algorithms which are not worst-case tolerant incur a large penalty when worst-case behavior arises. Our answers to the following questions will address these considerations:

Question 4 *What worst-case performance can an on-line unicast data selection policy deliver?*

Question 5 *To what extent does our data selection policy degrade performance over existing data selection policies on "typical" network behavior?*

An important consequence of our work is a better understanding of the interplay between bandwidth allocation policies and data selection policies. In general, given knowledge of the allocation policy, it may be possible to fine-tune a data selection policy to deliver higher performance for that particular allocation policy. This is precisely what is done in TCP; the protocol for packet retransmission is carefully intertwined with the policy for congestion control to deliver better performance. However, we prove that such fine-tuning can gain very little when the correct data selection policy is used. The unicast data selection policy we present provides *universally* high performance in conjunction with any allocation policy, and its performance can only be improved by an additive term.

Therefore, bandwidth allocation policies and data selection policies for unicast connections can be viewed as "orthogonal". This observation leads to a better understanding of unicast transmission protocols, in that it gives provable grounds supporting the idea of explicitly separating the two policies. This modular separation has been suggested by other authors in the networking community, including [MM 96], but is not found in the most widely used protocols.

1.3.1 Data Selection Policies for Multicast Connections

For developing data selection policies for multicast connections, the main additional complication is *scalability*. Ideally, we would like the performance of our data selection policies to remain fixed as the number of receivers increases, all other things equal. However, this may not be possible, as a transmitted word which arrives successfully at some receivers may be discarded en route to other receivers, leading to a situation where retransmission of that word benefits some, but not all destination endpoints. In this setting, we prove that any data selection policy which only transmits words of the original message is inherently unscalable, as performance degrades when the number of users increases. Also, it is not the case that we can develop a multicast data selection policy which provides universally high performance in conjunction with any allocation policy, as we prove for unicast. On the positive side, we prove that as long as the allocation regime provides a minimal quality of service guarantee, one can define a *scalable* multicast data selection policy which has worst-case performance within a multiplicative constant factor of optimal. This policy uses *forward error correction (FEC)*, which transmits packets with redundant encoded data along with packets containing words of the original message to recover from packet loss in the network. For multicast connections, we address the following questions:

Question 6 *What level of performance and scalability are achieved with the use of FEC?*

Question 7 *How much overhead is required to use the encoding technology?*

To summarize, the topics described in this thesis focus on implementation of policies which optimize bandwidth utilization for both unicast and multicast bulk-data applications. The first is a network-wide allocation strategy which regulates the *rates* at which applications may inject packets into the system. The second is an application-level strategy which determines the *content* of those packets. These policies can either be used separately, or in conjunction with one another, and both guarantee bounds on worst-case performance.

1.4 Theory vs. Practice

Although our work is theoretical in nature, we place a particularly heavy emphasis on developing algorithms and models which address real problems in existing networks. To this end, the models we develop draw from experience gained from both practical and theoretical research, a fact which is reflected in this introduction and in the material we summarize in the chapter on previous work. We have also implemented and simulated all of the algorithms which we present and incorporate our findings into the chapters which follow.

1.5 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we discuss related work on mechanisms for allocating bandwidth to bulk connections in the network, data selection policies, and multicasting. In each case, we begin by summarizing the state of the art in existing networks, as this viewpoint helps form the basis for our theoretical models. We then discuss related theoretical work on each of the topics.

In the next two chapters, we begin a technical discussion of the bandwidth allocation and data selection policies we have developed in the simpler case in which all applications are unicast applications, i.e. targeted to a single receiver. In Chapter 3, we focus on bandwidth allocation policies maximizing weighted throughput. We show how to formulate the bandwidth allocation problem as a positive linear program and extend the model of Papadimitriou and Yannakakis [PY 93] for solving positive linear programs with distributed decision-makers. In our extension of their model, the distributed decision-makers are the connections and routers comprising the network. Using a natural set of communication primitives, we show how these agents can compute a feasible $(1+\epsilon)$-approximate solution to the global optimum in a polylogarithmic number of distributed communication rounds. This theoretical result appeared in preliminary form in [BBR 97] as joint work with Yair Bartal and Danny Raz. We then provide a discussion of the implications of this technique for solving the bandwidth allocation problem, in part focusing on experience with a serial implementation of the algorithm.

In Chapter 4, we focus on data selection policies. We measure the performance of a data selection policy according to the prefix measure described above, and in comparison to the offline optimum performance obtained by an optimal, omniscient algorithm. We then develop a formalism for analyzing policies according to this measure, based largely on existing network protocols. We prove that no deterministic data selection policy, such as that used in TCP, can deliver worst-case performance whose performance can be bounded by an additive term away from optimal. However, we show that with randomization, we can develop an algorithm which delivers very high performance, within an additive term from optimal at any point in time with high probability. Many of the theoretical results in this chapter appeared in preliminary form in [ABBLR 97] as joint work with Micah Adler, Yair Bartal, Michael Luby and Danny Raz. We bolster our results with comparative simulations of our algorithm against other data selection policies on network traces taken from the Internet and on simulated worst-case behavior.

In Chapter 5, we extend our results to networks which admit both multicast and unicast connections. We show how to incorporate reliable multicast connections into our positive linear programming formulation developed in Chapter 2. We then consider the interesting problem of choosing a data selection policy for multicast connections. As described earlier, it is quite easy to prove that the use of *encoding* is a prerequisite for a high-performance, scalable data selection policy in our model. We describe recent innovations in the area of forward error correction (FEC) which we then use to develop a scalable data selection policy for multicasting which delivers high performance. Again, we present empirical results from Internet trace data and on simulations we generated to demonstrate the high performance of the algorithm relative to existing policies.

We conclude with a discussion of the work described in this thesis, along with directions for future research work in the area in Chapter 6.

Chapter 2

Related Work

In this chapter, we strive to develop a context for the problems we study in this thesis. The work we survey can largely be classified into two distinct categories. The first category consists of networking practitioners' experience with network protocols on existing networks and on simulated networks. We use this work to motivate our model of how components of the network operate and how network traffic behaves. The second category consists of theoretical work and theoretical techniques related to the problems we study and which impact the algorithms that we present. A complete survey of this large body of work is beyond the scope of this chapter; often we summarize research results which directly impact our work. We focus on the four following topical areas.

- Rate-based flow control.
- Theoretical work on related allocation problems.
- Factors adversely affecting data selection policies.
- Multicasting with forward error correction.

2.1 Rate-Based Flow Control

The factors which affect the rate at which a bulk application consumes network bandwidth are:

- the limiting rate at which the sender can inject packets into the network, due to the nature of the application or technological limitations

- the limiting rate at which the receiver can process received data
- the mechanism by which the connection or network regulates flow

In this discussion, we will concentrate on research studying the last of these factors as the first two factors introduce relatively simple constraints which are well understood. First, we note that existing networks do not support rate-based flow control, but rather rely on senders to individually regulate their own rates. For example, the TCP transport protocol uses a fine-grained, per packet strategy to tune the transmission rates at the senders based on factors such as observed congestion in the network. The effects of such sender-driven policies are not germane to our study of rate-based allocation, but do play a role in our work on data selection policies, which we discuss momentarily.

On the other hand, rate-based flow control has become a technique which is widely expected to see broad use in integrated-services networks. Recently, the ATM Forum on Traffic Management has adopted rate-based flow control as a standard for available bit rate traffic on ATM networks (see [BF 95] for details). The work on Phantom of Afek et al [AMO 96b] is a first step towards interoperability of TCP and rate-based flow control. Typically, rate-based flow control is implemented by having routers store data about connections which utilize them. These data values stored in the network can be accessed and modified by control messages which loop through the paths of the connections. For example, a router may stamp a control message from a given connection with an indication that the connection should reduce its transmission rate. When this control message loops back to the source of the transmission, it complies with this request.

Work on rate-based flow control can be traced back to the work of Jaffe [Jaf 81]. In 1981, he proposed an equilibrium condition he coined *max-min fairness* and a scheme for allocating transmission rates to a static set of connections to achieve this equilibrium. From a connection's viewpoint, in a max-min fair allocation, it either receives its desired allocation, or it receives a fair share of the "bottleneck link" of its connection such that no other connection using that link receives a larger share. The max-min fairness criterion has subsequently gained wide acceptance as an appropriate measure of fairness in the networking community.

Jaffe's early work on achieving max-min fairness turned out to be unsuited for implementation in distributed environments as it did not dynamically adapt to changes in network traffic. A large body of research has subsequently worked on remedying this problem. Theoretical results of Afek, Mansour and Ostfeld indicate that convergence to a max-min fair equilibrium can be obtained in $\Theta(n^2)$ distributed update steps, where n is the number of connections in the system [AMO 96a]. Awerbuch and Shavitt [AS 97a] have shown how to arrive at an approximately max-min fair equilibrium in a distributed model in a logarithmic number of distributed rounds. However, in both cases, the mechanism for performing the updates can be quite slow in practice, and in a more practical result of Afek et al [AMO 96b], the authors show how to improve convergence time when a portion of the bandwidth of each link is reserved for update operations, and is not used by any connections. Of course, this strategy incurs a penalty measured in terms of the decrease in network bandwidth available to the set of connections in the system. These complications have led some to question whether max-min fairness is a reasonable objective for rate-based flow control to achieve, but they have not yet proposed alternative objectives. For many related admission control and bandwidth allocation problems, a considerable amount of theoretical work has studied the objective of maximizing network throughput, but to our knowledge, no significant practical work has addressed this objective function for rate-based flow control.

2.2 Theoretical Work on Related Allocation Problems

Theoretical work on bandwidth allocation began with interest on allocating bandwidth to connections so as to maximize network throughput. One of the first theoretical studies in this area for general network topologies was work on the following on-line problem, studied by Awerbuch, Azar and Plotkin [AAP 93]. Given a capacitated network and a sequence of calls C which arrive on-line, each consisting of a source and destination endpoint in the network, a duration, and a profit associated with servicing the call, choose whether to accept the call, by routing it along some source-destination path, or reject the call. In their model, all calls use equal bandwidth and calls which are accepted cannot subsequently be preempted. Their performance objective is to minimize the worst-case ratio over all possible sequences of calls between the profit of the offline optimum and the profit of their on-line algorithm. This measure is the *competitive ratio* of their algorithm. For each call request, their centralized algorithm computes a weighted shortest path from source to destination, using an exponential weighting function on the edge loads as a length function. If the weight of this path is below a threshold, the call is routed on this path, otherwise the call is rejected. The use of the exponential weight function strongly discourages routing additional low-profit calls through edges which already have high load. Ultimately, their algorithm achieves a logarithmic competitive ratio.

In continuation of this work, Awerbuch and Azar [AA 94] studied a range of related resource allocation problems in a distributed setting. In a distributed round of their distributed client-server model, each client can broadcast a message to each of the servers it utilizes, and each server can broadcast a message to each of the clients it serves. One problem they consider in this framework is the bandwidth allocation problem we study, in which they model the autonomous connections as clients and routers as servers. Using their communication model, they prove that in a logarithmic number of distributed rounds, the connections can achieve an allocation of end-to-end rates which is a feasible logarithmic factor approximation to the throughput maximizing allocation.

Another approach for solving resource allocation problems with incomplete information was proposed by Papadimitriou and Yannakakis [PY 93]. They considered the problem of having distributed decision-makers assign values to a set of variables in a linear program (LP), where the agents have limited information about the constraint matrix. In one scenario they describe, each agent, acting in isolation, must set the value of a single primal variable, knowing only the constraints affecting that variable in the LP. When applied to the bandwidth allocation problem we investigate, their model corresponds to a setting in which connections must set their allocation rates based only on knowledge of how many other connections share each of the routers they intend to use. When all edge capacities are 1, their "safe" heuristic sets each connection's allocation to the reciprocal of the maximum number of connections which share an edge with that connection. It is not hard to see that the worst-case approximation ratio achieved by this heuristic is $O(\Delta)$, where Δ is the maximum number of connections that share a router. They also prove that the "safe" heuristic achieves the best possible worst-case ratio when agents may not communicate, leaving open the possibility that much better ratios can be obtained when agents can interact.

2.2.1 Work on Positive Linear Programming

Since the distributed allocation problem we are interested in solving is a positive linear program, a sequential implementation of our algorithm corresponds to an approximation algorithm for positive linear programming. There has been a substantial amount of previous work on efficiently generating approximate solutions to linear programs, and positive linear programs in particular, although not in the model we describe. The polynomial time algorithms for linear programming imply that a centralized administrator with complete information could generate an exact solution for our problem, albeit with substantial computational effort. Recently, algorithms that produce approximate solutions to positive linear programs within a $(1+\epsilon)$ factor of optimal have been developed which are much faster than exact algorithms. The sequential algorithm of Plotkin, Shmoys and Tardos [PST 94] models positive linear programs as restricted flow problems, and in this model, repeatedly identifies a globally minimum weight path, then increases the flow along that path. The algorithm of Luby and Nisan [LN 93] has both a fast sequential and parallel implementation. Their algorithm initializes all variables in the linear program to have very small, feasible values, then increases those values monotonically over time. Selecting *which* variables to increase at each point in time is perfomed basis of a cost measure using exponential weight functions similar to the one used in [AAP 93] and by increasing only those variables with sufficiently small cost. However, this threshold cost evolves over time, and its value both depends on global information and must be known globally, which makes their algorithm unsuitable for distributed implementation. Upon termination, they use the theory of linear programming duality to prove that their algorithm achieves a $(1+\epsilon)$ approximation to the value of the linear program. Although these algorithms have efficient implementations which run in $\tilde{O}(nm)$ sequential time, where $\tilde{O}$ hides logarithmic and $\frac{1}{\epsilon}$ terms, they both perform global operations which make them unsuitable for fast distributed implementation. Clearly, each of the global operations in these algorithms can be implemented in a polynomial number of distributed rounds, in which agents broadcast the values of relevant variables to all other agents. But we are interested in more time-efficient solutions.

The deterministic algorithm we present produces $(1+\epsilon)$ approximate solutions for positive linear programs, both in general and for the bandwidth allocation problem, and builds on ideas used in these other algorithms [AA 94, LN 93, PST 94]. The sequential version of our algorithm is most closely related to the algorithm of Luby and Nisan, and affords the following advantages. It eliminates the need for complex global operations and the enabling fast implementation in a distributed setting. Those simplifications carry over to serial and parallel settings as well, where we have a dramatically simpler implementation which saves a $\frac{1}{\epsilon}$ factor in the running time over the other approximation algorithms. In the distributed setting, we can parameterize the algorithm to quantify a tradeoff between the distributed running time and the quality of the approximation. The algorithm and the analysis are presented in Chapter 3.

2.3 Work Related to Data Selection Policies

In this section, we consider practical work which influenced the model we chose for developing and analyzing performance of data selection policies. Most influential have been recent studies of end-to-end dynamics of connections in the Internet, especially the work of Paxson [Pax 97]. Until quite recently, studies of unicast data selection policies were very limited, perhaps in part due to the satisfactory performance of the simple data selection policy used in most reliable transport protocols: transmit new data unless previously transmitted data has been lost, in which case that lost data should be retransmitted. In general, this simple policy has delivered high performance in protocols such as TCP when packet loss rates are low, but has recently drawn an increasing amount of attention (see for example [Hoe 96]) as the incidence of temporally correlated packet loss increases.

2.3.1 Trends in End-to-End Network Behavior

Paxson's end-to-end measurements of Internet traffic indicate an important trend: over the last several years, packet loss rates experienced by bulk transmissions are increasing, and by many indications, *accelerating* [Pax 97]. Whether this trend will continue is the subject of a great deal of speculation. Another interesting finding regards the distribution of packet loss. While it was once widely held that network traffic characteristics could be accurately modeled with simple Poisson processes, that interpretation has largely been discounted, due in large part to the breakthrough work of [LTWW 95]. Their work and numerous subsequent studies have confirmed that traffic patterns in wide-area networks, packet loss patterns, and other network phenomena are more accurately modeled by a complex, bursty process [WTSW 95, PF 95, YKT 96, Pax 97] in which long-term correlation is present. In particular, the distribution of packet loss is challenging to model, and is not yet completely understood. Also it is not yet known precisely which factors lead to bursty behavior nor whether those factors can be mitigated with the advent of new network protocols and routing technology.

In the realm of multicast applications, Yajnik et al [YKT 96] demonstrated that there was both a high temporal and spatial correlation in loss patterns for receivers listening to audio broadcasts originating in California over the worldwide multicast backbone (MBone). They ascribe high temporal correlation in loss patterns to the same phenomena underlying bursty packet loss observed in unicast settings. By high spatial correlation they indicate that geographically proximate users experienced correlated loss patterns. Their explanation for this phenomenon stems from an understanding of the multicast distribution system – if the single copy of a packet travelling over a given link (for example, a transatlantic link) is dropped, then all receivers downstream of that link will not receive the packet.

Another important consideration is the latency of packets in the system. Although bulk connections are often much more concerned with throughput than latency, fluctuations in packet latency pose difficulty for best-effort protocols such as TCP which use round-trip times as a barometer for adjusting transmission rates. Jacobson's early work on congestion control in TCP [Jac 88] was careful to address this issue, and indications point to a problematic increase in variance both for packet latencies and packet interarrival times [Pax 97].

2.3.2 Theoretical Work

Since these uncertainties underlying our current understanding of packet loss are likely to persist in integrated-service networks, these factors motivate our study of the performance of protocols against arbitrary, rather than probabilistic, behavior by the network. We develop protocols which deliver provably high performance whether packet loss is bursty, random, non-existent, or described by a process we do not understand. Fluctuations in packet latency are also taken to be arbitrary in our model. We are also interested in determining what minimal assumptions about the network we must make to ensure high performance. Of related interest is the extent to which existing protocols succeed in delivering high performance in worst-case environments. While we know of no related work on worst-case behavior of data selection policies, our approach is somewhat similar to the approach used in recent work in the adversarial queuing models of [BKRSW 95] and [AAFKLL 96]. Their work focuses on analysis of queuing policies in packet routing networks when the distribution of packet arrivals and packet routes is arbitrary, subject to the requirement that edge capacity constraints are not exceeded. In this manner, they consider the performance of queuing policies given worst-case arrival patterns. Our work on competitive analysis of data selection policies uses the standard definitions initiated by the work of Sleator and Tarjan [ST 85], but we defer discussion of these definitions to Chapter 4.

2.3.3 Multicasting

In a multicasting environment, there are additional considerations which data selection policies and flow control policies must address. First and foremost, as the number of receivers grows large, *scalability* of the protocol becomes a problem. If each receiver acknowledges receipt of each packet as in TCP, the number of acknowledgments routed to the sender becomes unmanageably large, and a phenomenon known as *feedback implosion* results. Furthermore, packets lost at a given receiver are not necessarily lost elsewhere, so unlike the situation in TCP, retransmission of those packets would waste bandwidth. Moreover, with a set of receivers with heterogeneous receive rates, it is typically a poor idea to tailor the multicast transmission rate to the worst-case users.

One natural approach for handling the feedback implosion problem is the use of random sampling, whereby only a small subset of receivers transmits feedback to the source at any point in time. Yavatkar and Manoj [YM 93] define a Quasi-reliable Multicast Transport Protocol (QMTP), which performs rate-based flow control with selective receiver feedback employing this technique. Unfortunately, their solution does not completely solve the implosion problem, moreover the limited feedback received may not be an accurate reflection of the multicast group at large.

Another approach is to use *layering*, in which a single transmission is striped across many different multicast groups, or layers. With this approach, receivers can subscribe to subsets of the layers to receive the transmission at various rates. Typically this approach is best suited towards unreliable streaming applications. In recent implementation-based work, McCanne et al [MJV 96] developed a receiver-driven layered multicast protocol (RLM) for packet video in which receivers can dynamically join (or depart from) additional layers of the transmission each improving the signal-to-noise ratio of the overall transmission. Since this protocol is run at the receivers, the receivers perform flow control and feedback implosion is not an issue. A similar approach has recently been put forth for carefully organizing data in layers for *reliable* bulk multicast transmissions by [Vic 97], although this scheme does not seem suitable for dynamic rate adjustment.

A third approach is the use of *forward error correction* (FEC). The idea underlying the use of FEC is to have the sender transmit message data with some additional redundant data, which the receiver can use to reconstruct the original data in the event that some of it is lost. Moreover, decoding is successful independent of which particular subset of message words in a block are lost, as long as sufficiently many redundant packets from the block are received. The immediate benefits of using FEC for reliable multicast are that retransmissions are unnecessary when the receiver receives enough data to reconstruct each message block, and far fewer acknowledgments are necessary. Also, the redundant FEC data can be used to recover from *different* losses at different receivers, giving greatly improved scalability.

Until very recently however, the performance of FEC schemes in practice had been too slow to warrant use except to recover from a very small proportion of packet loss. Those algorithms used Reed-Solomon based codes for the encoding, (see [MS 77] or Chapter 5 for a description of Reed-Solomon codes) and the fastest algorithms for performing the decoding in practice had a quadratic dependence on the size of the encoded message block in the running time. Work by Rizzo [Riz 97] describes such an encoding scheme to perform reliable multicast in settings where packet loss rates are low (between 1 and 5%). Nonnenmacher, Biersack and Towsley [NBT 97] define algorithms for reliable multicast with FEC and analytically demonstrate the benefit of FEC schemes over retransmitting lost packets for a variety of packet loss scenarios. The work of Rizzo and Vicisano [RV 97] defines a Reliable Multicast Data Distribution Protocol (RMDP) which use Reed-Solomon FEC codes to perform reliable multicast by (repeatedly) transmitting source data along with a set of redundant codewords. Receivers with heterogeneous receive rates simply receive words from the stream until they are able to reconstruct the message. However, the dependence of the decoding time on the message and redundancy block lengths still makes algorithms based on Reed-Solomon codes unsuitable for practical use in many interesting scenarios. For example, these limitations make it infeasible to code for substantial packet loss rates, such as the 25% loss rates with large bursts observed by Yajnik et al [YKT 96] for overseas receivers in their trace data.

Recently, Luby et al [LMSSS 97] have developed a randomized FEC scheme with encoding and decoding time linear in the length of the transmission. To obtain a high probability guarantee that decoding is successful, this algorithm requires slightly more redundancy (on the order of a few percent more) than packet loss to ensure successful reconstruction of the message, but the scheme works for arbitrary packet loss rates and runs in linear time. Their work opens the way for practical use of FEC in multicast scenarios where loss rates can be dramatically higher than could previously be tolerated. We describe their construction in more detail and our use of these codes to realize scalable data selection and flow control policies for multicast connections in Chapter 5.

Chapter 3

Throughput Maximizing Bandwidth Allocation Policies

Let us now revisit the formulation of the bandwidth allocation problem we are interested in solving. We start with the static scenario in which each of n bulk connections intends to transmit data along a fixed path in the network, represented by an ordered subset of the m routers which comprise the network. Each of the routers i has capacity C_i, which it may share among the connections which utilize it, and the objective is to maximize the total throughput of the set of connections. If we associate the variable y_j with the rate assigned to connection j, a rate we allow to be an arbitrary non-negative quantity, an optimal allocation is the solution to the following linear program:

$$\begin{aligned} \max \sum_{j=1}^{n} y_j \text{ s.t.} \\ \forall i, \sum_{j=1}^{n} \tilde{a}_{ij} y_j \leq C_i \\ \forall i, j, \tilde{a}_{ij} = 0 \text{ or } 1 \\ \forall j, y_j \geq 0 \end{aligned}$$

In this program, the 0/1 indicator variables $\tilde{a}_{ij}$ reflect whether connection j utilizes router i. The objective function maximizes the total rates assigned to the connections and the first set of constraints ensure that the capacity constraints are not violated.

An important aspect of our algorithm for solving this linear program on our distributed network involves linear programming duality. The *dual* program written below is a minimization problem whose optimal solution is equal to the optimal solution of the *primal* program written above, provided both programs have a feasible solution.

$$\begin{aligned} \min \sum_{i=1}^{m} C_i x_i \text{ s.t.} \\ \forall j,\ \sum_i \tilde{a}_{ij} x_i \geq 1 \\ \forall i, j,\ \tilde{a}_{ij} = 0 \text{ or } 1 \\ \forall i,\ x_i \geq 0 \end{aligned}$$

The interpretation of the *dual variables* x_i in our problem are non-negative "weights" assigned to the routers. The first set of constraints in the dual program ensure that for any connection in the network, the sum of the weights on the routers used by that connection is at least 1.

We can also generalize the objective function when we wish to weight the relative throughput to connections by introducing positive constants B_j and maximizing $\sum_j B_j y_j$ in the primal program, a modification which does not affect the analysis. A more convenient representation of this more general program can be written in a standard form obtained by transforming the values in the constraint matrix by the operation $a_{ij} = \frac{\tilde{a}_{ij}}{B_j C_i}$, resulting in the program:

PRIMAL	DUAL
$\max Y = \sum_{j=1}^{n} y_j$ s.t.	$\min X = \sum_{i=1}^{m} x_i$ s.t.
$\forall i,\ \sum_j a_{ij} y_j \leq 1$	$\forall j,\ \sum_i a_{ij} x_i \geq 1$
$\forall j,\ y_j \geq 0$	$\forall i,\ x_i \geq 0$
$\forall i, j,\ a_{ij} \geq 0$	$\forall i, j,\ a_{ij} \geq 0$

In this program, all members of the constraint matrix $A = \{a_{ij}\}$ are non-negative, and such a program is said to be a *positive* linear program. Moreover, this standard form is well known to be as general as arbitrary positive linear programming.

Chapter Outline

The remainder of this chapter is organized as follows. First, in Section 3.1, we develop a distributed model in which we generate approximate solutions to linear programs of this form. The primary emphasis of our model is to incorporate the constraints imposed by the organization of the network into an abstraction which captures those constraints accurately. In Section 3.2, we present our approximation algorithm first as an easily understandable and implementable sequential algorithm for approximately solving positive linear programming. Then we show that this sequential algorithm can be directly implemented in the distributed model we present. We prove that the algorithm achieves a feasible $(1+\epsilon)$ approximation to the optimum and that it runs in a polylogarithmic number of distributed rounds in Section 3.3. Next, we consider ways in which to augment the performance of the algorithm. We refine the analysis to deliver higher performance when, for example, values of network link capacities span a wide range of values in Section 3.4. We summarize our experience from implementing the algorithm and heurisitcs for improving its performance in practice in Section 3.5. Finally, we describe heuristic techniques for making short-term adjustments to our allocation as connections arrive and depart, and describe how to run our algorithm continuously to help achieve this objective.

3.1 Towards a Distributed Model

3.1.1 Practical Considerations

In recent work on rate-based flow control, the following understanding of the operation of the network communication is generally assumed (see for example [AMO 96a, AMO 96b, AS 97a]). Routers and connection endpoints cooperate in obtaining a distributed allocation, although routers do not have prior knowledge of which connections will be present, nor do connections have prior knowledge of the existence of other connections. Network routers may maintain *state* about the connections in the network which utilize them. However, space limitations make it infeasible for routers to maintain more than a constant number of variables for each connection which uses them, and in fact, some researchers note that scalable algorithms deployed at routers should use only a constant amount of space in all. Moreover, processing time at network routers and connection endpoints is at a premium. Routers and connections endpoints may communicate using *control* *messages*, but only in a limited, and rather indirect way. The source of a connection may transmit fixed-length control packets along the same communication path used by that connection's data packets. These control packets loop through the path and return back to the source. The message contained in a control packet may be read or written to by each and any of the routers on a connection's route. Therefore, a connection may easily transmit a message to the routers along its path. Since routers do not typically initiate transmissions to connection endpoints, a router may only communicate with a given connection by writing into a control packet initiated by that connection. Routers do not communicate directly with other routers, nor do connections communicate with other connections in this framework.

3.1.2 The Theoretical Model

In the linear program, each of the n connections j is associated with the corresponding primal variable y_j and each of the m routers i is associated with the corresponding dual variable x_i. Each connection and router is responsible for setting the value of their associated variable upon termination, such that the joint setting of the variables is a feasible solution to the LP within a $1+\epsilon$ factor of optimal, where ϵ is fixed in advance. The entries of the constraint matrix a_{ij} can be interpreted as specifying the incidence of connections and routers. For any i, j such that $a_{ij} > 0$, we say that router i and connection j are *neighbors*. In one distributed *round*, or *round-trip*, each of the n connections may transmit a fixed-size message which visits all of its neighboring routers before returning to the connection source. Each of the routers en route may read and modify such a control message. The algorithms which we present will assume that distributed rounds are *synchronous*, although we describe how to remove this assumption in our discussion. After a fixed number of rounds, connections and routers must choose feasible values for their variables to (in the case of the connections) minimize the approximation ratio: OPT / $\sum_j y_j$, where OPT is the value of the optimal solution to the LP. We are interested in the tradeoff between the number of rounds and the quality of the approximation ratio obtained.

We assume that all connections and routers initially know the value of m and the desired approximation parameter ϵ, both of which we view as fixed. However, we do not assume that the number of connections n which are present in the system is globally known. Connection j initially knows its benefit B_j and once transmission begins, learns the set of routers in the network on its path, and the capacities of those routers. Therefore, connection j can locally compute the value in standard form of its neighboring constraint coefficients a_{uj} for all u. Likewise, router i initially knows its capacity C_i. Moreover, in one distributed round, each connection j can transmit the value B_j to its neighboring routers. At the end of this round, each router i learns the set of connections which utilize them, and their benefit values, enabling router i to compute the value in standard form of a_{iv} for all v. Upon termination, each connection chooses a feasible transmission rate such that the sum of the rates is within a $1+\epsilon$ factor of the throughput-maximizing allocation.

This theoretical model was developed in the spirit of the work of Papadimitriou and Yannakakis [PY 93], described in Chapter 2, in which distributed agents generate approximate solutions to positive linear programs. In their model, distributed primal agents must independently set the values of their associated primal variable knowing only the constraints of the linear program in which their variable is involved. The objective in their model is to generate a feasible solution with the best possible approximation ratio. In the model we present, distributed primal agents (connections) and dual agents (routers) must set the values of their associated variables after communicating for a fixed number of distributed rounds in a restricted fashion imposed by the organization of the network. The objective in our model is the same, but we study the tradeoff between the number of communication rounds and the quality of the approximation ratio.

3.1.3 Additional Details

We make an additional assumption about the form of the linear program to simplify the analysis of our algorithm. We assume that it is given to the algorithm in a *normalized* form in which the a_{ij} are either 0, or satisfy $\frac{1}{\gamma} \leq a_{ij} \leq 1$. One can convert a problem in standard form to the normalized form simply by dividing all constraints by $a_{max} = \max\{a_{ij}\}$, thereby setting $\gamma = \frac{a_{max}}{a_{min}}$, (where $a_{min} = \min\{a_{ij} | a_{ij} > 0\}$). Performing this transformation is straightforward in both the serial setting and in the distributed setting if bounds on the values of a_{max} and a_{min} are known to all connections and routers in advance. In the context of the bandwidth allocation problem, each connection and router would only need to know bounds on the minimum and maximum values of the capacities C_i and connection weights B_j to compute γ. A disadvantage of this procedure is that the running time of our algorithm depends on the value of γ, which by this procedure depends on the values of the constraint matrix. So we show in Section 3.4 that solving a problem in standard form can be reduced to solving problems in normalized form using a different setting for γ which depends only on m and ϵ and does not significantly affect the approximation ratio or the running time of our algorithm. Moreover, this transformation can be done distributively in a constant number of rounds, without global knowledge of minimum and maximum constraint values, capacities and benefit values. On the other hand, this step adds significantly to the complexity of the analysis and the algorithm. We note that the message size we use in our implementation can be bounded by a number of bits polynomial in $\log m, \log \gamma$ and $1/\epsilon$.

3.2 The Approximation Algorithms

We begin with a high-level description of the algorithms we present for approximately solving positive linear programs. Our algorithm runs in a sequence of phases, which can be implemented distributively or sequentially. At the end of each phase, the non-negative settings for the primal variables y_j and for the dual variables x_i are primal and dual feasible respectively, satisfying the constraints below.

$$\textbf{Primal Feasibility: } \forall i, \sum_{j=1}^{n} a_{ij} y_j \leq 1.$$

$$\textbf{Dual Feasibility: } \forall j, \sum_{i=1}^{m} a_{ij} x_i \geq 1.$$

A hypothetical depiction of the intermediate primal and dual feasible solutions $Y = \sum_j y_j$ and $X = \sum_i x_i$ generated at the end of each phase relative to the value of the optimal solution is shown in Figure 3.2. One invariant that our algorithm maintains, which can be seen from this depiction, is that the value of the intermediate primal feasible solutions is monotonically non-decreasing over time. No such guarantee is provided for the intermediate dual feasible solutions. Linear programming feasibility ensures that values of intermediate primal feasible solutions are necessarily smaller than or equal to the value of the program, denoted by OPT, and similarly, values of intermediate dual feasible solutions are necessarily larger than or equal to the value of the program. Upon termination, we prove that the final (and maximal) primal feasible solution is within a $(1+\epsilon)$ factor of the minimal intermediate dual feasible solution. By linear programming duality, this implies that the final primal feasible solution gives a $(1+\epsilon)$ approximation to the value of the program.

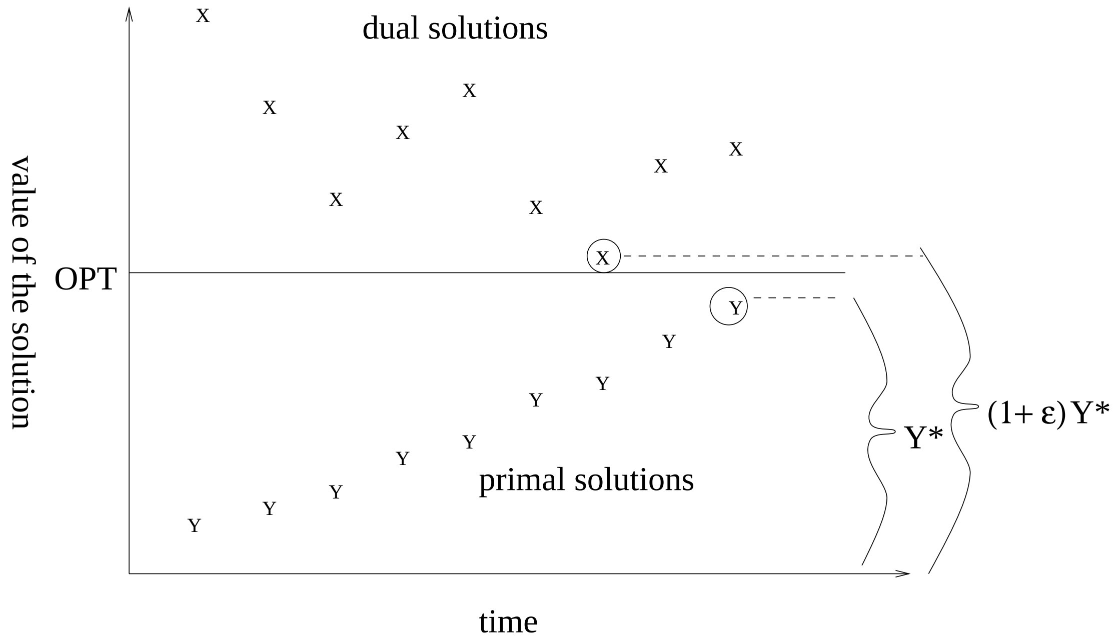

Figure 3.1: Intermediate Primal and Dual Feasible Solutions

3.2.1 Moving Between Pairs of Feasible Solutions

Since the values of primal variables increase monotonically in our algorithm, it is crucial that we choose the "right" variables to increase in each phase as the algorithm progresses to move towards an optimal solution. To this end, our algorithm uses exponential weight functions which have been used before in related contexts by many authors, including [AAFPW 92, AAP 93, LN 93, PST 94, AA 94]. Throughout the algorithm, the values of the dual variables x_i depend on the values of neighboring primal variables y_j by the exponential weighting function: $x_i = \exp(\sum_j a_{ij} y_j \phi)/\psi$, where ϕ is a constant which depends on the desired approximation ratio, and ψ is a scaling factor which grows geometrically with the phase number. In one distributed round, each connection j can then compute the quantity $\alpha_j = \sum_i a_{ij} x_i$. The value of this variable is used as a cost measure for increasing primal variable y_j – the smaller the value of α_j, the more inclined connection j will be to increase y_j. Viewed in the context of our flow control problem, the dual variables x_i are set to a quantity exponential in the load of router i. The quantity α_j sums the exponentiated loads along a connection's path, and connections j with smaller values of α_j are more willing to increase their flow value.

Notice that computing the quantity $\alpha_j = \sum_i a_{ij} x_i$ allows connection j to *witness* one dual constraint. If the value α_j is less than 1, the dual constraint $\sum_i a_{ij} x_i \geq 1$ is violated. This suggests the following idea for an algorithm, which we employ. Start with a pair of dual and primal feasible solutions. Scale down the dual feasible solution by a multiplicative factor, making the solution dual infeasible, and causing some dual constraints to be violated. Then move back to a dual feasible solution by allowing any connection which witnesses a dual infeasible constraint to increase the value of its primal variable. We will refer to a set of increase operations (which can be performed in a distributed manner) as an *iteration* of our algorithm. Routers incident to connections which have increased the values of their primal variables increase the values of their dual variables as prescribed by the exponential weight function. After a certain number of iterations, dual feasibility is again achieved, completing a phase. Primal feasibility is maintained throughout the execution of the algorithm.

3.2.2 The Sequential Implementation

The details of the sequential implementation, complicated primarily by the lengthy initialization procedure, are given in Figure 3.2. Some parameters of the algorithm require further explanation. The parameter r in the algorithm influences the number of phases the algorithm executes, which trades off against the quality of the approximation, in a manner we define precisely in Theorem 8. We prove that after running for a constant number of phases, we obtain a logarithmic approximation ratio, and after running for a logarithmic number of phases, we obtain a $1+\epsilon$ ratio. Furthermore, we show that in each phase, we perform at most a polylogarithmic number of iterations. In the sequential implementation presented in Figure 3.2, the bottleneck operation is to recompute the α_js after each iteration, which takes $O(nm)$ time. In fact, the running time of this bottleneck operation can be more precisely written as $O(E)$, where E is the total number of non-zero entries of the constraint matrix of the linear program. Since this bottleneck operation can be computed in one distributed round in the distributed model, it turns out that this factor of $O(nm)$ will be the difference between the sequential running time (measured in operations) and the distributed running time (measured in rounds).

```
procedure Initialize() {
    δ = (1 + ε)²;      ρ = 1/r;      Q = ρ ln(6γme^ε);
    φ = (r + δ)(Q + ρ ln(Q + ρ ln(2ρQ)));
    ψ = m;      ψ_F = (6mφ)/(r+δ) · e^(δφ/(r+δ));
    ∀i, ñ_i = Σ_{j∈J_i} a_ij;      ∀j, n_j = max_{i∈I_j} ñ_i;
    y_j = ε/(n_j φ);
}

procedure Update-Weights() {
    ∀i, λ_i = Σ_j a_ij y_j;
    ∀i, x_i = e^(λ_i φ)/ψ;
    ∀j, α_j = Σ_i a_ij x_i;
}

Algorithm Sequential-LP-Approx() {
    call Initialize();
    repeat until (ψ > ψ_F) {                          /* Begin a phase */
        call Update-Weights();
        repeat until (min_j α_j ≥ 1) {                /* Begin an iteration */
            ∀j, if (α_j < 1) then  y_j = y_j (1 + ε/φ);
            call Update-Weights();
        }
        ψ = ψ(1 + ε);
    }
    ∀j, output y_j;
}
```

Figure 3.2: The Sequential Positive LP Approximation Algorithm

3.2.3 The Distributed Implementation

Until now, our discussion of the algorithm has centered on the serial implementation. Additional considerations must be taken into account in the definition and description of the distributed algorithm. Since global operations cannot be performed quickly, each connection and router must be able to independently compute the values of all of the variables described in the serial implementation. In the case of parameters which are fixed, such as the value of m, and for the parameters which affect the approximation ratio, r and ϵ, we assume that these values are known to all connections and routers. The parameter γ may not be globally known, but we describe how to specify this parameter in the event that it cannot be locally computed in Section 3.4.2. The one remaining parameter is n, the number of connections. We do not assume that this value is globally known - instead, each connection j uses a value n_j which it can compute locally from its neighbors in two distributed rounds.

The distributed implementation of our algorithm is shown in Figures 3.3 and 3.4. The first half of the implementation specifies the code executed at the routers and the second half specifies the code executed at the connections. Connections and routers communicate by message-passing, in the model of distributed rounds described in Section 3.1.2. For the purpose of defining our algorithm in Figures 3.3 and 3.4, we specify the following operations which connections and routers may perform on control messages. The operation **transmit** (x) can be performed only by a connection and initiates a control message with initial contents x. In the event that the control message is used to *gather* information from neighboring routers, we use the syntax $y =$ **transmit** $(\emptyset)$ to denote the transmission of an empty message, whose contents upon return to the source of the connection are stored in the variable y. A router may perform the operation $x =$ **read** (j) to read the contents of a control message from connection j and store the contents in variable x. Finally, a router may also perform the operation **modify** (j, y) to replace the contents of a control message sent by connection j with data y. Of course, whether or not a router modifies the contents of a message it receives, it subsequently forwards the control message to the next router on the connection's path.

Aside from syntactic considerations, much of the complexity in converting the serial implementation into a distributed implementation involves local synchronization. In our implementation, message-passing primitives enable control to alternate between connections and routers at a local level. This is not to say that control is globally synchronized – in fact, at any instant in time, connections in separate areas of the network might not even be working on the same phase. The other technical obstacle in converting the centralized algorithm to a distributed algorithm is the condition for ending a phase. In the centralized algorithm, a phase terminates when $\min_k \alpha_k \geq 1$. Since we cannot hope to track the value of this global expression in our distributed model, we instead let each connection j check whether $\alpha_j \geq 1$ locally and independently. When the condition is satisfied, connection j terminates its phase, by sending an end-of-phase control message to its neighboring routers. Since those routers may be serving other active connections, connection j must periodicallly poll them with additional control messages until they complete the current phase. The analysis of feasibility and the bounds on the quality of the approximation are identical to those for the centralized algorithm.

procedure Router-Initialize$_i$() { /* Initialization for Router i */

$J_i = \{j | a_{ij} > 0\}$; $\tilde{n}_i = \sum_{j \in J_i} a_{ij}$; $\psi = m$;

for all connections $j \in J_i$ {

$n_j =$ **read**(j);

if $(\tilde{n}_i > n_j)$, **modify**$(j, \tilde{n}_i)$; /* Update n_j if appropriate */

}

for all connections $j \in J_i$, $y_j =$ **read**(j);

}

procedure Router-Update$_i$() { /* Update λ_i and x_i. */

$\lambda_i = \sum_j a_{ij} y_j$; $x_i = \frac{e^{\lambda_i}}{\psi}$;

for all connections $j \in J_i$ {

$\alpha_j =$ **read**(j);

modify$(j, \alpha_j + a_{ij} x_i)$; /* Add our contribution to α_j */

}

}

Algorithm Router-DLP$_i$() { /* Algorithm for Router i */

call Router-Initialize$_i$**();**

repeat until (all $j \in J_i$ terminate) {

call Router-Update$_i$**();** /* Phase begins <u>after</u> this update */

repeat until (all $j \in J_i$ send end-of-phase control message) {

for all active connections $j \in J_i$, $y_j =$ **read**(j);

call Router-Update$_i$**();**

} /* End of phase */

acknowledge end-of-phase to all connections $j \in J_i$;

$\psi = \psi(1 + \epsilon)$;

}

}

Figure 3.3: The Distributed Algorithm at Router i

procedure Connection-Initialize$_j$() { /* Initialization for Connection j */

$I_j = \{i | a_{ij} > 0\}$; $\delta = (1+\epsilon)^2$; $\rho = \frac{1}{r}$;

$Q = \rho \ln(6\gamma m e^{\epsilon})$; $\phi = (r+\delta)(Q + \rho \ln(Q + \rho \ln(2\rho Q)))$;

$\psi = m$; $\psi_F = \frac{6m\phi}{r+\delta} \cdot e^{\frac{\delta\phi}{r+\delta}}$;

$n_j =$ **transmit** $(\emptyset)$; /* $n_j = \max_{i \in I_j} \tilde{n}_i$ */

$y_j = \frac{\epsilon}{n_j \phi}$;

transmit (y_j);

}

procedure Connection-Increase$_j$() {

$y_j = y_j \left(1 + \frac{\epsilon}{\phi}\right)$;

transmit (y_j);

}

Algorithm Connection-DLP$_j$() { /* Algorithm for Connection j */

call Connection-Initialize$_j$**();**

repeat until $(\psi > \psi_F)$ {

$\alpha_j =$ **transmit** $(\emptyset)$; /* $\alpha_j = \sum_i a_{ij} x_i$ */

/* Phase begins after this update */

repeat until $(\alpha_j \geq 1)$ {

call Connection-Increase$_i$**();**

$\alpha_j =$ **transmit** $(\emptyset)$; /* $\alpha_j = \sum_i a_{ij} x_i$ */

} /* End of phase */

transmit end-of-phase control message;

wait until (all routers $i \in I_j$ complete phase);

$\psi = \psi(1+\epsilon)$;

}

output y_j **and terminate**;

}

Figure 3.4: The Distributed Algorithm at Connection j

3.3 Analysis of the Algorithm

In this section, we prove the following main results about our approximation algorithm. The bounds on the approximation ratio hold for both the distributed and sequential implementations. The time bounds are stated for the distributed implementation and we later prove bounds on the running time of the sequential algorithm for approximately solving positive linear programming.

Theorem 8 *For any $0 < \epsilon \leq 1$ and $0 < r \leq \ln(\gamma m)$, the algorithm produces a feasible $(r + (1+\epsilon)^2)$-approximation to the optimum primal linear programming solution, and runs in $O\left(\frac{\ln^2(\gamma m)\ln(\gamma m n/\epsilon)}{r\epsilon^2}\right)$ rounds.*

The following corollary clarifies the tradeoff between the distributed running time and the quality of the approximation and follows directly from Theorem 8.

Corollary 9 *For any $\epsilon \leq 1$, the algorithm produces a $(1+\epsilon)$ approximation to the optimum in $O\left(\frac{\ln^2(\gamma m)\ln(\gamma m n/\epsilon)}{\epsilon^3}\right)$ rounds. For any $1 \leq r \leq \ln(\gamma m)$, the algorithm produces a $(1+r)$ approximation to the optimum in $O\left(\frac{\ln^2(\gamma m)\ln(\gamma m n)}{r}\right)$ rounds.*

Proof: (of Corollary) To prove the first claim of the corollary, set $r = \epsilon$ in Theorem 8, and to prove the second, set $\epsilon = 1$ in Theorem 8. ∎

The remainder of this section is divided into three parts. First we prove a claim we made earlier, that at the end of each phase both the primal and the dual solutions are feasible. From the definition of the algorithm, the values of primal variables increase monotonically, therefore the value of intermediate primal solutions are monotone. Thus to carry out the analysis of the approximation ratio, it remains only to prove that the value of the final primal feasible solution and the value of the minimal dual feasible solution are at most a $(1+\epsilon)$ factor apart. Finally, we prove the claimed running time. In the proof we use the following two facts:

Fact 10 $\phi \geq \epsilon$.

Fact 11 $e^{\phi-\epsilon} \geq \gamma\psi$.

Fact 10 follows from the definition of ϕ in Figure 3.4 and from the fact that $\epsilon \leq 1$. The technical proof of Fact 11 is deferred to the end of Section 3.3.

3.3.1 Feasibility

Recall that the algorithm has the property that at the end of each phase (prior to increasing ψ), the α_js are all increased to at least 1, implying that dual feasibility is achieved:

Fact 12 (Dual Feasibility) *At the end of each phase, for all j, $\alpha_j \geq 1$.*

We next prove that the y_js are primal feasible throughout the execution of the algorithm, using Claim 13 to help perform the induction. Throughout the proof it will be convenient to think of the y_js as initially increasing from 0 to their actual initial value set in the algorithm. We will refer to this operation as iteration 0. Also, we define the load on a router i, $\lambda_i = \sum_j a_{ij} y_j$. Just as the value α_j indicated the feasibility or infeasibility of a dual constraint, if the value of λ_i is greater than 1, the corresponding primal constraint is violated.

Claim 13 *For all i and for every iteration, $\Delta\lambda_i \leq \frac{\epsilon}{\phi}$.*

Claim 14 (Primal Feasibility) *For all i, $\lambda_i \leq 1$ throughout the execution of the algorithm.*

We prove these two claims simultaneously by induction over iterations of the algorithm.

Proof: Let $J_i = \{j | a_{ij} > 0\}$. The first step is to prove that Claim 13 holds for iteration 0:

$$\begin{aligned}\lambda_i &= \sum_{j \in J_i} a_{ij} y_j \leq \sum_{j \in J_i} a_{ij} \frac{\epsilon}{n_j \phi} \\ &\leq \frac{\epsilon}{n_i \phi} \sum_{j \in J_i} a_{ij} = n_i \cdot \frac{\epsilon}{n_i \phi} \leq \frac{\epsilon}{\phi}.\end{aligned}$$

Since $\phi \geq \epsilon$ this also implies that Claim 14 holds at iteration 0.

Consider a subsequent iteration, and let Δv denote the change in variable v during that iteration. We have that for all j, $\Delta y_j \leq y_j \frac{\epsilon}{\phi}$ by the rate of increase in an iteration, so for all i,

$$\Delta\lambda_i = \sum_j a_{ij} \Delta y_j \leq \sum_j a_{ij} y_j \frac{\epsilon}{\phi} = \lambda_i \frac{\epsilon}{\phi} \leq \frac{\epsilon}{\phi}$$

where the final inequality holds by the inductive hypothesis of Claim 14. This completes the proof of Claim 13.

To complete the proof of Claim 14, we consider two cases for λ_i separately. We first consider routers i for which $\lambda_i < 1 - \frac{\epsilon}{\phi}$ prior to an iteration. From the proof of Claim 13 we have that after an iteration on such a router, $\lambda_i' \leq \lambda_i + \frac{\epsilon}{\phi} < 1$, giving the desired result.

Next we consider routers i for which $\lambda_i \geq 1 - \frac{\epsilon}{\phi}$ prior to an iteration. Let i be such an router and fix $j \in J_i$. We have that:

$$\alpha_j = \sum_k a_{kj} x_k \geq a_{ij} x_i = a_{ij} \frac{e^{\lambda_i \phi}}{\psi} \geq a_{ij} \frac{e^{\phi - \epsilon}}{\psi}.$$

By Fact 11, we have that by our choice of ϕ, $e^{\phi-\epsilon} \geq \gamma\psi$, and hence $\alpha_j \geq a_{ij}\gamma \geq 1$, by the definition of γ. By the definition of the algorithm, we never increase the value of primal variable y_j if $\alpha_j \geq 1$, so in fact, no connection in J_i increases its primal variable in this iteration. Therefore, λ_i does not increase during this iteration and remains smaller than 1 by the induction hypothesis, completing the proof. ∎

3.3.2 Proof of a $(1+\epsilon)$ Approximation Ratio

We now turn to bound the approximation ratio obtained by the algorithm stated as the first half of Theorem 8:

Claim 15 *For any $0 < \epsilon \leq 1$ and $0 < r \leq \ln(\gamma m)$, the algorithm produces a feasible $(r + (1+\epsilon)^2)$-approximation to the optimum primal linear programming solution,*

We use the notation $\Delta Y = \sum_j \Delta y_j$ to denote the aggregate change in the y values over the course of an iteration and similar notation for other variables. We begin with the following lemma.

Lemma 16 *For every iteration,*

$$\frac{\Delta X}{\Delta Y} \leq \phi(1+\epsilon).$$

Proof: As $\epsilon \leq 1$, we have from Claim 13 that $\Delta\lambda_i\phi \leq \epsilon \leq 1$. It follows that

$$\begin{aligned} \Delta x_i &= x_i \left(e^{\Delta\lambda_i\phi} - 1\right) \\ &\leq x_i \Delta\lambda_i \phi (1 + \Delta\lambda_i\phi) \\ &\leq x_i \Delta\lambda_i \phi (1+\epsilon) \end{aligned}$$

using the inequality $e^z - 1 \leq z(1+z)$ for $z \leq 1$.

Let $S = \{j | \alpha_j < 1\}$ be the set of active connections in the iteration, i.e. those connections which increase their value of y_j. The lemma follows from the following sequence of inequalities:

$$\begin{aligned}
\Delta X &= \sum_i \Delta x_i \le \sum_i x_i \Delta\lambda_i \phi(1+\epsilon) \\
&= \sum_i x_i \sum_{j\in S} a_{ij} \Delta y_j \phi(1+\epsilon) \\
&= \sum_{j\in S} \Delta y_j \sum_i a_{ij} x_i \phi(1+\epsilon) \\
&= \sum_{j\in S} \Delta y_j \alpha_j \phi(1+\epsilon) \\
&< \Delta Y \phi(1+\epsilon).
\end{aligned}$$

The final inequality holds from the definition of S. ■

In stating and proving the next lemma, we require a more precise description of our notation. We now consider the change in the values of the dual variables X over the course of a *phase*. In the proof, we let X' denote the sum of the x_i at the end of the current phase, and we let X denote the sum of the x_i at the end of the previous phase, i.e. before they are scaled down. We let ΔX denote the change in a the sum of the x_i over the course of the current phase. We further define X^* to be the minimum over all dual feasible solutions obtained at the end of each phase and let Y_L be the primal feasible solution obtained at the end of the final phase. Fact 12 and Claim 14 respectively imply that X^* is dual feasible and Y_L is primal feasible. In conjunction with Lemma 17 below, this implies the approximation result stated in Claim 15, by linear programming duality.

Lemma 17

$$Y_L \ge \frac{X^*}{r + (1+\epsilon)^2}$$

Proof: We prove the lemma for two separate cases; the first being an easy case in which the initial primal feasible solution is a close approximation to the optimum, and the second where we repeatedly apply Lemma 9 to bound the ratio between X^* and Y_L. Let X_0 denote the value for X before the initialization of the y_js, that is $X_0 = \sum_i \frac{e^0}{m} = 1$, and let X_1 denote the value for X after the first phase. Similarly, let X_L denote the value of the dual solution at the end of the last phase

Case I: $X_L \le \frac{r+\delta}{\phi(1+\epsilon)}(X_1 - X_0)$.

Letting Y_1 denote the value of the primal solution at the end of the first phase, we apply Lemma 9 to the iterations comprising the first phase, giving:

$$X_1 - X_0 \leq Y_1 \phi(1+\epsilon).$$

Since the values of the primal feasible solutions increase monotonically throughout the course of the algorithm the Lemma holds by the following sequence of inequalities:

$$X^* \leq X_L \leq (r+\delta) Y_1 \leq (r + (1+\epsilon)^2) Y_L.$$

Case II: $X_L > \frac{r+\delta}{\phi(1+\epsilon)}(X_1 - X_0)$. Since the values X are scaled by a $\frac{\psi}{\psi'} = \frac{1}{1+\epsilon}$ factor just following the end of each phase, the earlier definitions imply that:

$$X' = X\frac{\psi}{\psi'} + \Delta X.$$

By rewriting this expression and applying the inequality $e^z \geq 1 + z$, we have:

$$X' = X\frac{\psi}{\psi'}\left(1 + \frac{\Delta X(1+\epsilon)}{X}\right) \leq X\frac{\psi}{\psi'}\left(e^{\frac{\Delta X(1+\epsilon)}{X}}\right)$$

Now, using $X^* \leq X$ and applying Lemma 16 yields

$$X' \leq X\frac{\psi}{\psi'}\left(e^{\frac{\Delta Y \phi(1+\epsilon)^2}{X^*}}\right).$$

Let ψ_1 and ψ_L to be the initial value of ψ and the value of ψ in the last phase of the algorithm, i.e. $\psi_L < \psi_F$. Using the bound above repeatedly to compare X_L with X_1 gives us:

$$X_L \leq X_1\frac{\psi_1}{\psi_L}\left(e^{\frac{Y_L \phi(1+\epsilon)^2}{X^*}}\right). \tag{3.1}$$

Since X_L is dual feasible and the optimal solution is bounded below by 1 (by the normalized form of the program) we have that $X_L \geq 1 = X_0$. Also note that by the assumption $r \leq \ln(\gamma m)$, we have $\phi \geq (r+\delta)$. We can now use this fact in conjunction with the fact that $X_L > \frac{r+\delta}{\phi(1+\epsilon)}(X_1 - X_0)$. to obtain

$$X_1 < X_L\left(\frac{\phi(1+\epsilon)}{r+\delta} + 1\right) \leq X_L\frac{\phi(2+\epsilon)}{r+\delta}.$$

Using the bound above in (3.1) and observing that $\psi_1 = m$ and $\psi_L \geq \psi_F/(1+\epsilon)$ we get

$$\begin{aligned} e^{\frac{Y_L \phi(1+\epsilon)^2}{X^*}} &\geq \frac{\psi_F}{m^{\frac{\phi(2+\epsilon)}{r+\delta}}(1+\epsilon)} \\ &\geq \frac{6m^{\frac{\phi}{r+\delta}} e^{\frac{\delta\phi}{r+\delta}}}{m(2+\epsilon)(1+\epsilon)\frac{\phi}{r+\delta}} \\ &= \frac{6e^{\frac{\delta\phi}{r+\delta}}}{(2+\epsilon)(1+\epsilon)} \\ &\geq e^{\frac{(1+\epsilon)^2\phi}{r+(1+\epsilon)^2}} \end{aligned}$$

where the final inequality holds by substituting $\delta = (1+\epsilon)^2$, and using $\epsilon \leq 1$. We therefore get

$$\frac{Y_L}{X^*} \geq \frac{1}{r+(1+\epsilon)^2}$$

concluding the proof of the Lemma for Case II. ■

3.3.3 Running Time

For the distributed algorithm, we prove the second half of Theorem 8, bounding the amount of time before termination:

Claim 18 *The distributed algorithm runs in* $O\left(\frac{\ln^2(\gamma m)\ln(\gamma mn/\epsilon)}{r\epsilon^2}\right)$ *rounds.*

Since a distributed round of our algorithm can be implemented in O(nm) time by a sequential algorithm, we have the following bound on the running time of a sequential algorithm for arbitrary positive linear programming.

Corollary 19 *The sequential algorithm runs in* $O\left(\frac{nm\ln^2(\gamma m)\ln(\gamma mn/\epsilon)}{r\epsilon^2}\right)$ *time.*

Proof: We bound the number of phases by measuring the change in ψ. The equalities below follow from the definitions in the initialization step of our algorithm in Figure 3.4.

$$\left\lceil \log_{1+\epsilon}\left(\frac{\psi_F}{\psi_0}\right)\right\rceil = O\left(\frac{\frac{\delta\phi}{r+\delta} + \ln(\frac{\phi}{r+\delta})}{\epsilon}\right) = O\left(\frac{\ln(\gamma m)}{r\epsilon}\right).$$

We now bound the number of iterations in a phase by computing the maximum number of iterations needed to increase all α_j values above 1. In particular, we prove that if a connection y_j participates in a polylogarithmic number of iterations in a phase, α_j increases above 1.

For a given j, we say that y_j is *large* once $y_j \geq \frac{\gamma}{\epsilon^2}$. Initially, $y_j = \frac{\epsilon}{n_j \phi}$ and at every iteration it increases by a factor of $1 + \frac{\epsilon}{\phi}$. Therefore the number of iterations y_j can participate in before y_j becomes large is at most

$$\log_{1+\frac{\epsilon}{\phi}} \left(\frac{\gamma}{\epsilon^2} \cdot \frac{n_j \phi}{\epsilon} \right).$$

Now once y_j is large, we have that

$$\Delta y_j \geq y_j \cdot \frac{\epsilon}{\phi} \geq \frac{\gamma}{\epsilon \phi}.$$

Let the set $I_j = \{i | a_{ij} > 0\}$. Therefore for all $i \in I_j$:

$$\begin{aligned} \Delta \lambda_i &\geq \sum_k a_{ik} \Delta y_k \geq \frac{1}{\gamma} \Delta y_j \geq \frac{1}{\epsilon \phi} \qquad \text{and} \\ \alpha_j^{'} &= \sum_i a_{ij} x_i' = \sum_i a_{ij} x_i \cdot e^{\Delta \lambda_i \phi} \geq \alpha_j \cdot e^{1/\epsilon}. \end{aligned}$$

Therefore, after $\epsilon \ln(\gamma m(1+\epsilon))$ additional iterations:

$$\alpha_j^{'} \geq \alpha_j \cdot \gamma m (1 + \epsilon)$$

where α_j and α_j' denote values at the start and end of these iterations respectively. At the beginning of the first phase, all $\alpha_j \geq \frac{1}{m\gamma}$, since all x_i are initialized to $\frac{\epsilon^0}{m} = \frac{1}{m}$. At the beginning of subsequent phases, all $\alpha_j \geq \frac{1}{1+\epsilon}$, by Fact 12. So, from the discussion above, $\epsilon \ln(\gamma m(1+\epsilon))$ iterations suffice to ensure $\alpha_j \geq 1$ for all j.

The bound on the number of iterations during a phase is therefore:

$$\begin{aligned} &\left\lceil \log_{1+\frac{\epsilon}{\phi}} \left(\tfrac{\gamma \phi n}{\epsilon^3} \right) + \ln(\gamma m (1+\epsilon)) \epsilon \right\rceil \\ &= O\left(\tfrac{\phi}{\epsilon} \ln (\tfrac{\gamma m n}{\epsilon}) + \ln(\gamma m) \right) \\ &= O\left(\frac{\ln(\gamma m) \ln(\frac{\gamma m n}{\epsilon})}{\epsilon} \right) \end{aligned}$$

With the bound on the number of phases this completes the proof of Claim 18 and Theorem 8. ∎

Proof of Fact 11

We have to prove that $e^{\phi-\epsilon} \geq \gamma\psi$. Substituting ψ_F for ψ and multiplying by e^{ϵ}, we must show

$$\begin{aligned} e^{\phi} &\geq (\gamma e^{\epsilon})\frac{6m\phi}{r+\delta}e^{\frac{\delta\phi}{r+\delta}} && \text{or} \\ e^{\frac{r\phi}{r+\delta}} &\geq (6\gamma m e^{\epsilon})\left(\frac{\phi}{r+\delta}\right) && \text{or} \\ e^{\frac{\phi}{r+\delta}} &\geq \left(6\gamma m e^{\epsilon}\frac{\phi}{r+\delta}\right)^{\frac{1}{r}} \end{aligned}$$

Now by taking the natural logarithm of both sides above, we need to show:

$$\frac{\phi}{r+\delta} \geq \frac{1}{r}\left[\ln\left(6\gamma m e^{\epsilon}\right) + \ln\left(\frac{\phi}{r+\delta}\right)\right]$$

Recall from the definitions that $\rho = \frac{1}{r}$, $Q = \rho\ln(6\gamma m e^{\epsilon})$, and $\frac{\phi}{r+\delta} = Q + \rho\ln(Q+P)$, where $P = \rho\ln(2\rho Q)$. Therefore it is enough to prove the following.

$$\begin{aligned} &Q + \rho\ln(Q+P) - Q - \rho\ln(Q+\rho\ln(Q+P)) \\ &= \rho\ln\left(\tfrac{Q+P}{Q+\rho\ln(Q+P)}\right) \geq 0. \end{aligned}$$

Since Q, P and ρ are clearly non-negative, to show that $\rho\ln(\frac{Q+P}{Q+\rho\ln(Q+P)})$ is non-negative we need $P \geq \rho\ln(Q+P)$ or, substituting for P,

$$\rho\ln(Q+P) - \rho\ln(2\rho Q) = \rho\ln(\tfrac{Q+P}{2\rho Q}) \leq 0.$$

It is left to show then that $\frac{Q+P}{2\rho Q} \leq 1$, but

$$\begin{aligned} &\tfrac{Q+P}{2\rho Q} = \tfrac{Q+\rho\ln(2Q)+\rho\ln(\rho)}{2\rho Q} \\ &\leq \tfrac{1}{2\rho} + \tfrac{\ln(2Q)}{2Q} + \tfrac{\ln(\rho)}{2Q} \leq \tfrac{1}{2\rho} + \tfrac{1}{e} + \tfrac{\ln(\rho)}{2\rho}. \end{aligned}$$

The final inequality follows from the fact that $Q \geq \rho$ and from $\frac{\ln x}{x} \leq 1/e$. Since $1 + \ln\rho \leq \rho$ we can further simplify the expression as:

$$\frac{Q+P}{2\rho Q} \leq \frac{1}{2} + \frac{1}{e} < 1$$

which completes the proof. ∎

3.4 Improvements to the Running Time

We describe modifications of the linear program and the analysis which enable us to implement our algorithm when the value of γ is not globally known or when we wish to eliminate the dependence of the running time on values of the matrix, i.e. when setting $\gamma = \frac{a_{max}}{a_{min}}$ is unacceptable for performance reasons. While this technique adds only moderate complexity in the sequential setting, considerable legwork, and an extra $\frac{1}{\epsilon}$ factor in the running time is required to perform these transformations in our distributed model. We sketch the procedure in Section 3.4.2.

3.4.1 A Special Form of the Linear Program

When we wish to avoid setting $\gamma = \frac{a_{max}}{a_{min}}$, we can transform an LP in standard form to a *special form* (similar to one used by Luby and Nisan) in place of the transformation to *normalized form* described in Section 3.1.2. A precondition for transforming an LP Z in standard form to an LP Z' in special form is that we can approximate the value of the optimal solution for Z to within a factor of τ: $c \leq \text{OPT} \leq c\tau$. If this precondition is satisfied, we can perform the following transformation, which bounds the value of the a'_{ij} in Z' by $\frac{\epsilon^2}{m\tau} \leq a'_{ij} \leq 1$ for all i and j, giving $\gamma' = \frac{m\tau}{\epsilon^2}$. Note that the value of γ now depends on the extent to which we can bound the value of the LP, but not on the relative values of the constraints (which could be very small).

Define $\nu = \frac{m}{c\epsilon}$, and perform the following transformation operation on the constraints:

$$a'_{ij} = \begin{cases} \frac{\epsilon}{\nu\tau c} & \text{if } a_{ij} < \frac{\epsilon}{c\tau} \\ 1 & \text{if } a_{ij} > \nu \\ \frac{a_{ij}}{\nu} & \text{otherwise} \end{cases}$$

This transformed LP has the following properties:

1. If $\{y'_j\}$ is a primal feasible solution for Z' then

$$\left\{ y_j = \begin{cases} 0 & \text{if } \exists i \text{ such that } a'_{ij} = 1 \\ \frac{y'_j}{\nu} & \text{otherwise} \end{cases} \right\}$$

is primal feasible for Z, and $\sum_j y_j \geq \frac{\sum_j y'_j}{\nu} - \epsilon \cdot \text{OPT}$.

2. If $\{y_j\}$ is primal feasible for Z then $\left\{y'_j = \frac{y_j \nu}{1+\epsilon}\right\}$ is primal feasible for Z', and $\sum_j y'_j = \sum_j \frac{y_j \nu}{1+\epsilon}$.

3. $\frac{\epsilon}{\nu \tau c} \leq a'_{ij} \leq 1$ for all i and j.

Property 3 is clearly true, but the other two properties require the short proofs below.

Proof: (of Property 1)

Take a feasible solution $\{y'_j\}$ for Z' and let $\{y_j\}$ be as specified. Then for any fixed value of i,

$$\begin{aligned}
\sum_j a_{ij} y_j &= \sum_{j | a'_{ij} < 1} a_{ij} y_j \leq \sum_{j | a'_{ij} < 1} \nu a'_{ij} y_j \\
&\leq \sum_{j | a'_{ij} < 1} \nu a'_{ij} \frac{y'_j}{\nu} \leq 1.
\end{aligned}$$

The final inequality holds by the feasibility of the solution $\{y'_j\}$, so the solution $\{y_j\}$ is feasible for Z. The value of this solution satisfies:

$$\begin{aligned}
\sum_j y_j &= \sum_j \frac{y'_j}{\nu} - \sum_{j | \exists i \text{ s.t. } a'_{ij} = 1} \frac{y'_j}{\nu} \\
&\geq \sum_j \frac{y'_j}{\nu} - \sum_{j | \exists a'_{ij} = 1} \frac{1}{\nu} \\
&\geq \sum_j \frac{y'_j}{\nu} - \frac{m}{\nu} \\
&\geq \sum_j \frac{y'_j}{\nu} - \epsilon \cdot \mathrm{OPT}.
\end{aligned}$$

The first inequality holds by the fact that for any feasible $\{y'_j\}$ and for any i and j, $a'_{ij} y'_j \leq 1$. The second inequality holds by the bound on the number of routers in the network, and the final inequality holds by the definition of ν. ■

Proof: (of Property 2)

Take a feasible solution $\{y_j\}$ for Z and let $\{y'_j\}$ be as specified. Then for any fixed value of i,

$$\begin{aligned}
\sum_j a'_{ij} y'_j &= \sum_j a'_{ij} \left(\frac{y_j \nu}{1+\epsilon} \right) \\
&= \sum_{j | a_{ij} > \frac{\epsilon}{c\tau}} a'_{ij} \left(\frac{y_j \nu}{1+\epsilon} \right) + \sum_{j | a_{ij} \leq \frac{\epsilon}{c\tau}} a'_{ij} \left(\frac{y_j \nu}{1+\epsilon} \right)
\end{aligned}$$

$$\begin{aligned}
&\leq \sum_{j|a_{ij}>\frac{\epsilon}{c\tau}} \frac{a_{ij}y_j}{1+\epsilon} + \sum_{j|a_{ij}\leq\frac{\epsilon}{c\tau}} \frac{\epsilon y_j}{(1+\epsilon)\tau c} \\
&\leq \frac{1}{1+\epsilon} + \frac{\epsilon}{(1+\epsilon)\tau c}\sum_j y_j \\
&\leq \frac{1}{1+\epsilon} + \frac{\epsilon}{(1+\epsilon)} \leq 1
\end{aligned}$$

The final line holds from the bound on the optimal solution for Z: $\sum_j y_j \leq$ OPT $\leq c\tau$, so the solution y'_j is feasible. ■

Now we generate an approximate solution to Z by performing the transformation to special form and then computing a $(1+\epsilon)$ approximation $\{y'_j\}$ for Z' using our algorithm. We transform this solution to $\{y_j\}$ using the transformation described in property (1) and get a primal feasible solution Y for our LP such that:

$$\begin{aligned}
Y &= \sum_j y_j \geq \frac{\sum_j y'_j}{\nu} - \epsilon \cdot \mathrm{OPT} \geq \frac{\mathrm{OPT}'}{\nu(1+\epsilon)} - \epsilon \cdot \mathrm{OPT} \\
&\geq \mathrm{OPT}\left(\frac{1}{(1+\epsilon)^2} - \epsilon\right).
\end{aligned}$$

The first inequality is from property (1), the second is based on the fact that $\{y'_j\}$ is a $(1+\epsilon)$ approximation to the value of Z' (denoted by OPT$'$) and the final inequality is from property (2).

Next we need to explain how to choose the parameters c and τ as to guarantee the precondition: $c \leq$ OPT $\leq c\tau$. Recall that I_j denotes the set of edges incident to connection j: $I_j = \{i|a_{ij} > 0\}$ and J_i denotes the set of connections incident to edge i: $J_i = \{j|a_{ij} > 0\}$. Now define

$$\beta_i = \min_{l\in J_i} \max_{k\in I_l} a_{kl}$$

a quantity which can be locally computed in one round for each router i. Also, let $\beta = \min_i \beta_i$ and for each connection j, define $\widehat{\beta_j} = \min_{i\in I_j} \beta_i$. It is relatively easy to show that $\frac{1}{\beta} \leq$ OPT $\leq \frac{m}{\beta}$. The first inequality holds from the primal feasibility of the solution in which the connection j used in the evaluation of the minimum β_i is assigned flow $y_j = \frac{1}{\beta}$. The second inequality holds from the dual feasibility of the solution in which each router i is assigned weight $x_i = \frac{1}{\beta_i}$.

Therefore, we can set $c = \frac{1}{\beta}$, and $\tau = m$ in the sequential implementation, giving $\gamma' = \frac{m^2}{\epsilon^2}$, and bounding the running time by $\mathrm{O}\left(\frac{nm\ln^2(m/\epsilon)\ln(mn/\epsilon)}{r\epsilon^2}\right)$ time.

3.4.2 Approximating β in the Distributed Setting

In the sequential case, knowledge of β is enough to perform the transformation to special form, but connections and routers may not know this value. We now describe a technique in which we distributively subdivide the LP into subprograms based on local estimates of β. The value of each subprogram is bounded, so we can work in special form. Then, we recombine solutions in such a way as to only assign non-zero rates to connections with good estimates of β, but we prove that this only reduces the sum of the rates by a small factor.

Set $p = \left\lceil \frac{1}{\epsilon} \right\rceil$ and for $q = 0, \ldots, p-1$, define the sets

$$G_t^q = \left\{ j \;\middle|\; \left(\frac{m}{\epsilon}\right)^{p(t-1)+q} \leq \widehat{\beta_j} < \left(\frac{m}{\epsilon}\right)^{pt+q} \right\}$$

for integer t. It is clear that each connection belongs to exactly p of these sets. Independently for each value of q, each router i assigns flow only to connections which are members of $G_{T_{iq}}^q$, where T_{iq} is the minimal value of t for which $G_t^q \bigcap J_i$ is non-empty. In effect, this means that the algorithm is run on the network p successive times. From the connection's point of view, it runs p successive algorithms, using β_j as an approximation for β. In each of these p trials, it can be rejected (i.e. given no flow) by some of the routers. The final flow assigned to connection j is the average of the flows given in the p independent trials. We will prove that this procedure does not decrease the sum of the rates by more than an additional $(1-\epsilon)^2$ factor.

Now define $\mathrm{OPT}(X)$ to be the value of the modified LP when flow can *only* be assigned to connections in the set X. It is not difficult to show that $\mathrm{OPT}(G_t^q)$ is bounded between $(\frac{\epsilon}{m})^{p(t-1)+q}$ and $(\frac{\epsilon}{m})^{p(t-1)+q} \cdot (\frac{m}{\epsilon})^{1/\epsilon} \cdot m$. Thus, we have that for each set G_t^q, the special form of the modified LP for connections in G_t^q satisfies the precondition with $c = (\frac{\epsilon}{m})^{p(t-1)+q}$ and $\tau = (\frac{m}{\epsilon})^{1/\epsilon} \cdot m$. Therefore, for each of these LPs, we can use $\gamma = \frac{m\tau}{\epsilon^2} = (\frac{m}{\epsilon})^{2+1/\epsilon}$.

We now turn to bound the approximation ratio. Consider a particular $q \in \{0, \ldots, p-1\}$, and let T and Q be the unique integers such that β is in the interval defined by G_T^q and $\beta > (\frac{m}{\epsilon})^{pT+Q-1}$. For $q \neq Q$ and for the dual feasible setting $\{x_i = \frac{1}{\beta_i}\}$:

$$\mathrm{OPT}\left(\bigcup_{t>T} G_t^q\right) \quad \leq \quad \sum_{i | i \in I_l, l \in G_t^q, t>T} x_i$$

$$\leq \quad \frac{m}{(\frac{m}{\epsilon})^{pT+Q}} \leq \frac{\epsilon}{\beta} \leq \epsilon \cdot \mathrm{OPT}.$$

This implies that OPT $(G_T^q) \geq (1-\epsilon)$ OPT for all $q \neq Q$. The quality of the solution we obtain is therefore bounded below by:

$$\frac{1}{p} \sum_{q \neq Q} \mathrm{OPT}\ (G_T^q) \geq \frac{p-1}{p}(1-\epsilon)\ \mathrm{OPT} \geq (1-\epsilon)^2\ \mathrm{OPT}.$$

Putting everything together, we have a distributed algorithm that assumes global knowledge only of m and the approximation parameters r and ϵ. This algorithm finds a primal feasible $(r + (1+\epsilon)^5)$-approximation of the optimal solution, and terminates in $\mathrm{O}\left(\frac{\ln^2(m/\epsilon)\ln(mn/\epsilon)}{r\epsilon^4}\right)$ distributed rounds.

3.5 Implementation and Observations

Experimental results confirm the expected efficiency of our sequential algorithm, as defined in Figure 3.2. Output of a typical run of the algorithm on a sparse 20 x 20 constraint matrix with 0/1 entries appears in Figures 3.5 and 3.6. Recall that a 0/1 constraint matrix corresponds to the case in which all routers in the network have equal capacity. For this example, an entry of the matrix was assigned a value of 1 with probability .2 and 0 otherwise. We set the desired approximation ratio to be 1.25, i.e. the feasible solution should be within 25% of optimal. For each phase, we plot the values of the primal feasible and dual feasible solution produced at the end of that phase. A brief inspection of the graph reveals the following interesting facts:

- After only 10% of the run is complete, the intermediate dual feasible solution is within the desired approximation ratio.
- The primal feasible solutions exhibit a nearly linear increase, although the analysis does not indicate such behavior in general.
- The ratio between the best dual solution and the current primal solution fell below the desired approximation somewhat earlier than predicted by the analysis.
- The approximation ratio actually achieved (1.04) was far better than the desired ratio (1.25).

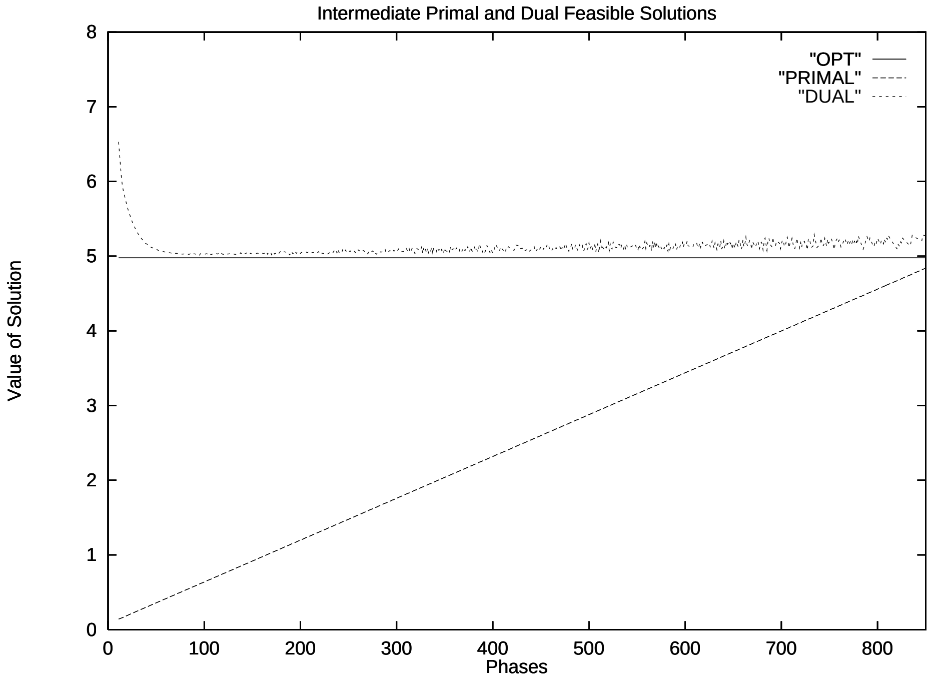

Figure 3.5: Evolution of primal and dual feasible solutions.

- The number of phases may be considered unacceptably large for some practical scenarios.

Not depicted on the graph is the interesting point that an average of only 3 iterations per phase were necessary on all the simulations which we ran, while one might expect far more on the basis of the analysis alone.

The implications of these observations imply that by tuning the algorithm and/or the analysis, we might hope to dramatically speed up the actual running time to make it amenable to practical use. The first observation is interesting – if it is possible to get a close dual approximate solution so quickly, perhaps it is also possible to get a close primal approximation very quickly by dualizing our algorithm. The linear increase exhibited by the primal solutions are interesting in their own right – perhaps it is possible to extrapolate the value of the optimum given the running time and the slope of the increase. Unfortunately, even if we knew the value of the optimum solution exactly, we would still need to find a feasible *setting* of the primal variables which nearly achieves that value. Work on reducing the number of phases to more acceptable levels could proceed on many fronts, and it is in fact heartening to see that the number of iterations needed per phase is small for the

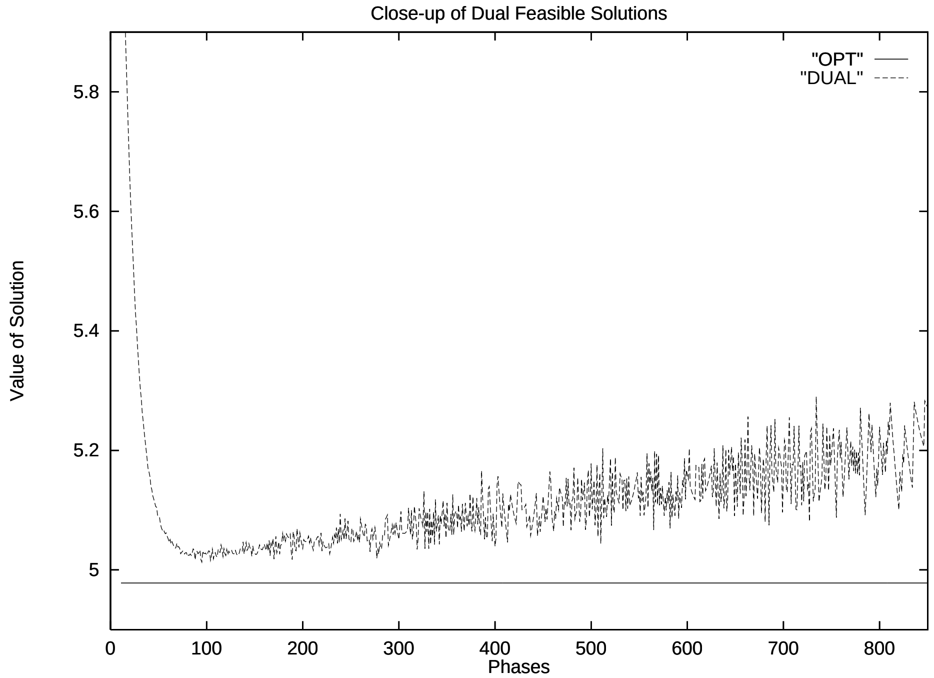

Figure 3.6: Close-up of dual feasible solutions.

simulations we ran. One approach would be to reduce the relative magnitude of Ψ_F in the analysis. Unfortunately, we have not yet succeeded in doing so, nor have we identified heuristic techniques which can reduce the number of phases in practice.

Another technique we can try is to use the observation that in practice, the ratio we actually achieve is far better than the desired ratio. Therefore, in practice, we can set the target approximation ratio to be far higher than the ratio we desire, thereby drastically reducing the number of phases, and exploiting the fact that the actual ratio will be much better. The next two figures indicate the practical potential of this technique. In Figure 3.7, we take a sparse 20 x 20 constraint matrix, except that now we draw the non-zero entries uniformly from the interval (.05, 1), rather than constraining them to be 1. Our selection process ensures that γ, the ratio between the largest and smallest non-zero constraint, is at most 20. Recall that in networking terms, γ corresponds to the ratio between the largest and smallest router capacities. Again setting the desired approximation ratio to 1.25, we see that in Figure 3.7, the number of phases increases somewhat over Figure 3.5, (due to the increase in γ) and we achieve a 1.05-approximation. But, in Figure 3.8, we can take this same constraint matrix, set the desired approximation ratio to 2.5, and still achieve a

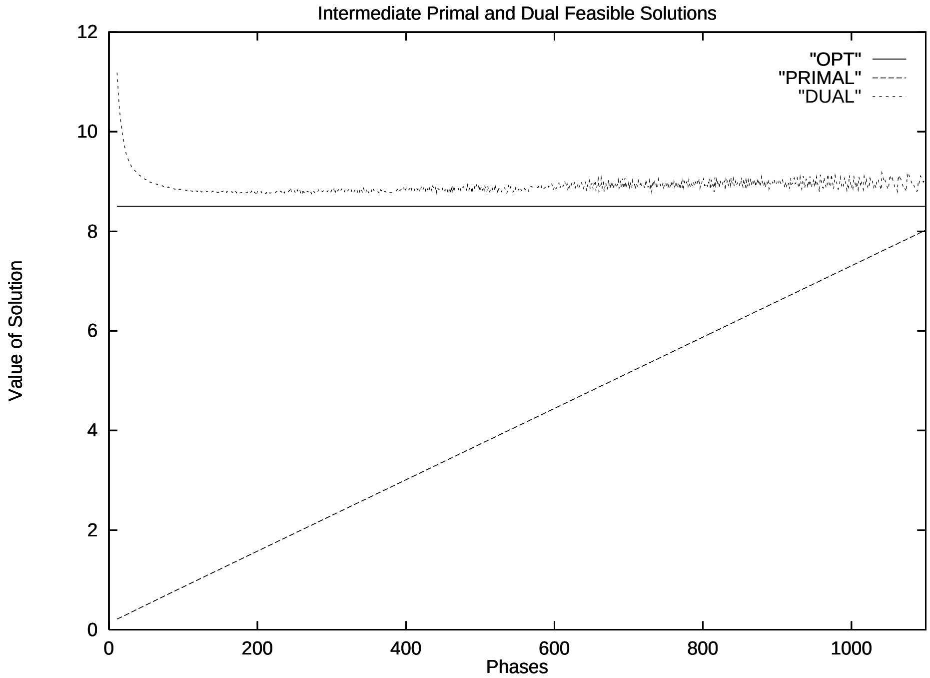

Figure 3.7: Primal and dual solutions for $\gamma = 20$.

1.25 (actually, 1.23) approximation, saving more than a factor of 15 in the running time! Ideally, one could hope to prove that our algorithm always performs this well, but in the meantime, it seems useful to use this heuristic technique to accelerate convergence to the desired approximation ratio.

3.6 Dynamic Settings

The results we have developed so far address the static case of bandwidth allocation: given a set of connections, the benefit function and the capacity constraints, allocate bandwidth to maximize the aggregate benefit. A more interesting scenario, which turns out to be relatively easy to address given the machinery we have so far, is the dynamic situation in which connections arrive and depart as time evolves. Clearly, in this scenario, it is necessary to continuously reappraise the quality of the current allocation, and update the allocation when necessary. One straightforward way in which to achieve this is to have the linear programming approximation algorithm continuously running in the background as time elapses. Each time that the algorithm terminates to give a $(1 + \epsilon)$ approximation

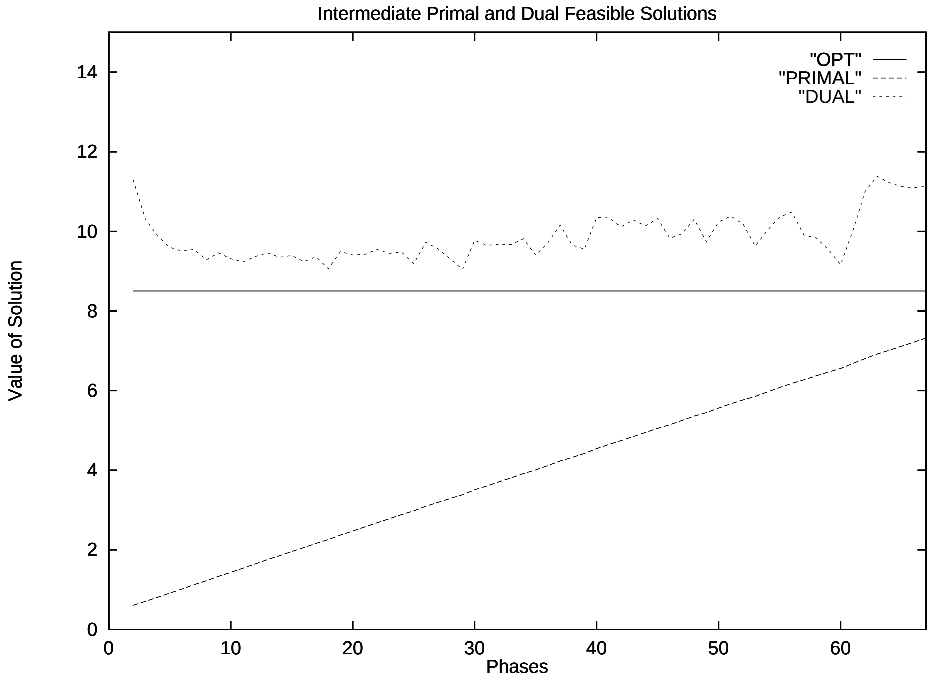

Figure 3.8: Accelerating convergence in practice.

for the linear program that was valid when the algorithm began, the connections move to the resulting allocation. This would also require flexibility on the part of connections, in which they tolerate periodic adjustments to their flow values, a flexibility which bulk connections certainly must have to operate on existing networks. A departing connection simply does not participate in iterations of the algorithm subsequent to its departure, and newly arriving connections must wait until a fresh iteration of the algorithm commences.

The tentative flow values which the routers and connections use in the computation of intermediate solutions to the LP do not necessarily have to correspond to the actual flow values which are used by connections. The flow control algorithm executed by the routers and connections can be thought of as an experiment being performed in the background as flow is being routed through the network. It is also possible in this set-up to use a scheme analogous to a halfspace garbage collection scheme, in which half of the bandwidth reflects an old allocation, and the other half is used to generate a new allocation. Of course, there is nothing which requires that half of the bandwidth be used to compute the new allocation – much slower turnover rates could also be used.

3.7 Discussion

We studied the problem of generating feasible solutions to positive linear programs in a distributed environment, and in particular, the application of flow control. Our results explore the tradeoff between the amount of communication between connections and routers and the quality of the solution we obtain, measured by the approximation ratio. We give an algorithm which obtains a $(1+\epsilon)$ approximation ratio in a polylogarithmic number of distributed communication rounds. Yet, many theoretical and practical questions remain open.

One obvious question is can the running time be improved? It is not difficult to show that $\frac{1}{\epsilon}$ distributed rounds are required to achieve a $1+\epsilon$ approximation, but other less trivial lower bounds are yet to be found. Another interesting theoretical question is the scope of these results, i.e. can they be applied to non-positive LPs?. Finding fast sequential approximation algorithms for general linear programs could be a start in this direction.

On the more practical side, we are interested in implementing our algorithm as a flow control policy. Our preliminary implementation indicates that further work on fine-tuning the algorithm to improve performance could enable the use of the algorithm for flow control on real networks.

Chapter 4

Data Selection Policies

Although explicit rate allocation is widely viewed as a useful technique for providing a guaranteed amount of bandwidth to a bulk connection, most routers which are currently deployed are insufficiently powerful to perform distributed allocations. Rather, connections must cope with the "best-effort" service provided by protocols such as TCP/IP. Use of these protocols introduce the following difficulties. The rate at which a connection may transmit packets may fluctuate rapidly over time, due to congestion control mechanisms. Transmitted packets may be discarded at routers in the event of congestion, as may acknowledgments of successfully received packets. Also, the round-trip times of packets in the system may vary widely over time.

Assuming that a connection does not circumvent these difficulties inappropriately, such as by disabling congestion control, the only recourse which the sender of a bulk connection may use to mitigate these factors is a judicious choice of the *content* of each packet it transmits in an effort to maximize the utility of the data that arrives at the receiver at each point in time. We call this policy for determining the content of transmitted packets a *data selection* policy. This chapter formalizes the on-line scheduling problem which such a policy must solve and develops efficient, competitive on-line algorithms to solve them. For the algorithms we develop, we also prove nearly matching lower bounds on their performance, indicating that only small performance improvements to these algorithms can be made.

The formulation of the unicast data selection scheduling problem is as follows. The sender has a bulk transmission T which is logically decomposed into M words of equal size $w_1, w_2, \ldots, w_M$ each of which fit into the fixed-size payload of a network packet. As time elapses, a sequence of events $\rho_1, \rho_2, \ldots, \rho_N$ occurs at the sender. These events either request the sender to send a word of the message, acknowledge that the receiver has received a previously transmitted word, or acknowledge that a previously transmitted word has been lost in transit. At each of the events which request that a word be transmitted, the sender must select which word to transmit at that point in time. We measure the performance of our algorithm based on the length of the *prefix* of the message acknowledged as received by the receiver at each point in time. Using the tools of competitive analysis in a manner we formalize momentarily, we study the worst-case ratio between the prefix obtained by an optimal algorithm and the prefix obtained by our algorithm, where the worst-case is taken over all legal event sequences and all points in time.

In this chapter, we consider only algorithms which only transmit words which are words of the original message. The motivation for this choice is twofold. First, the algorithms which we develop in this chapter deliver very high performance in this model without resorting to other delivery mechanisms, such as by transmitting redundant codewords. Second, even though the overhead involved in using encoding technology is not very large, it is significant enough that networking practitioners would probably be justified in deeming their use impractical for the relatively modest improvements we could hope to obtain. In the sequel, we show that encoding provides dramatic improvements for data selection policies for multicast connections, and the magnitude of the improvement clearly justifies the additional overhead.

4.1 Motivation

Before we begin the discussion of our approach to studying the data selection policy, we first contrast it with the flow control policy. In flow control, modifications to the allocated rate of a given connection in a highly loaded network has global impact, in that other connections must adjust their rates to keep the network at, but not beyond, full utilization. On the other hand, altering a connection's data selection policy has a much more subtle impact on the other users of the network. Insofar as a connection's data selection policy regulates the *content* of fixed-size packets transmitted through the network, this content is immaterial to other unrelated connections in the network. But the data selection policy can have a global impact, in that the choice of data selection policy can impact the number of packets which are required to complete a bulk transmission. For multicast applications in particular, an appropriate choice for the data selection policy can minimize the impact of a bulk transfer, measured in terms of packet transmissions.

Whereas modifying the flow control policy has an impact on all the users of the network, the data selection policy does not. In a highly loaded network, increasing one connection's flow typically entails decreasing the flow of other connections. But altering a connection's data selection policy has no impact on other users, as the content of one connection's fixed-size packets is immaterial to other, unrelated connections. This indicates an extra flexibility associated with the implementation of a data selection policy, in that one does not need to consider external factors, such as other users, that present obstacles for bandwidth allocation policies.

4.1.1 Modelling Issues

As we described in the section on previous work, the operation of existing networks has become complex and difficult to model, since these networks can have arbitrary topologies, variable edge capacities and network load can be bursty and unpredictable [LTWW 95, YKT 96]. Furthermore, packets traversing these networks can be dropped by intermediate routers due to congestion or can experience unpredictable and widely varying round-trip times [Pax 97]. Therefore, drawing conclusions about a network protocol based on a particular theoretical model or interpretation of network behavior is sharply limited by those assumptions introduced to derive the model. On the other hand, networking practitioners typically tune protocols for deployment in working networks, and consider their design a success when high performance in that setting is achieved. This approach is limited by the degree to which these protocols succeed as the network and the utilization of the network evolves. This considerable and as of today, unquantifiable, discrepancy between conditions under which existing networks operate and conditions under which protocols have been proven to work well motivates our work.

We study the performance of protocols in a worst-case setting where the network topology, the pattern of packet loss and the latencies of packets in the system are determined by an adversary. One of our main goals in using this model is to abstract away the details of a precise prescription of how the network behaves, so as to avoid the contentious problem of settling on a particular statistical model for the behavior of the network and other connections in the system. This is somewhat similar to the approach used in recent work in the adversarial queuing models of [BKRSW 95] and [AAFKLL 96]. Their work focuses on analysis of queuing protocols on packet routing networks even when the distribution of packet arrivals and packet routes is arbitrary, subject to the requirement that edge capacity constraints are not exceeded.

We analyze the performance of our data selection policy in the spirit of *competitive analysis* of on-line algorithms introduced in [ST 85]. The analysis measures the worst-case performance of our data selection policy when compared with the performance of an optimal data selection policy, where comparisons are made at all points in time over each possible event sequence, or behavior of the network. We use the *competitive ratio*, the ratio between the profit of the on-line algorithm and the profit of the optimal algorithm (up to a constant additive term), as our yardstick for performance. We formalize this approach precisely in the next section. A common complaint about competitive algorithms is that they are often impractical, since they optimize the worst-case ratio, and this ratio may be exhibited in examples which do not arise in practice. Moreover, algorithms which do not achieve these worst-case ratios often perform better on instances which actually arise in practice. This is not the case with the competitive data selection policies which we develop. In our analysis, we achieve the best possible competitive ratio of 1, so to provide a tighter analysis which focuses on the *competitive difference* (also referred to as minimax regret) of our algorithm, bounding the worst-case additive term between the optimal profit and the on-line profit. Our nearly tight bounds on the competitive difference imply that the performance of our randomized algorithm is near optimal at any given point in time with high probability. Our empirical results bear this theoretical result out, and show that our algorithm's worst-case performance degrades typical performance very slightly, on the order of 1 to 2 %.

4.1.2 The Prefix Measure

As the problem has been presented, one natural goal of the data selection policy is to successfully transmit the bulk message in its entirety to a receiver in minimum time. We choose to measure the performance of such a data transmission protocol even more stringently: At each point in time, our measure is the length of the longest message *prefix* acknowledged as received up to that point in time. This reflects the reality that during long transmissions, the receiver can often perform useful work on an intact prefix of the completed message.

This choice is motivated by applications such as video and audio broadcasting, where only the intact received prefix of the broadcast can be played back in full fidelity. Another justification for our measure is the strategy used by commercial Internet browsers to display downloaded pages with graphical content: the browser incrementally updates the image as quickly as possible as the stream arrives. In this example, there may be only a small number of breakpoints in the stream where the image can be updated, and thus it may be argued that a more realistic measure would only compare the progress of policies at these breakpoints. On the other hand, since our data selection policy performs well at any possible breakpoint, it clearly also performs well when comparisons are made only at certain breakpoints. Thus, using our stringent performance measure ensures that our data selection policy works well independent of where the breakpoints are within the stream. It is also worth noting that our algorithm performs well even when the receiver derives no benefit at intermediate points in the transmission, since our algorithm is guaranteed to have near-optimal completion time as well.

4.1.3 Benefits

By explicitly decoupling the data selection policy from the other components of a transmission protocol, such as flow control and congestion control, we realize some additional benefits. Our result shows that the performance of the overall transmission protocol would not degrade in performance (and could improve dramatically) if it used our universal data selection policy in place of its own. This gives the main justification for our explicit separation between rate control policies and data selection policies. This should be contrasted with existing protocols such as TCP/IP, in which congestion control, flow control and data selection policies have been interwoven over time to engineer high performance for specific network scenarios.

Our modular approach to protocol design has other obvious advantages: It encourages the design of simple and easy to understand protocols, while explicitly discouraging consideration of complex interactions between the different parts. This notion is not new – the architects of NETBLT [CLZ 87] and the proponents of forward acknowledgment in TCP/IP [MM 96] both advocate explicit decoupling of congestion control algorithms from other parts of the protocol. As in [MM 96], we propose decomposing a transmission protocol into two parts: A rate allocation policy that determines *when* the packets are to be sent, ([MM 96] calls this congestion control in the context of TCP/IP) and a *data selection* *policy* that determines *what* data is placed in each sent packet.

The simple and universal data selection policy we introduce is provably close to optimal in the following sense: For *any* rate allocation policy and *any* network behavior, our data selection policy provably performs almost as well as *any other* data selection policy when used in conjunction with the given scheduling policy and network behavior. This is true even when the data selection policy we compare against is specifically designed to work with the given scheduling policy and network behavior. Therefore it would be natural to use the data selection policy we develop in this chapter in conjunction with the allocation policy described in Chapter 3. But we emphasize that the data selection policy we define would work equally well with other mechanisms for bandwidth allocation or congestion control.

The remainder of this Chapter is organized as follows. In Section 4.2, we define the model, with particular emphasis on our design decisions, and pin down our performance measure. In Section 4.3 we present a deterministic data selection policy and analyze its competitive ratio along with a complementary lower bound for deterministic policies. In Section 4.4 we present a randomized algorithm and prove nearly matching upper and lower bounds on the performance achievable by randomized data selection policies. We then give empirical results demonstrating the advantages and disadvantages of implementing our policy in conjunction with existing network protocols.

4.2 The Specification of the Model

The specification of the model which we develop is *sender-centric*, meaning that the policy is run from the sender's perspective, based only on *events* which have already occurred at the sender. The sender receives feedback about the progress of its transmissions based on receiver-driven acknowledgments in the style of TCP/IP, i.e. each packet arrival at the receiver triggers an acknowledgment. However, both packets and acknowledgments may be lost, so the algorithms which we develop must address this possibility. Assuming that packets, acknowledgments and each of the message words can be uniquely identified, the following is a complete list of events that may occur at the sender at a time-step:

- Transmit packet i containing data w_i.
- Receive acknowledgment ACK (i) acknowledging packet i, or NACK(i), negatively acknowledging packet i.

- Terminate transmission.

Our model supports negative acknowledgments (NACKs), when the receiver can infer that a packet has been lost. It can also support the use of acknowledgments which acknowledge multiple packets, similar to the use of selective acknowledgment in TCP/IP (see for example [FF 96, JB 88]), simply by unpacking such a multiple acknowledgment into a sequence of separate acknowledgments. Although the model implicitly permits the receiver to choose *what* to acknowledge at each time-step, our algorithms use a rather simple mechanism for doing so. We note that studying this acknowledgment policy in models where minimizing the number and size of acknowledgments is rewarded would be an interesting departure point for further study.

We associate the following quantities with packet i:

- s_i, the packet transmission time
- a_i, the time at which the first (possibly negative) acknowledgment for packet i arrives at the sender
- b_i, a boolean variable whose value is known to the sender at time a_i, and has value 1 iff packet i was positively acknowledged.
- $a_i - s_i$, the round-trip time for packet i
- w_i, the data, or payload, contained in packet i

We assume that a rate allocation policy, such as that developed in Chapter 3, is responsible for specifying the times s_i at which the connection may inject a packet into the network. Our data selection policy must choose which message word w_i to place in the payload of a packet for each of these transmission times s_i. Subsequently, at some time a_i, the sender learns whether or not packet i arrived successfully at the destination, and it can compute the round-trip time for packet i. The one constraint we place on legal sequences of events for the competitive analysis is that any packet i which has not been acknowledged at the receiver by time $s_i + D$, for some absolute constant D, is timed out, and presumed lost. This model reflects the use of coarse-grained timers in existing network protocols, which operate under the realistic presumption that badly delayed or misrouted packets have been discarded and are unlikely to arrive at the receiver. For such a packet i, it is presumed to be negatively acknowledged by setting $b_i = 0$ at time $a_i = s_i + D$ and the round-trip time of the packet is D. Since our model allows arbitrary specification of event sequences subject to this constraint, our algorithms must be capable of handling arbitrary sequences of packet and acknowledgment loss, in addition to out-of-order arrivals and widely varying round-trip times.

4.2.1 Transcripts and the Adversary

We define any specification of the s_i and pairs (a_i, b_i) satisfying the constraints above as a *transcript*, on which our data selection policy must operate. Since our policy is designed to guarantee worst-case behavior, we model the transcript as being chosen adversarially. The adversary we use is *oblivious*, i.e. it must specify the transcript in advance of the execution of the algorithm, and may not, in the case of randomized algorithms, adaptively set the event sequence of a transcript as it observes the coin tosses of the data selection policy. In the context of this policy, oblivious algorithms are the correct choice, since networks make packet routing decisions based on packet routes, transmission times and transmission rates, but not based on the content of fixed-size packets. One important quantity we specify for a transcript θ is the *backlog* at time t, $B_t(\theta)$, which we define to be the number of transmissions sent prior to time t which have not yet been acknowledged. We also define $B_\theta = \max_t B_t(\theta)$ and when the transcript θ is understood, we use the shorthand B_t to specify the backlog at some time t and B to specify the maximum backlog. The backlog of a transmission can depend on factors such as the source transmission rate, the network round-trip times experienced by the connection, and the setting for the coarse-grained timeout value. The larger the backlog, the greater the challenge in choosing a data selection policy. For example, if the backlog is always fixed at one, then the trivial policy of selecting the smallest unsent message word is optimal. In fact, in the special case in which the transcript maintains a fixed backlog, a simpler algorithm achieves a slightly better result than the more general one we describe.

In Figure 4.1, we provide a pictorial representation of a transcript. As time elapses along the horizontal axis, we separately plot the transmission and acknowledgment times of the packets. In this particular transcript, acknowledgments arrive in order, and therefore the vertical distance between the two plots at any time t is exactly B_t, the backlog at time t. A complete specification of a transcript would also indicate whether each acknowledgment is a positive or negative acknowledgment.

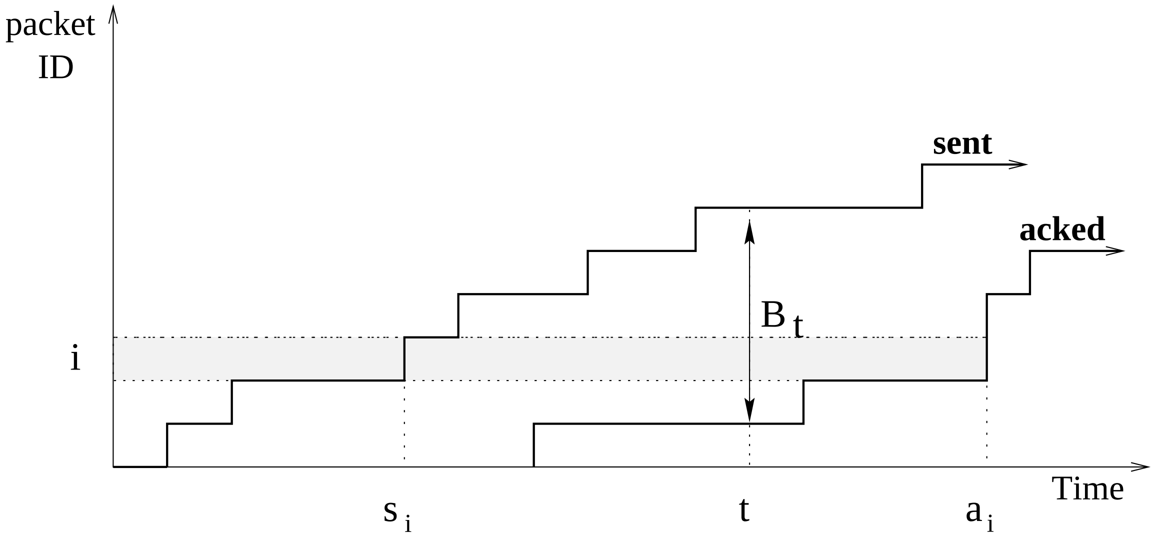

Figure 4.1: A sample transcript

4.2.2 Quantifying the Performance of a Data Selection Policy

To define our performance measure, we need the following definitions: The *available bandwidth* through time t on a transcript θ is $P^*(t, \theta) = \sum_i b_i$, taken over i such that $a_i \leq t$, i.e. $P^*(t, \theta)$ is the number of distinct transmissions which have been positively acknowledged by time t. The *maximum prefix* of the message with data selection algorithm A and transcript θ at time t, denoted $P^A(t, \theta)$, is the largest integer k such that packets containing message words indexed 1 through k have all been positively acknowledged prior to t.

Given any transcript θ, we can define an optimal, omniscient data selection policy, OPT, whose maximum prefix at time t is $P^*(t, \theta)$, i.e. it always manages to use all of the available bandwidth in constructing the prefix. This omniscient algorithm knows the values of a_i and b_i initially, so it knows the fate of each packet i in advance. Therefore, it can optimize use of the available bandwidth simply by transmitting the packet with index j at transmission time s_i for i such that a_i is the jth smallest arrival time among transmissions for which b_i is 1. This off-line optimum forms the baseline for our comparison. For an algorithm A, we assess its performance at time t using the theory of competitive analysis [ST 85] using the definitions and measures below.

Definition 20 *The competitive ratio of a data selection policy A on a transcript θ at time*

t is defined as:

$$R^A(t,\theta) = \frac{P^*(t,\theta)}{P^A(t,\theta)}.$$

The policies we describe for unicast data selection have competitive ratio which goes to 1 as t goes to infinity. Therefore, we measure these policies with respect to the worst-case competitive *difference*, or regret, rather than the competitive ratio using the following measure.

Definition 21 *The competitive difference of a deterministic data selection policy A on a transcript θ at time t is defined as:*

$$\Delta^A(t,\theta) = P^*(t,\theta) - P^A(t,\theta).$$

Similarly, the competitive difference of a randomized data selection policy A on a transcript θ at time t is:

$$\Delta^A(t,\theta) = P^*(t,\theta) - \mathbf{E}[P^A(t,\theta)]$$

where the expectation is taken over the random coin tosses of the algorithm.

4.2.3 Discussion and Statement of the Main Results

First, we present a deterministic data selection policy that achieves a worst-case difference that grows slowly as the length of the transcript goes to infinity. We complement this result with a proof that no deterministic policy can achieve a bounded worst-case difference. These results are stated as the following theorems:

Theorem 22 *There is a deterministic data selection policy A, such that for all transcripts θ and for all times t,*

$$\Delta^A(t,\theta) = O\left((B_\theta \log B_\theta)^{1/2}\sqrt{P^*(t,\theta)}\right).$$

Theorem 23 *No deterministic data selection policy achieves competitive difference which is bounded by a function of B_θ. In particular, there is a transcript θ and a time t such that:*

$$\Delta^A(t,\theta) = \Omega\left(\sqrt{P^*(t,\theta)}\right).$$

Note that in our data selection policies, the backlog of the transcript, B_θ, turns out to be of fundamental importance, as it captures an information gap between our algorithm and OPT, as it measures the maximum number of transmissions whose fate is unknown to our algorithm at any time. For randomized data selection policies, we can do better and focus on bounding the minimax regret, or competitive difference. Our results for randomized policies are summarized by the following theorems:

Theorem 24 *There is a randomized algorithm A, such that for all transcripts θ and at any time t, the following holds with high probability:*

$$\Delta^A(t,\theta) = \tilde{O}(B_\theta \log^2 B_\theta).$$

Theorem 25 *For any randomized algorithm A, and for any sufficiently large integer B, there is a transcript θ with $B_\theta = B$ and a time t_A for which*

$$\Delta^A(t_A,\theta) = \Omega(B \log B), \text{ with constant probability.}$$

4.3 Deterministic Data Selection Policies

In this section we focus on deterministic data selection policies. In this case the transcript uniquely determines the content of the packets. The simplest algorithms such as sending the smallest unacknowledged message word, or repeatedly transmitting the minimal word until it is positively acknowledged, have poor competitive ratios. We present an easily implementable algorithm and prove that it has a competitive ratio that tends to one as the message size goes to infinity. We then prove a nearly matching lower bound, which also shows that the competitive difference for deterministic data selection policies cannot be bounded by any function of B_θ.

4.3.1 A Simple Deterministic Algorithm

Our deterministic algorithm for the problem runs in a sequence of rounds. We say that a word of the message is *completed* once the sender has received a positive acknowledgment for it. The following invariant is maintained upon termination of round j, for all j and for a sequence of monotonically increasing k_j to be specified momentarily:

Invariant for round j: The set of message words $\{W_1, \ldots, W_{k_j}\}$ are completed.

From the invariant, we can infer that the algorithm need only transmit message words with indices in the interval $[k_{j-1}+1, \ldots, k_j]$ in round j. We denote the left and right endpoints of this interval for round j by l_j and r_j, respectively and we let $k_j = r_j - l_j + 1$ denote the width of this interval. During round j, For each index e in $[l_j, r_j]$, the algorithm keeps a reference count f_e. The reference count is incremented when a packet containing the word indexed by e is transmitted, and decremented when a negative acknowledgment for a packet containing word w_e is received at the sender. Thus f_e tracks the number of packets containing w_e that are currently in transit. When a packet containing w_e is positively acknowledged, the word is completed, and the reference count is frozen and does not change thereafter. The algorithm is defined by the following rule: In round j, transmit the word with index $e \in [l_j, r_j]$ with minimum f_e value that has not not yet been completed, breaking ties arbitrarily. Terminate round i when all words in the interval are complete. Also, we impose an integer ranking on the words with indices in $[l_j, r_j]$ based on their completion time, where the word completed first in round j has rank $\rho_e = 1$ and the word completed last has rank k_j, breaking ties arbitrarily. The following lemma is a first step towards the proof of Theorem 22.

Lemma 26 *In round j, for all $e \in [l_j, r_j]$ and for all $d \geq 1$, if $\rho_e \leq k_j - \frac{B_\theta}{d}$ then $f_e \leq \lceil d \rceil$ when word e is completed.*

Proof: Fix some d and consider a word e with final rank at most $k_j - \frac{B_\theta}{d}$ which was completed at time t_f and was last transmitted at time $t_a < t_f$. When it was last sent, there were at least $\frac{B_\theta}{d} + 1$ words which had not yet been completed. The sum of reference counts of this set of words at time t_a is at most B_θ, by the bound on the number of words in transit. Therefore, there must be some uncompleted word at time t_a which has reference count $< d$. Since w_e was sent at this time, $f_e < d$ at time t_a, thus $f_e \leq \lceil d \rceil$ after it was sent and when it is completed, proving the lemma. ∎

To compute a bound on the total number of successful transmissions in a phase, which also serves as a bound on the increase of OPT over the course of the phase, we sum the reference counts of the words in the interval $[l_i, r_i]$:

$$\begin{aligned} \sum_e f_e &= \sum_{e|\rho_e < k_i - B_\theta} f_e + \sum_{a=1}^{\log B_\theta} \sum_{e|\rho_e = k_i - \frac{B}{2^{a-1}}}^{e|\rho_e = \leq k_i - \frac{B}{2^a}} f_e \\ &\leq k_i - B_\theta + \sum_{a=1}^{\log B_\theta} \frac{2^a B_\theta}{2^a} \end{aligned}$$

$$\leq \quad k_i - B_\theta + B_\theta \log B_\theta .$$

Therefore, for any time t in phase i,

$$P^*(t,\Theta) \leq \sum_i k_i + iB_\theta(\log B_\theta - 1).$$

To bound the magnitude of the competitive difference over time, we now take t be a time in phase $i+1$, and specify the k_i so as to minimize the asymptotic growth of $\Delta^A(t,\theta)$:

$$\begin{aligned}\Delta^A(t,\theta) = P^*(t,\Theta) - P^A(t,\Theta) \quad &\leq \quad \sum_{j\leq i+1}(k_j + B_\theta \log B_\theta) - \sum_{j\leq i} k_j \\ &\leq \quad k_{i+1} + (i+1)B_\theta \log B_\theta\end{aligned}$$

Choosing $k_i = i$ we have:

$$\Delta^A(t,\theta) \quad \leq \quad (i+1)(B_\theta \log B_\theta + 1)$$

Now, by using $P^*(t,\theta) \leq \frac{(i+1)(i+2)}{2} + (i+1)B_\theta \log B_\theta$ we can bound the growth of the competitive difference, and prove Theorem 22:

$$\Delta^A(t,\theta) = \mathrm{O}\left(\sqrt{P^*(t,\theta)}(B_\theta \log B_\theta)^{1/2}\right)$$

4.3.2 Deterministic Policies Cannot Achieve Bounded Regret

If the algorithm uses a deterministic strategy to place the data into the packets, then the following adversarial strategy guarantees that $\Delta(t,\theta)$ grows without bound. We say that an adversary *admits* a packet i in a given transcript if the bit b_i is set to 1 on the transcript. The adversarial strategy is to repeatedly allow the algorithm to transmit two packets into the system – if they contain identical words, admit both copies; otherwise admit only the packet containing the word with larger index. The adversary can set the round-trip time of the packet pairs to be zero, so that the algorithm receives instantaneous feedback about each pair of packets. Clearly, the backlog of the transcript resulting from this adversarial strategy with any time-scheduling policy alternates between two and zero. The following claim directly implies Theorem 23:

Claim 27 *For any deterministic data selection policy A, and any transcript θ produced from the adversarial strategy described above, there is an infinite sequence of times t in which $\Delta^A(t,\theta) \geq \frac{1}{2}\sqrt{P^*(t,\theta)}$.*

Proof: For any n, we fix a time τ at which $P^*(\tau,\theta) = 2n^2$. To prove the result, it then suffices to show that $\Delta^A(t,\theta) \geq n$ for some time $t \leq \tau$. Consider any interval of time $[u,v]$ prior to τ in which the algorithm transmits n packet pairs, and let e be the word with minimum index among these $2n$ packets. If the word e is not admitted within these $2n$ transmissions, then clearly $\Delta^A(v,\theta) \geq n$, giving the result. On the other hand, if a packet containing e is admitted, a duplicate packet containing e must also be admitted, by the adversarial strategy. Hence, after admitting $2n^2$ packets, $\Delta^A(t,\theta) \geq n$, proving the result. ■

4.4 Randomized Data Selection Policies

We next want to determine whether better data selection policies can be obtained using randomization. Our goal is to define a randomized policy A which for all times t and transcripts θ, has value of $\Delta^A(t,\theta)$ which is bounded above by a fixed function of B_Θ with high probability.

4.4.1 The Randomized Algorithm

To specify how our algorithm places message words into packets, we use the following terminology. Recall that a word of the message is *completed* once the sender has received a positive acknowledgment for it. We say that it is *locked* for the intervals of time during which a packet containing the word is in transit (sent but not yet acknowledged). Words which are neither completed nor locked are *available*. We distinguish between those available words which have been *previously sent* (and negatively acknowledged), and those which have not yet been sent. Using these definitions, the size of the backlog at time t, B_t, is exactly equivalent to the number of locked words.

We next develop a randomized data selection policy with competitive difference bounded by values which are independent of the schedule length and which satisfies the conditions of Theorem 24. A formal description of Algorithm A is given in Figure 4.2. It can be summarized by the following simple rule:

At the time t of any send event, the candidates for transmission are the set of the $\mathcal{S}(B_t)$ smallest available, previously sent words and the minimum unsent available word.

Each of the candidates which were previously sent are selected at random with uniform probability $\frac{1}{\mathcal{S}(B_t)}$, and the minimum unsent word is sent with the remaining probability (which might be zero). We choose the function $\mathcal{S}(x)$ so that $\sum_i \frac{1}{\mathcal{S}(i)} \leq \frac{1}{2}$, e.g. $\mathcal{S}(x) = \lceil dx \log(x+1) \log\log^{1+\epsilon}(x+1) \rceil + 1$, where d and ϵ are appropriately chosen constants. By this policy, we never retransmit words unless they are known not to have arrived, so redundant (repeated) arrivals are not possible.

We can now precisely state the technical claim from which Theorem 24 immediately follows.

Claim 28 *For any transcript θ, any constant c, and any time τ, let $\widehat{B}_\tau = \max_{t \leq \tau} B_t$ and define $\delta = \lceil 4c\mathcal{S}(\widehat{B}_\tau) \ln 2\mathcal{S}(\widehat{B}_\tau) \rceil$. Then,*

$$\Pr[\Delta^A(\tau, \theta) \geq \delta] \leq (2S(\widehat{B}_\tau))^{-2c+2}.$$

Proof: Fix a transcript θ and a time τ. Now define τ_δ be the first time (prior to τ) at which $P^*(\tau_\delta, \theta) = P^*(\tau, \theta) - \delta$, and let $x = P^*(\tau_\delta, \theta)$. By these definitions, the optimal algorithm delivers a prefix through x by time τ_δ; our goal is to show that our algorithm delivers a prefix through x by time τ with high probability. This would imply that $P^A(\tau, \theta) \geq P^*(\tau, \theta) - \delta$, proving the claim.

We define word i as *eligible* at time t if it is either valid or locked at time t, and we denote the set of eligible words $E_t = V_t \bigcup L_t$. By the fact that our algorithm does not use redundancy, some word $y \geq x$ is transmitted by time τ_δ. At the instant this word y is transmitted, there are at most $\widehat{B}_\tau$ locked words and at most $\mathcal{S}(\widehat{B}_\tau)$ candidates. Therefore, if we let Ψ denote the set of eligible words smaller than x at this same instant, we have that $|\Psi| \leq \widehat{B}_\tau + \mathcal{S}(\widehat{B}_\tau) \leq 2\mathcal{S}(\widehat{B}_\tau)$. We now want to prove that all elements of Ψ are delivered by time τ with high probability, which implies that an intact prefix through x is delivered by time τ. The first step is to show a lower bound on the probability that an eligible member of this set is transmitted in a given time step in the interval $[\tau_\delta, \tau]$.

Lemma 29 *Let t be the time of a successful* **'send'** *event between τ_δ and τ, and let y be an arbitrary element of Ψ.*

$$\Pr[\ y \text{ sent at } t \mid y \text{ is eligible at } t] \geq \frac{1}{2\mathcal{S}(B_t)}.$$

Algorithm A

```
L = {};  /* Locked words */
C = {};  /* Completed words */
P = {};  /* Previously sent and currently unlocked words */
U = M;   /* Unsent (clean) words */
/* Invariant: Sets L, C, P, U are disjoint and together cover M. */

while (C ≠ M) do {
   Case 'Send' {
      choose r uniformly at random from {1, ..., S(|L|)};
      if (r ≤ |P|) {
         e = the rth smallest element from P;
         L = L ∪ {e};  P = P \ {e};
         send e;
      }
      else {
         e = the smallest element from U;
         L = L ∪ {e};  U = U \ {e};
         send e;
      }
   }
   Case 'ACK(e)' {
      C = C ∪ {e};  L = L \ {e};
   }
   Case 'NACK(e)' {
      P = P ∪ {e};  L = L \ {e};
   }
}
```

Figure 4.2: The randomized data selection policy

Proof: Fix a time t satisfying the conditions of the lemma and an element $y \in \Psi$ that is eligible at t. If y is unlocked at t, it is sent with probability $\frac{1}{\mathcal{S}(B_t)}$. Let $t_1, t_2, \ldots t_{B_t}$ be the times of the B_t most recent (unacknowledged) transmissions prior to t. The probability that y is locked at t is bounded by the sum of the probabilities that y was sent in one of those B_t most recent transmissions. Since the probability of y being chosen at t_i is $\frac{1}{\mathcal{S}(B_{t_i})}$, we have the following:

$$\Pr[y \text{ locked at } t] \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(B_{t_i})}.$$

For all i, the transmission at time t_i is not acknowledged until after time t, and hence the backlog at t_i is at least i. Since $\mathcal{S}$ is monotone, $\frac{1}{\mathcal{S}(B_{t_i})} \leq \frac{1}{\mathcal{S}(i)}$. Therefore by the definition of $\mathcal{S}$ we conclude:

$$\Pr[y \text{ locked at } t] \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(B_{t_i})} \leq \sum_{i=1}^{B_t} \frac{1}{\mathcal{S}(i)} \leq \frac{1}{2}.$$

Therefore, the probability of y being sent at t is at least $\frac{1}{2} \times \frac{1}{\mathcal{S}(B_t)}$. ■

Now consider an element $y \in \Psi$. If y is neither completed nor eligible at a successful send event in $[\tau_\delta, \tau]$, then it must be the case that some other member of Ψ (with index $< y$) is successfully delivered at that time. By our earlier bound on $|\Psi|$, this situation can occur at most $2\mathcal{S}(\widehat{B}_\tau)$ times. Since there are δ successful transmissions in $[\tau_\delta, \tau]$, y must therefore be eligible in at least $\delta - 2\mathcal{S}(\widehat{B}_\tau)$ of them, or must have been successfully delivered by τ. By applying Lemma 6, the probability that y is not successfully delivered at time t in $[\tau_\delta, \tau]$ in which it was eligible is at most $\left(1 - \frac{1}{2\mathcal{S}(B_t)}\right) \leq \left(1 - \frac{1}{2\mathcal{S}(\widehat{B}_\tau)}\right)$, so the probability that y is not delivered by time τ is bounded by

$$\left(1 - \frac{1}{2\mathcal{S}(\widehat{B}_\tau)}\right)^{\delta - 2\mathcal{S}(\widehat{B}_\tau)} \leq e^{-(2c \ln 2\mathcal{S}(\widehat{B}_\tau) - 1)}.$$

Now by taking a union bound over the at most $2\mathcal{S}(\widehat{B}_\tau)$ elements of Ψ, the probability that the algorithm does not complete the prefix through word x by time τ is bounded by $(2\mathcal{S}(\widehat{B}_\tau))^{-2c+2}$. ■

4.4.2 A Lower Bound for Randomized Data Selection Policies

We next prove a nearly matching lower bound on $\Delta(t, \theta)$ for randomized policies. Recall that the adversary is oblivious, fixing its strategy in advance of the execution of the algorithm, and does not adapt its strategy to the random choices made by the algorithm.

The simple strategy our adversary employs is to fix a value for B_θ, then admit each packet independently with probability $\frac{1}{2}$. Using this adversary, we prove the following lemma, which implies Theorem 25 directly.

Lemma 30 *For any algorithm $\mathcal{A}$,*

$$\exists\theta\ \exists t \textit{ such that } \Pr\left[\Delta^A(t,\theta) \geq \frac{B_\theta \log B_\theta}{22}\right] \geq \frac{1}{16}$$

Proof: Fix an algorithm A and a sufficiently long message size m. The adversary then chooses B_θ such that $B_\theta^2 \log^2 B_\theta = m$ and uses transcripts in which each *round* of B_θ send events is followed by an acknowledgment for the entire round. The adversary admits each packet into the network independently with probability $\frac{1}{2}$. We define a *phase* of the transcript to be a sequence of $\frac{\log B_\theta}{2}$ consecutive rounds. and we let $k = 2B_\theta \log B_\theta$ be the number of phases. If in a run of the algorithm, A transmits two or more copies of some message word in any round of phase i of the transcript, we say that A uses *redundancy* in phase i. Then we let r_i be the probability over all coin tosses of A and the adversary that A uses redundancy in phase i.

The probabilities r_i are used to select the value for t in the lemma. Intuitively, if r_i is large for all phases, the redundancy used by the algorithm will waste bandwidth when multiple copies of a message word are admitted, while if some r_i is small, the algorithm will be unlikely to succeed in having each of its transmissions contribute to $P^A(t,\theta)$ in phase i. In particular, if $r_i \geq \frac{3}{8}$ for all i then we choose t to be the time of the last send event ending phase k. Otherwise, there must exist a phase $j \in \{1 \ldots k\}$ such that $r_j < \frac{3}{8}$, and we choose t to be the time of the send event ending the first such phase j.

It is left to show that the lemma holds in both of these cases for our choice of t and θ. We make frequent use of the following fact in our analysis.

Fact 31 *Consider a random variable X which takes on values $\leq D$. Then for any α and γ, both in the interval* $(0,1)$.

$$\mathbf{E}[X] \geq \alpha D \Longrightarrow \Pr\left[X \geq \gamma(\alpha D)\right] \geq \frac{1-\gamma}{\frac{1}{\alpha}-\gamma}$$

Proof: (of Fact 31)

$$\alpha D \leq \ \mathbf{E}[X] \quad \leq \quad (1-\Pr[X \geq \alpha\gamma D])\alpha\gamma D + (\Pr[X \geq \alpha\gamma D])D$$

$$\Longrightarrow \quad 1 \leq (1 - \Pr[X \geq \alpha\gamma D])\gamma + (\Pr[X \geq \alpha\gamma D])\frac{1}{\alpha} \text{ or}$$
$$1 \quad \leq \quad \gamma + \left(\frac{1}{\alpha} - \gamma\right) \Pr[X \geq \alpha\gamma D]$$

■

Case I: $r_i \geq \frac{3}{8}$ for all $i \in \{1 \ldots k\}$.

In each phase in which A uses redundancy, the probability that two copies of the same message word are both admitted into the network is at least $\frac{1}{4}$, since each copy is admitted independently with probability $\frac{1}{2}$. We refer to such a phase as one in which the algorithm is *caught*. Each such phase increases the redundancy and hence $\Delta^A(t, \theta)$ is at least the number of such phases in $\{1 \ldots k\}$, which we denote by C. The expected value of C, $\mathbf{E}[C]$, is at least $\frac{1}{4} \times \frac{3k}{8}$, and C is bounded above by k, so by applying Fact 31 with $\alpha = \frac{3}{32}$ and $\gamma = \frac{1}{4}$:

$$\Pr\left[C \geq \frac{1}{4} \times \frac{3k}{32}\right] \geq \frac{9}{125} > \frac{1}{16}$$

Using C as a lower bound on the value of $\Delta^A(t, \theta)$, and plugging in for k yields:

$$\Pr\left[\Delta^A(t, \theta) \geq \frac{B_\theta \log B_\theta}{22}\right] \geq \frac{1}{16}$$

which completes the proof for Case I.

Case II: There is a $j \in \{1 \ldots k\}$ such that $r_j < \frac{3}{8}$.

Recall that t is set to be the time ending the first phase j in which the algorithm is unlikely (i.e., with probability $< \frac{3}{8}$) to use redundancy. To show the result, we assume first that the algorithm will not use redundancy in phase j and let the algorithm run on a transcript θ up until time s, the beginning of phase j.

Let $N_j(\theta, \rho)$ denote the event that algorithm A does not use redundancy on transcript θ with random tape ρ while executing phase j. We use the shorthand N_j when θ and ρ are understood. Let $Y = P^*(t, \theta) - P^*(s, \theta)$, the random variable which measures the available bandwidth in phase j, noting that Y does not depend on the contents of ρ. We first show that the available bandwidth in phase j must be large with constant probability, even when conditioning on the event that our algorithm chooses not to use redundancy. By our choice of the length of a phase and since each packet is admitted with probability $\frac{1}{2}$, $\mathbf{E}[Y] = \frac{B_\theta \log B_\theta}{4}$. Now, since $\mathbf{E}[Y] = \Pr[N_j]\ \mathbf{E}[Y|N_j] + (1 - \Pr[N_j])\ \mathbf{E}[Y|\bar{N}_j]$, and Y (and therefore $\mathbf{E}[Y|\bar{N}_j])$ is bounded above by $\frac{B_\theta \log B_\theta}{2}$, we have:

$$\begin{aligned}
\mathbf{E}[Y|N_j] &\geq \frac{1}{\Pr[N_j]}\left(\mathbf{E}[Y] - (1-\Pr[N_j])\ \mathbf{E}[Y|\bar{N}_j]\right) \\
&\geq \frac{B_\theta \log B_\theta}{4\Pr[N_j]} - \frac{(1-\Pr[N_j])B_\theta \log B_\theta}{2\Pr[N_j]} \\
&\geq \frac{B_\theta \log B_\theta}{4}\left(\frac{1-2(1-\Pr[N_j])}{\Pr[N_j]}\right)
\end{aligned}$$

$$\geq \frac{B_\theta \log B_\theta}{4}\left(2 - \frac{1}{\Pr[N_j]}\right) \geq \frac{B_\theta \log B_\theta}{10} \tag{4.1}$$

The next step is to show that any algorithm which does not use redundancy in phase j makes far less progress than the optimal algorithm with very high probability. For an algorithm G let $\overline{P}^G(t,\theta) = \max\{P^G(t,\theta), P^*(s,\theta)\}$. We will show that

$$\max_G\ \mathbf{E}[\overline{P}^G(t,\theta) - P^*(s,\theta)] < B_\theta$$

where the max is taken over algorithms G which do not use redundancy in phase j. Let C denote the algorithm achieving the maximum above. Now consider a word y not admitted prior to the beginning of the phase by C. By the fact that C does not use redundancy, y is transmitted at most $\frac{\log B_\theta}{2}$ times in round j. And since the adversary chooses whether to admit each transmission independently with probability $\frac{1}{2}$, the probability that y is admitted in the phase is $x = 1-(\frac{1}{2})^{\log B_\theta/2} = 1 - \frac{1}{\sqrt{B_\theta}}$. Therefore,

$$\mathbf{E}[\overline{P}^C(t,\theta) - P^*(s,\theta)] = \sum_{i=0}^{\infty} ix^i$$

$$= \frac{x}{(1-x)^2} = B_\theta\left(1 - \frac{1}{\sqrt{B_\theta}}\right) < B_\theta.$$

It follows from the fact that C achieves the maximum expectation that

$$\begin{aligned}
&\mathbf{E}[\overline{P}^A(t,\theta) - P^*(s,\theta)|N_j] \\
&\leq \frac{1}{\Pr[N_j]}\ \mathbf{E}[\overline{P}^C(t,\theta) - P^*(s,\theta)] \ < \ \frac{8B_\theta}{5}.
\end{aligned} \tag{4.2}$$

Subtracting (4.2) from (4.1) we get:

$$\mathbf{E}[P^*(t,\theta) - \overline{P}^A(t,\theta)|N_j] > \frac{B_\theta \log B_\theta}{10} - \frac{8B_\theta}{5} \geq \frac{B_\theta \log B_\theta}{11}.$$

Since $P^*(t,\theta) - \overline{P}^A(t,\theta) \le \frac{B_\theta \log B_\theta}{2}$, we may use Fact 31 with $\alpha = \frac{2}{11}$ and $\gamma = \frac{1}{2}$ to obtain

$$\begin{aligned} \Pr\left[P^*(t,\theta) - P^A(t,\theta) \ge \frac{B_\theta \log B_\theta}{22} | N_j\right] &\ge \\ \Pr\left[P^*(t,\theta) - \overline{P}^A(t,\theta) \ge \frac{B_\theta \log B_\theta}{22} | N_j\right] &\ge \frac{1}{10}. \end{aligned}$$

Hence,

$$\Pr\left[P^*(t,\theta) - P^A(t,\theta) \ge \frac{B_\theta \log B_\theta}{22}\right] \ge \frac{5}{8} \cdot \frac{1}{10} > \frac{1}{16}.$$

■

4.5 Empirical results

We tested the performance of our data selection policies by simulating their behavior under various conditions. The bulk of the testing was done on traces of MBone audio broadcasts, each lasting approximately one hour, broadcasting packets of size 2.5Kb at a constant rate of one packet every $40ms$, taken from [YKT 96]. Each broadcast was received by between ten and sixteen receivers from the U.S. and Europe, and each receiver recorded the set of packets which it received. Although timing information was not recorded in the traces, in our subsequent experiments, we estimated a typical RTT to the most distant receiver of about 1200 ms. This estimate was then used to simulate a mean *backlog* of approximately 30 packets in our experiments. Although the variance of round-trip times has empirically been shown to be quite large, especially in periods of high congestion, we did not have sufficient information to attempt to simulate this effect. We conjecture that performance of polices without worst-case performance guarantees would perform no better if this effect were present. Additional tests were done on loss patterns we generated ourselves as described below.

We tested the performance of the randomized data selection policy given in Section 4 against the simple deterministic strategy that selects (in our terminology) the smallest *available* word for transmission at each time step. Figure 4.3 shows a head-to-head comparison on a representative portion of an MBone audio trace. As can be seen in Figure 4.3, our algorithm performs slightly worse than the deterministic algorithm, which performed

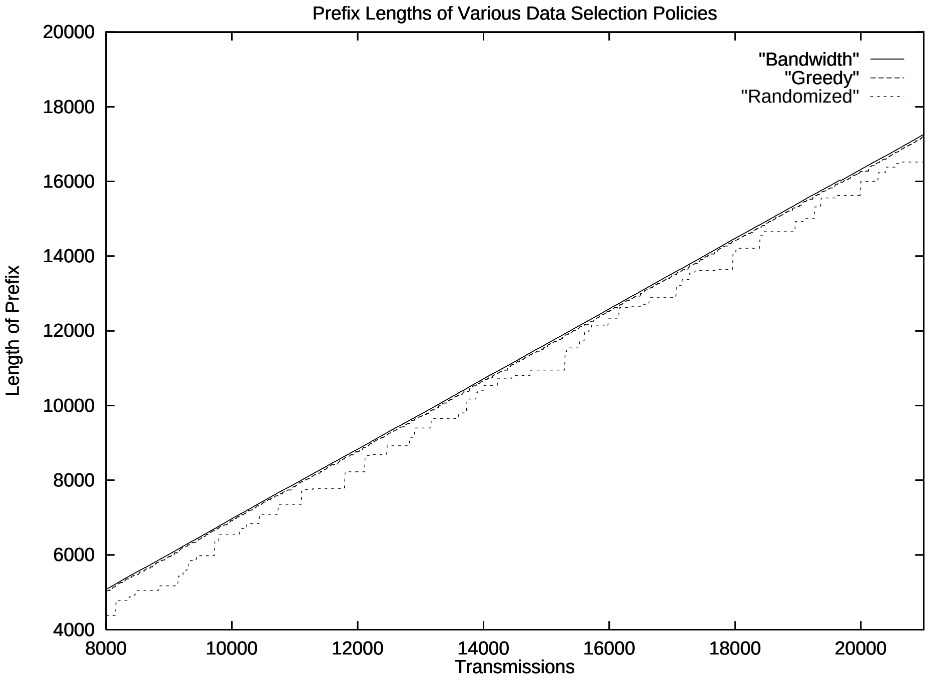

Figure 4.3: Comparative performance on an Internet trace.

almost indistinguishably close to optimal in this trace. In general, the randomized nature of our algorithm after an initial packet loss causes a performance degradation in terms of the prefix measure, since the message words selected often do not contribute to the prefix for a short time. However, this performance degradation is bounded in the expectation by a function of the backlog which remains fixed throughout this MBone session.

We also tested the performance of these policies under simulated data which we generated, where periodic burst loss was present. To generate these traces, we modeled a burst as an on/off process in which the burst length and burst interarrival times were the random variables. For a given burst, with high probability $p = .9$, we let the parameters of the burst remain identical to that of the previous burst. With the remaining probability, values for both the burst length and interarrival time were re-chosen uniformly at random. One can clearly see in Figure 4.4 that in this scenario, the performance of our randomized algorithm appears virtually identical to that obtained in the MBone trace, whereas the greedy algorithm occasionally suffers from dramatic drops in performance, particularly around transmissions numbered in the 18000's. If we were to correlate the burst interarrival time exactly with the round-trip time, we arrive at a situation in which the greedy

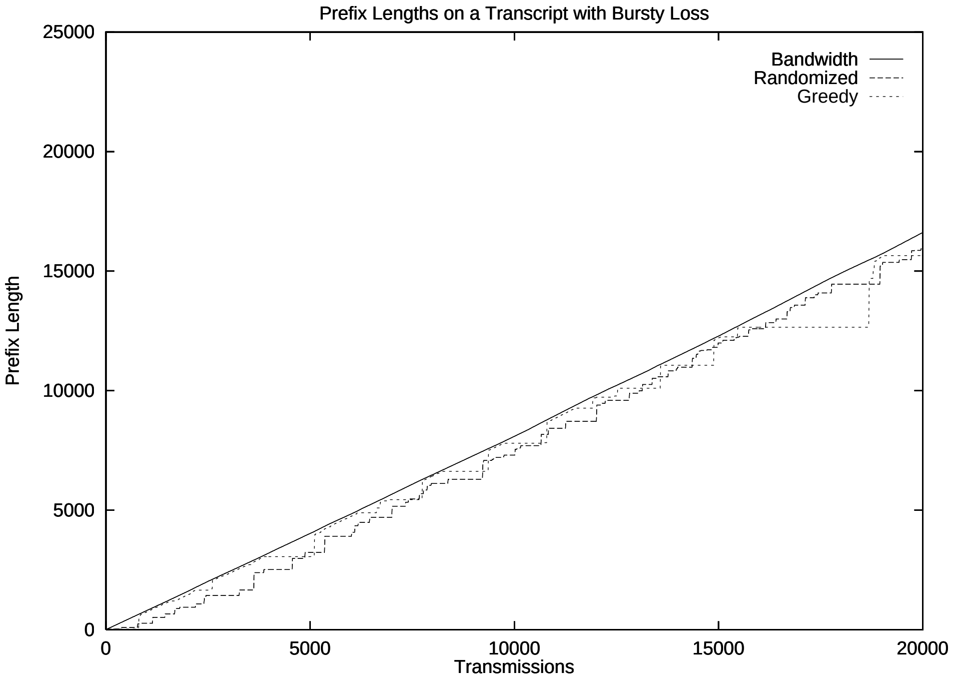

Figure 4.4: Comparative performance on bursty loss patterns.

algorithm fails to deliver a word of the message for an unbounded amount of time.

The slightly inferior performance of our algorithm under the typical network behavior as indicated by the traces can be viewed as a premium one is paying so that in the event of worst-case behavior, performance does not drop dramatically. A reasonable question to ask is whether the bad behavior sketched in our trace data really occurs in practice? Evidence from empirical studies indicate that loss patterns in which packets are lost at fixed, periodic time intervals have recently been observed in the Internet [FJ 94, YKT 96]. Therefore it is likely that sometimes, under existing network conditions, the simple deterministic strategy performs poorly.

The exact performance of our algorithm depends directly on the size of the set of candidates which we draw from at random at each transmission time. This size is determined by the constant d in the definition of the function $\mathcal{S}(x)$. Indeed, if we increase d beyond the point where $\sum_i \frac{1}{\mathcal{S}(i)} \leq \frac{1}{2}$, the condition required to carry out the analysis, simulations on what we believe to be worst-case examples indicate that the performance of the algorithm continues to improve until d is substantially beyond this threshold. We conjecture that our analysis is not tight, and therefore, in practical settings one could and should fine-tune the behavior of the algorithm by setting d appropriately, perhaps beyond the threshold where performance is provably good.

4.6 Discussion

Our objective in this work was to provide a theoretical analysis of TCP-like protocols in a model which strives to make as few assumptions about the network as possible. The model we chose enables an adversary to regulate the speed at which a protocol may transmit, the pattern with which the transmitted packets are lost, and the latency with which feedback about transmissions propagates back to the sender. We then measure our algorithm's performance on this transcript in comparison to the theoretically optimum performance, to determine the algorithm's regret on the worst case transcript.

One of our main findings is that there is substantial benefit to be derived from decoupling a TCP-style transmission protocol into component policies; in particular, separating out the decision of what to transmit from other policies, such as congestion control. Our main result develops a universal data selection policy, which has a near-optimum performance guarantee on *every* transcript, i.e. for any specification of the behavior of the network.

Chapter 5

Policies for Multicast Applications

We now extend the models set forth in Chapters 3 and 4 to handle bulk transmission point-to-multipoint, or *multicast* connections. Multicast is a technique used for scalable transmission of a message or data stream to a set of receivers in distributed networks. Whereas broadcast typically transmits data to all endpoints in the network, multicast transmits the data only to a subset of interested endpoints. The IP Multicast protocol, currently deployed on a multicast-enabled subnetwork of the Internet known as the MBone, is a representative implementation of this technique. In this protocol, senders establish an identifying IP multicast address to which they transmit packets. Receivers who are interested in participating *register* with the associated multicast group. They do so by transmitting a join message with the identifying multicast address along a path to the source. Any intermediate router which receives this join message now begins forwarding messages from this multicast session along the link on which it received the join message if it is not already doing so. In this fashion, a *multicast tree* rooted at the source is established, where leaves of the tree correspond to receivers in the multicast group and internal nodes of the tree correspond to routers who are actively forwarding packets of this multicast session to their children in the tree.

One important consideration in the design of multicast protocols is that the work performed by the sender does not grow with the size of the group, i.e. the number of receivers. For example, whereas unicast transmission protocols allow the receiver to acknowledge received transmissions, it is unscalable and infeasible for each receiver in a multicast group to acknowledge each received packet, as this can overwhelm the source. Addressing the potential of this feedback implosion problem and related scalability issues is a primary concern in the design of multicast protocols.

For many existing reliable multicast applications, the network uses best-effort delivery service for each packet initiated by the source, i.e. the routers attempt to forward *every* transmitted packet to *all* destinations in the multicast tree. But for some high-bandwidth streaming applications, notably packet video, the presence of low-bandwidth links or network congestion can limit the ability of certain endpoints to receive a high-fidelity transmission. Rather than have the sender transmit at a low rate which even the most constrained receiver can tolerate, a recent approach is the use of *layering*. In layered multicast, the sender transmits at a high data rate, comprised of many layers, and each of the heterogeneous receivers subscribe to one or more layers of this transmission. Therefore, different receivers in the same multicast tree may simultaneously be associated with different incoming transmission rates. The main prerequisite for implementing layered multicast is an encoding technique which enables receivers to recover lower-fidelity versions of the transmissions given their subscription to a subset of the layers.

In this chapter, we describe how to extend the techniques and results of Chapters 3 and 4 to apply in the realm of multicast connections. A prerequisite for many of our results is the existence of efficient forward error correcting codes, which we describe in Section 5.1. The next step is to consider how to incorporate multicast connections into our framework of establishing transmission rates so as to maximize the throughput of the system. In Section 5.2, we present a natural extension to the model of Chapter 3 to enable the network to maximize throughput for unicast connections and unlayered multicast connections. More challenging is the problem of also handling layered multicast connections, and we outline a recent result of Awerbuch, Azar and Bartal which maps that problem into our positive linear programming framework and solves it in our distributed communication model.

We then turn to the question of developing data selection policies for multicast connections in Section 5.3. One main result we present is a proof that there is an inherent performance gap between algorithms which use *encoding* schemes such as forward error correction (FEC) and those which do not, i.e. only transmit words of the original message. In particular, we show that algorithms which do not use encoding also do not scale, as the competitive ratio of these algorithms necessarily grows with the number of receivers. We then present a centralized multicast data selection policy which uses FEC and demonstrate its performance on simulations and multicast trace data taken from the MBone by Yajnik et al [YKT 96] in Section 5.3.4. These empirical results indicate that our algorithm outperforms natural centralized algorithms which do not use FEC.

5.1 Forward Error Correction

Systematic forward error correcting codes (FEC codes) enable network sources to transmit original data along with additional redundant data which can be used to reconstruct the original data at a receiver in the event that some of the original data is lost en route.

5.1.1 Theory

Forward error correction coding techniques strive to realize the following ideal behavior. For any arbitrary values of k, h and L, given a set of k message words $M = \{m_1, m_2, \ldots m_k\}$, each of fixed length L, efficiently construct a set of h redundant words $R = \{r_1, r_2, \ldots r_h\}$ also of length L such that M can be efficiently reconstructed from any subset $Y \subseteq M \bigcup R$ for which $|Y| \geq k$.

Reed-Solomon Erasure correcting codes (RSE codes) as described in [MS 77] realize the ideal behavior above efficiently for small values of k and h as follows. Let $m_1, \ldots m_k$ be elements from the Galois field $GF(2^L)$ and define the polynomial $F(x)$ on $GF(2^L)$ as:

$$F(x) = m_1 x^{k-1} + m_2 x^{k-2} + \ldots + m_{k-1} x + m_k$$

The redundant words r_i are taken to be evaluations of this polynomial: $r_i = F(g^{i-1})$, where g is a generator of $GF(2^L)$. Now, given any subset of size k of $M \bigcup R$, it is possible to uniquely interpolate $F(x)$, thereby reconstructing M. The main drawback of these codes is the overhead in performing the encode and decode operations. Letting $n = k + h$, encoding takes time $O(n \log n)$, and although the same asymptotic decoding time has been established in theory, the most efficient decoding operation in practice takes quadratic time. For the networking applications we consider, this quadratic dependence on n dramatically limits throughput at receivers even for settings of k and h as small as 50 and 10, respectively.

Much more efficient encoding and decoding algorithms have recently been realized which slightly relax the ideal behavior described above. The Tornado codes of Luby et al [LMSSS 97] generate a set of redundant words R such that M can be reconstructed from any subset of $M \bigcup R$ of size at least $k(1 + \epsilon)$ with high probability. Moreover, the encoding and decoding time is $O(n \ln(1/\epsilon))$, where $n = k + h$, as defined above. Their codes are based on a cascade of bipartite graphs, constructed as follows: Consider a bipartite graph G with l left nodes, each storing a message word, and βl right nodes which will each store redundant words. A redundant word stored at a right node is computed by taking the XOR of the values of words stored in its neighbors on the left hand side of the graph. The bipartite graphs used are chosen at random from bipartite graphs having a carefully chosen degree sequence such that the *edges* witness specific degree distributions on the left and right sides. One property of these degree sequences is that the resulting graphs are sparse, with only a linear number of edges in all. The other key property of the choice of degree sequences is that if all βl redundant words are recovered, the message can be recovered with high probability provided that at most $(1 - \epsilon)\beta l$ of the message words have been lost. Now, to build a code which can tolerate h erasures in transmitting a message of k words, bipartite graphs with these properties are cascaded together recursively and capped by a small conventional code, where the decoding process initiates. Provided that the conventional code can reconstruct the right side of the rightmost bipartite graph, and a sufficiently small fraction of the remaining words are lost, then with high probability, all words stored in the cascaded bipartite graphs can be reconstructed. The encode and decode operations for these codes are extremely efficient in practice, and the code delivers high performance guarantees as k grows large, unlike the operations for RSE codes.

5.1.2 Benefits of Using FEC

Integrating forward error correction with reliable multicast realizes the following benefits. Different receivers can recover different lost message words with the same redundant data, which helps to minimize the number of unnecessary retransmissions and receptions. Feedback from receivers can be implemented at a coarser granularity, in that receivers need only acknowledge *how many* packets they have received from a given message block, and not which particular packets they have received. This also simplifies the scheduling process at the sender, in that at any point in time, the sender need only decide whether or not to transmit more redundant packets for a given block of message data.

Clearly, the use of forward error correction is ideal for unlayered reliable bulk transmissions in which all receivers have sufficient connectivity to receive data at or near the source transmission rate. But forward error correcting codes can also be used to facilitate reliable bulk multicast transmission to a set of receivers with heterogeneous maximum receive rates using ideas from the layered multicast approach. For this application, the source of the transmission treats the entire bulk message as one large block and generates a large set of redundant codewords for this block. In practice, this technique can presently be realized only by using the codes of [LMSSS 97] described above, which can efficiently perform encoding and decoding even when the block size and redundancy are very large, although a similar approach is advocated with RSE codes by [RV 97]. The source then multicasts message words and codewords, and intermediate routers forward these packets at the maximum of the rates demanded by receivers in their subtree. As a result, receivers which receive packets at lower rates must participate in the multicast session for a longer period of time than receivers with higher rates in order for them to reconstruct the entire bulk message.

5.2 Flow Control for Multicast Connections

In our flow control model, a multicast connection is represented as a *multicast tree*, spanning a subset of the edges in the network. Each receiver corresponds to a leaf in the tree, and routers reside at the internal nodes of the tree. As in the case of unicast connections, flow control is the problem of regulating the end-to-end transmission rates of these connections. As multicast deployment is only just underway, algorithms for multicast flow control have not yet received the wide attention paid to rate-based flow control for unicast connections. In this section, we consider two objective functions for multicast flow control, both relating to throughput maximization. The first objective assumes unlayered multicast in which a single rate is assigned to each multicast tree, so that each receiver in the tree requests a rate equal to the source transmission rate. In the communication model we describe, for this objective function, we can apply the techniques and algorithms of Chapter 3 to obtain a solution which is a $(1+\epsilon)$ approximation to the optimum in a polylogarithmic number of distributed rounds.

A related objective function is motivated by recent work on layered multicast. We have already described how forward error correction can be used to implement layered multicast for bulk transfer applications. A related approach for implementing reliable, bulk multicast with layering is described in [Vic 97]. Other work, notably that of McCanne [McC 96], is deployed for use with real-time video applications on the MBone. Their work uses encoding technology which is capable of encoding a video stream in such a way that certain substreams of the original stream are lower fidelity representations of the original stream. The simplicity of the operation needed to select these substreams makes it feasible for deployment at network routers, and is currently used in applications on the MBone. Of course, the nature of this technique implies that receivers are able to tolerate degraded transmission quality, so it is not suitable for reliable transmissions. These approaches for layered multicast motivate an objective for flow control in which the objective is to maximize the aggregate rates delivered to the receivers of all multicast trees, where receivers in a given tree may receive different rates. Using these techniques, the relaxed constraint on the flows on each of the trees is that an internal node of the tree can deliver an outgoing rate no greater than its incoming rate.

5.2.1 The Communication Model for Bandwidth Allocation

As in the unicast case, the source of the multicast transmission will communicate with intermediate routers involved in transmitting data for the connection to arrive at a feasible rate allocation. In the unicast case, in one distributed round, the sender was able to broadcast or receive a fixed-size message from the routers which it utilized by the use of control messages which looped through the source-destination path of the connection. In the multicast setting, we also employ control messages, but must use them in a scalable manner.. On the outbound routes, i.e. on root to leaf paths, control messages are routed in the same way as data messages, in that intermediate routers at branch points of the multicast tree copy the message and route it along all outgoing links of the tree. On the return paths, we assume reverse path forwarding, whereby the control messages return along edges of the multicast tree. To avoid unscalable feedback implosion, in which one outbound message triggers a set of responses of size proportional to the number of receivers, we assume that the routers at branch points of the tree forward only *one* message to their parent on the reverse path. A distributed round in this framework is defined as the time to transmit a control message and receive a response from the receivers. In this communication model, the sender can broadcast a message to the nodes of the tree, or receive a fixed-size message aggregated from the nodes of the tree, in an analogous manner to the unicast setting.

5.2.2 Formulation of the Multicast Allocation Problems

The first objective we discuss arises in the case of reliable, unlayered multicast in which a single, fixed rate is assigned to each of the multicast trees in the network. In this setting, we wish to assign rates y_j to each of the trees j such that the capacity constraints of the underlying routers and network links are not violated. If our objective is to maximize the throughput of the network measured as the rates received by the receivers, this can be modeled by maximizing the weighted sum of the transmission rates, where the weights are the sizes of the sets of receivers. In this specification, larger multicast groups accrue more benefit than smaller groups, although the weights can be adjusted to realize alternative allocations. In any case, our problem may be expressed by the familiar positive linear program:

$$\begin{aligned} \max \sum_{j=1}^{n} S_j y_j \text{ s.t.} \\ \forall i, \sum_{j=1}^{n} a_{ij} y_j \leq C_i \\ \forall i, j,\ a_{ij} = 0 \text{ or } 1 \\ \forall j,\ y_j \geq 0 \end{aligned}$$

The constant S_j denotes the benefit associated with multicast tree j and the variables a_{ij} are 0/1 indicator variables denoting whether router i is a node of multicast tree j. This program is identical to the general formulation for unicast; moreover, the distributed algorithms developed to solve that problem may be employed in the communication model for multicast which we present here. The main difference is that a multicast source must now compute the sum of the weights of all routers in the tree to assess whether to increase its rate. The manner in which control messages propagate through multicast trees facilitates the implementation of this operation.

A more challenging problem arises in layered multicast applications. We then require the following natural property of the rates allocated to each multicast tree:

No node in the tree may be allocated a higher rate than its parent.

As always, we will require that the capacity constraints of each router are not violated, and we define a throughput maximizing allocation as one in which the sum of the receive rates is maximized. Unfortunately, this new set of constraints on the rates at internal nodes of the tree adds negative entries to the linear programming constraint matrix, meaning that the straightforward formulation of the progam is no longer positive. But an alternative formulation of this optimization problem can be expressed as a positive linear program, albeit with a number of variables for each multicast connection which is exponential in the size of that multicast group.

This formulation expresses the flow on a multicast tree as an aggregation of subflows on all minimal subtrees of the multicast tree rooted at the source and containing a non-empty subset of the receivers. For a multicast tree i with a set of receivers R_i, let $S_{i,1}, \ldots, S_{i,2^{|R_i|}-1}$ denote these minimal subtrees. Then let y_{ij} denote the flow of commodity i (data multicast along multicast tree i) on subtree S_{ij}. Also, let $B(S_{ij})$ denote the aggregate benefit derived from receivers for each unit of flow on subtree S_{ij}, $B(S_{ij}) = \sum_{r \in S_{ij}} B_{ir}$. Then we can express the objective as the following positive linear program:

$$\begin{aligned} \max \sum_i \sum_{j=1}^{2^{|R_i|-1}} y_{ij} B(S_{ij}) & \qquad \text{subject to} \\ \forall e : \sum_i \sum_{j \mid e \in S_{ij}} y_{ij} & \qquad \leq c(e) \\ \forall i, j : y_{ij} & \qquad \geq 0 \end{aligned}$$

The benefit received by a given receiver is summed over the flows it receives from all subtrees for which it is a leaf. Capacity constraints are enforced by bounding the sum of rates on all subtrees which utilize a given router (or edge) e. This program is now a positive linear program in the form for which we have fast approximate solutions, but with a potentially intractable number of variables for each multicast connection to handle.

Recently, Awerbuch, Azar and Bartal [AAB 97] made observations that make this problem tractable in the distributed model we present, enabling the use of an algorithm derived from the work in Chapter 3 for solving this layered multicast allocation problem. They show that at each iteration, flow can be increased on a single subtree of each multicast tree in a distributed fashion such that the feasibility and performance guarantees proved for the primal-dual approach of Chapter 3 hold. Moreover, these *max-dense* subtrees can be identified quickly in our distributed model. Therefore, it is not necessary for connections to explicitly compute flow values for each of its subtrees; it suffices for each multicast connection to distributively increase the flow on a single subtree of the multicast tree at each iteration. The (non-unique) subtree on which to pump flow is a max-dense subtree S_{ij}

satisfying the following two criteria:

$$\begin{aligned} \sum_{e \in S_{ij}} \frac{x_e}{c(e)} &< B(S_{ij}) && \text{[Density]} \\ \forall k, S_{ij} \subseteq S_{ik}, \sum_{e \in S_{ik} - S_{ij}} \frac{x_e}{c(e)} &\geq B(S_{ik}) - B(S_{ij}) && \text{[Maximality]} \end{aligned}$$

Their work also describes how to identify such a max-dense subtree, if one exists, for each multicast tree in a single distributed round of the model we present. Given these definitions, the algorithm of Chapter 3 can then be applied, where the only modification is that each iteration, rather than pumping flow on *connections* which witness dual infeasibility, the distributed algorithm pumps flow on a max-dense subtree of each multicast tree. They prove that this algorithm still achieves a $(1+\epsilon)$ ratio in a polylogarithmic number of rounds in the distributed model in spite of the fact that the positive linear program has exponential size.

The results outlined in this section can be expressed as the following theorem:

Theorem 32 *For* $0 < \epsilon \leq 1$ *and* $0 < r \leq \ln(\gamma m)$*, there is a distributed algorithm which produces a feasible* $(r + (1+\epsilon)^2)$*-approximation to the optimum rate allocation for unicast connections, unlayered mulicast connections, and layered multicast connections and runs in* $O\left(\frac{\ln^2(\gamma m)\ln(\gamma mn/\epsilon)}{r\epsilon^2}\right)$ *distributed communication rounds.*

5.3 Multicast Data Selection Policies

The problem of efficiently and reliably disseminating bulk data from a single sender to a group of many recipients is a fundamental problem underlying important networking applications. In a network where there are no other users and data is never lost, the main obstacle to solving this problem is again scalability, i.e. it is essential that the sender is not required to perform an amount of work that grows with the number of receivers. Unfortunately, our current network infrastructure is neither lossless nor at the sender's personal disposal, which poses the following difficulties for *reliable* bulk multicast data selection policies:

- Data lost by one or more of the receivers must be retransmitted.
- Additional communication (such as retransmission and acknowledgments) between the sender and the receivers may not scale well.

- Multicast transmission rates may fluctuate over time.

As mentioned earlier, data selection policies for unicast have not been studied extensively, whereas the data selection policy in the multicast setting has generated much more widespread discussion. A promising technique which has already been implemented in many lower level applications is error detection and correction. Recently, the use of forward error correction has been proposed to improve reliability in the higher level setting of multicast transmissions. Nonnenmacher, Biersack, and Towsley [NBT 97] present multicasting algorithms that use FEC and provide performance analysis of these protocols under a wide variety of probabilistic packet loss scenarios where the probability of packet loss is low (typically .01). Other approaches for using FEC for reliable multicasting can be found in [Mos 84], [Hui 96] and [Riz 97]. In spite of the recent focus of attention on this area, there is a distinct lack of analytic evidence to support the benefits of this approach over others in spite of the empirical evidence. Our work on multicast data selection policies provides some of this analytic support.

5.3.1 Modifications to the Theoretical Model

As in the unicast case, we assume that the sender receives feedback about the progress of its transmissions based on receiver-driven acknowledgments, i.e. each packet arrival at each receiver triggers an acknowledgment. Of course, if all acknowledgments propagate back to the sender in a large multicast transmission, this acknowledgment style causes feedback implosion, in which the number of acknowledgments per packet transmission does not remain constant as the number of receivers grows, so our model is defined in a way which addresses this potential problem. However, for the purpose of comparing algorithms which use coding against those which do not, we can still prove a performance gap even when we allow the latter algorithms to use this unscalable style of acknowledgment.

One widely used technique for improving scalability in multicast is *random sampling*, whereby each packet arrival triggers an acknowledgment from only a small fraction of the receivers. While this technique has been used effectively for small sets of receivers [YM 93], it does not appear to scale well. The technique which we employ instead is *acknowledgment aggregation*, using a scheme analogous to the aggregation of control messages in our policy for multicast bandwidth allocation. In this scheme, each receiver reports an acknowledgment value for each source transmission to its parent in the multicast tree. In the event that a receiver can infer that a packet has been lost, this value may represent a negative acknowledgment. As acknowledgment values propagate up the multicast tree using reverse path forwarding, internal nodes may compute a function of the acknowledgment values transmitted by their children, and transmit this value to their parent, thereby aggregating acknowledgment values. An important component of the data selection policies we develop is to define *what* value the receivers transmit as an acknowledgment for each transmission. When this scalable aggregation process is used, we assume a synchronized model in which acknowledgments flow up the multicast tree in order, and with bounded latency. While this assumption is somewhat restrictive, the algorithm we develop with the best performance guarantee (using FEC) will not depend on this fine-grained, synchronous acknowledgment aggregation process. Using this multicast acknowledgment strategy, we can use much of the same terminology that was used in Chapter 4, in particular, each packet i is associated with a transmission time s_i, an acknowledgment time a_i, and a boolean acknowledgment value b_i, which the sender learns at time a_i.

Expressing The Prefix Measure

In the unicast environment, recall that we measured the performance of a data selection policy based on the longest intact prefix of the message received by the receiver at each point in time. In the multicast environment, we further distinguish between the *minimum prefix measure*, which measures the smallest maximal prefix of all active receivers and the *average prefix measure*, which measures the average maximal prefix, averaging over all receivers. Our analysis again compares the worst case ratio in the minimum or average prefix delivered by our data selection policy with that delivered by an optimal policy, where the worst case is taken over all points in time during the transmission and over all possible specifications of the network behavior.

The formulation of the multicast data selection problem is as follows. The sender has a bulk message which is logically decomposed into M words $w_1, w_2, \ldots, w_M$, each of which fit into a packet of fixed size. As time elapses, a sequence of events $\rho_1, \rho_2, \ldots, \rho_N$ occurs at the sender. These events either occur at transmission times s_i, and request the sender to send a word of the message, or occur at acknowledgment times a_i, informing the sender of the value b_i. At each of the events which request that a word be transmitted, the sender must select which word to transmit at that point in time. Since we eventually describe algorithms which use FEC codes, the transmitted word may either be a message word, or a redundant codeword generated for a fixed block of message words.

For a multicast data selection policy A, we let $P^A_{\text{avg}}(t, \Theta)$ denote the average prefix length obtained by the set of receivers at time t on transcript Θ and we let $P^*_{\text{avg}}(t, \Theta)$ denote the average throughput over all receivers through time t. Since the competitive difference for any multicast data selection policy grows without bound, in this chapter we use the competitive ratio as our performance measure, using the following definition from Chapter 4.

Definition 33 *The competitive ratio of a randomized multicast data selection policy A on a transcript θ at time t with respect to the average prefix measure is :*

$$R^A(t, \theta) = \frac{P^*_{avg}(t, \theta)}{\mathbf{E}[P^A_{avg}(t, \theta)]},$$

where the expectation is taken over the coin tosses of the algorithm.

We use analogous definitions for the minimum prefix measure.

Although the algorithms we develop applies in the general adversarial model presented in Chapter 4, the competitive analysis holds under a restricted model, in which the power of the adversary is bounded. In particular, when a multicast group has more than a constant number of receivers, one can easily show that no on-line algorithm can perform competitively with respect to the prefix measures when the adversary fails to deliver even a constant fraction of the transmissions to each receiver. Therefore, we measure our algorithm on transcripts which maintain the following minimal performance, or quality of service, guarantee:

In each period of W consecutive transmissions, at least a constant fraction cW of the transmissions arrive at each receiver.

In practice, on existing networks, it is often possible to choose the parameters c and W such that one can have a high degree of certainty that the guarantee will be met for virtually all receivers. We say that an adversary which generates transcripts that adhere to this quality of service guarantee for given values of c and W is a (c, W) bounded adversary.

Overview of Our Results

In Section 5.3.2, we analyze the performance of a simple randomized multicast data selection policy and show that it has a logarithmic competitive ratio. We then prove a nearly matching lower bound, showing that any data selection policy which does not use encoding *cannot* deliver a prefix within a logarithmic factor from optimal on worst-case inputs. In Section 5.3.3, we present a description of techniques for forward error correction (FEC) and our data selection policy for multicast bulk transmission which employs FEC. In the same section, we prove that our FEC multicast data selection policy delivers a prefix that is within a multiplicative constant factor of optimal at all points during the transmission. In Section 5.3.4, we bolster our theoretical arguments in favor of our algorithms with empirical simulations and network traces. In the multicast setting, the FEC approach is comparable to algorithms which do not use encoding when the number of receivers is very small, but becomes markedly superior as the number of receivers grows beyond ten, and exhibits much better scalability. Finally, we conclude with a discussion of the advantages and limitations of our study, along with directions for future work.

5.3.2 Algorithms and Lower Bounds

We first consider the performance of multicast data selection policies which do not use encoding. The algorithm which we present runs in the (c, W) bounded adversary model, and therefore its performance is parameterized by c and W. The transcript on which the proof of the lower bound is based is one which can be generated by a (c, W) bounded adversary as well. To give the algorithm every possible advantage in proving the lower bound, we show that the lower bound holds even in an unscalable full information acknowledgment model. For a reliable multicast connection with N receivers, we prove the following:

Theorem 34 *There is a randomized algorithm which transmits only message words (i.e. does not use encoding) and which has competitive ratio (with respect to either prefix measure) of:*

$$O\left(\frac{\log NW}{\log\left(\frac{1}{1-c}\right)}\right).$$

Furthermore, no algorithm which transmits only message words can achieve a better worst-case competitive ratio for either prefix measure than:

$$\Omega\left(\frac{\log(\min(N,W))}{\log\left(\frac{1}{c}\right)}\right).$$

Proof: We begin with the presentation and analysis of a randomized algorithm which proves the first half of the theorem. The randomized algorithm we develop runs in a sequence of rounds, each consisting of W transmissions. We maintain the following invariant:

All receivers have a prefix of length at least iW after $\left(\frac{i\log NW}{\log\left(\frac{1}{c}\right)}\right)$ expected rounds.

By the bound on the number of transmissions in each round, no algorithm could deliver a prefix of length larger than iW to any receiver by the end of round i. Thus, the competitive ratio we achieve is $\mathrm{O}\left(\frac{\log(NW)}{\log\left(\frac{1}{c}\right)}\right)$. The randomized algorithm maintains the invariant by repeatedly transmitting random permutations of the message words with indices in the interval $(iW,\ldots,(i+1)W)$ in rounds of length W. Rounds of this form continue until all receivers have received all message words in that interval.

The sender may determine whether this has been accomplished by use of the following synchronous acknowledgment aggregation strategy in accordance with the earlier specification. For each message word k in a given transmission interval, each internal node of the tree stores a boolean value for each of its children which is set to 1 if and only if every receiver in that child's subtree has received message word k. All of these values are initialized to zero. When a packet containing word j arrives at a leaf, that receiver acknowledges successful receipt of word j by sending a 1 to its parent. When a packet fails to arrive at a leaf, that leaf transmits a negative acknowledgment "?" to its parent. By the randomized nature of the algorithm, that leaf does not necessarily know *which* message word it is negatively acknowledging. In fact, due to retransmissions, it may be negatively acknowledging a packet containing a word which it has successfully received! Therefore, each internal node must perform the following steps before transmitting an acknowledgment value to its parent:

- If the node did not receive the packet, transmit a "?" to the parent.
- Otherwise, for each child which transmitted a 1, set that child's variable to 1.
- Then, compute the logical AND of the children's variables, and transmit that value to the parent.

By this strategy, the root of the tree receives an acknowledgment value of 0 if and only if there is a receiver who did not receive the word transmitted in the acknowledged packet.

Phase i consists of the set of rounds in which message words in the interval $(iW, \ldots, (i+1)W)$ are transmitted. We now bound the expected number of rounds in a phase. For a receiver j and a message word k, let the binary indicator variable A_{jkl} be 1 in the event that j has successfully received k by the end of round ℓ of a phase and 0 otherwise. For all j, k, l: $\Pr[A_{jk\ell} = 1 | A_{jk,\ell-1} = 0] = c$, by the bounded adversary and the random permutation of the set of transmitted words. There are a total of NW (receiver, message word) pairs of interest in each phase. Therefore, by the linearity of expectations, after any round ℓ, $\mathbf{E}[\sum_{jk} A_{jk\ell}] = NW(1 - c^\ell)$. To bound the number of rounds per phase in the expectation, we compute ℓ for which this quantity falls below 1:

$$c^\ell \geq 1 - \frac{1}{NW} \Rightarrow \ell > \frac{\log\left(1 - \frac{1}{NW}\right)}{\log c}$$

The value for ℓ which achieves this objective is:

$$\ell = \mathrm{O}\left(\log_{\frac{1}{1-c}}(NW)\right) = \mathrm{O}\left(\frac{\log NW}{\log \frac{1}{1-c}}\right)$$

The bound on the competitive ratio follows directly. ■

Without Encoding, Performance Cannot Scale

Now we prove the second half of Theorem 34, demonstrating that any multicast data selection policy which does not use encoding has a logarithmic competitive ratio.

Proof: (of Theorem)

Define a *loss pattern* to be a subset of ck packets which are lost in a sequence of k consecutive transmissions, where k is specified momentarily. To prove the lower bound, we will allow the adversary to map all distinct loss patterns onto one of the N receivers.

Clearly, the number of distinct loss patterns of this form is $\binom{k}{ck}$, and we choose k as the minimum integer such that $\binom{k}{ck} \geq \min(N, W)$. We then specify a one-to-one, onto mapping of loss patterns to a subset of receivers of size $\binom{k}{ck}$. Transmissions to a given receiver in this subset are lost according to the corresponding loss pattern, repeated indefinitely. This setting ensures that the quality of service guarantee provided by the bounded adversary is satisfied. A diagram of loss patterns specified in this manner is given in Figure 5.1 for four receivers where $c = \frac{1}{2}$ and $k = 4$. The adversary benefits from this specification of packet loss as follows:

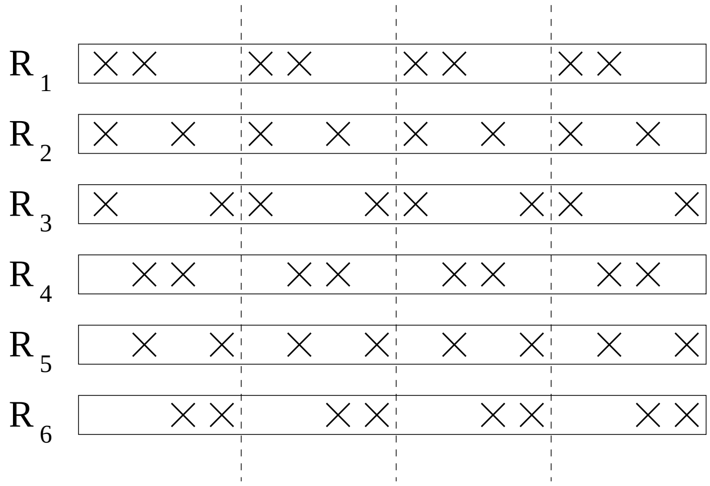

Figure 5.1: Loss patterns for 6 receivers with $c = 1/2$ and $k = 4$.

Claim 35 *If any data word is sent at most ck times, it is guaranteed not to arrive at some receiver.*

Proof: (of Claim) Given any sequence of at most ck transmissions, consider their position in the transmission sequence modulo k: $x_1, x_2, \ldots x_i$, where $i \leq ck$. By our specification of a loss pattern, there is at least one loss pattern which drops all packets with positions $x_1, x_2, \ldots x_i$ modulo k in the transmission sequence. Furthermore, by our choice of k, we map every such loss pattern to a receiver, so there is a receiver who receives none of the transmissions in the sequence. ∎

The claim above implies that in order for a data word to arrive at all receivers, it must be transmitted at least $ck+1$ times. By standard bounds on $\binom{k}{ck}$:

$$\min(N,W) \simeq \binom{k}{ck} = \Omega\left(\frac{1}{c}\right)^{ck}.$$

Therefore, each data word must be transmitted at least

$$ck = \Omega\left(\frac{\log(\min(N,W))}{\log\left(\frac{1}{c}\right)}\right)$$

times, which immediately gives the bound on the ratio for the minimum prefix measure A similar argument gives the same bound for the average prefix measure and the theorem follows. ■

5.3.3 Multicast Data Selection with FEC

Given the definitions and descriptions of efficient forward error correcting codes and the properties of the bounded adversary, the algorithm whose performance proves the following theorem is simple to state:

Theorem 36 *There is a multicast data selection policy which uses FEC with competitive ratio $O(1/c)$ with respect to both the minimum and average prefix measure on transcripts produced by a (c, W) bounded adversary.*

Proof: Our algorithm uses FEC codes and relies on the property of the bounded adversary that each user receives at least cW packets out of every consecutive set of W transmitted packets. For each *block* of cW consecutive message words, the algorithm constructs a set of redundant words of size $(1-c)W$ using FEC encoding. The algorithm then runs in a sequence of phases, each consisting of W transmissions, such that during each phase i, the algorithm transmits the cW message words and $(1-c)W$ codewords generated for the block of message words in the interval $(ciW, ..., c(i+1)W)$. By the bounded adversary, each receiver receives at least cW of the W transmitted words in each block. Then by the FEC coding guarantee described earlier in this chapter, every receiver has sufficient information to quickly reconstruct the original cW message words of this block in their entirety.

The algorithm clearly maintains the following invariant:

Invariant: At the end of phase i, each receiver has a prefix of length exactly ciW.

By the bound on the number of transmissions in each phase, no algorithm could deliver a prefix of length larger than iW to any receiver by the end of round i. Since this argument also bounds the maximum and average prefix at the end of round i, an algorithm which achieves the above invariant performs within a factor of $1/c$ of optimal under both measures. The performance ratio for the algorithm follows. ∎

Of course, this algorithm is limited by the extent to which it is possible to accurately estimate the values of the parameters c and W, which may be difficult to realize in practice on existing networks. In the event that these performance guarantees are not provided, we suggest the following algorithm. For some fixed x and y, decompose the message into blocks of size x and generate a set of redundant codewords for each block of size y. Then, for each block, transmit the message words and the codewords. For each packet arrival at each receiver, that receiver acknowledges whether it is able to reconstruct the block to which the message word in the packet belongs. Alternatively, that receiver could specify *how many* additional words it would need to reconstruct the block. By employing the same acknowledment propagation technique at internal nodes of the tree that was used for the randomized algorithm, the source can either determine whether additional transmissions about the block are necessary, or a lower bound on how many additional transmissions are necessary. The source could then interleave those additional transmissions with transmissions about subsequent blocks of the message.

In the event that packet loss rates and the duration of bursty loss periods can be bounded, the use of efficient FEC codes can provide dramatic performance improvements over retransmission based strategies which still predominate in existing multicast protocols. The empirical work described in the following section provides a preliminary indication of the magnitude of the improvement.

5.3.4 Empirical results

We tested the performance of our data selection policies by simulating their behavior under various conditions. The bulk of the testing was done on traces of MBone audio broadcasts, each lasting approximately one hour, broadcasting packets of size 2.5Kb at a constant rate of every $40ms$, taken from [YKT 96]. Each broadcast was received by between ten and sixteen receivers from the U.S. and Europe, and each receiver recorded which packets it received. Although timing information was not recorded in the traces, in our subsequent experiments, we estimated a typical round-trip time to the most distant receiver of about 1200 ms. Using this estimate, we estimated a typical backlog of approximately 30 packets, a number we used in simulating the behavior of our algorithms. Using these packet loss traces as transcripts on which the performance of data selection policies can be tested, we compared the throughput of our algorithm which employs FEC against two natural deterministic retransmission-based algorithms which we define momentarily. Additional tests were done on transcripts we generated at random as described in more detail below.

For the purpose of the comparison, we assumed that the retransmission based algorithms received full feedback from each of the receivers on the packets they received after a round-trip time. (Of course, this would not be possible in practice because of feedback implosion, but we chose to give these algorithms this extra advantage). Both of the retransmission-based policies are greedy, in that they deterministically choose the word not currently in transit which, if delivered successfully, maximizes the benefit at the receiver. The first algorithm attempts to send a word which will increase the length of the minimum prefix across all receivers:

GreedyMin: At each transmission time, send the smallest unlocked message word that has not arrived at one or more of the receivers.

The second algorithm attempts to increase the average prefix length across all receivers, preferably by sending a word which would extend the prefix of many receivers simultaneously.

GreedyAvg: At each time step, send the smallest unlocked message word which *if successfully received* would increase the lengths of the prefixes of the receivers by the maximum aggregate amount.

Notice that both of these algorithms have the property that it could be the case that no unlocked word can increase the minimum (or average) prefix, and hence the algorithm would then send the smallest clean word. This observation indicates a potential flaw in these algorithms that is borne out in our simulations: it is possible that in a setting with substantial backlog, all words which can contribute to a receiver's prefix are depleted rapidly, after which the algorithm must transmit clean words. This in turn tends to cluster important words close together in the transmission sequence. But with the bursty loss patterns observed in these traces by Yajnik et al [YKT 96], temporal correlation of loss implies that entire clusters may be lost and not be retransmitted until a round trip time elapses. This provides some intuition as to why randomization in the unicast setting can be beneficial, as it can separate these clusters from close temporal proximity, making them less susceptible to burst loss. Similarly, the use of forward error correction in the multicast setting makes all words in the stream equally important, and diminishes the harmful consequences of a burst loss event.

In Figure 5.2 we present the result of our simulations using a representative MBone trace as our transcript. For our algorithm we used $W = 200$ packets and $c = .75$, implying that the network would guarantee delivery of some 150 packets out of every 200. On this transcript, two overseas receivers experienced a blackout period in which they received no packets for several minutes, so these receivers were not included in the simulation. In the plot in Figure 5.2, we plot the performance of our algorithms in delivering a bulk message to the 8 remaining receivers in the transcript. The available bandwidth is a plot of the (unachievable) value of the average prefix length at each point in time, under the assumption that *every* successful transmission to a receiver contributed to that receiver's prefix. This baseline gives a theoretical upper bound on the performance of any data selection policy. For the GreedyAvg policy, which does not attempt to provide any guarantee with respect to the smallest prefix over all receivers, we plot the length of the average prefix obtained over time. For both the FEC policy and for GreedyMin, the minimum prefix and the average prefix achieved by the algorithm are essentially the same at any point in time, as both algorithms focus exclusively on the receiver with minimal prefix. To provide a fair comparison, we also plot the performance of these algorithms with respect to the average prefix measure.

In order to test the scalability of the algorithm we also tested it on larger group sizes. Since we know of no trace data for large multicast groups that are publicly available, we performed the tests on simulated data where packets were lost independently at random for each receiver with probability .2. We present the performance tests for receiver sets of size 12 and 24 in Figures 5.3 and 5.4.

The main observation we make is that while there is a performance penalty associated with using the FEC algorithm due to transmission of redundant words, the magnitude

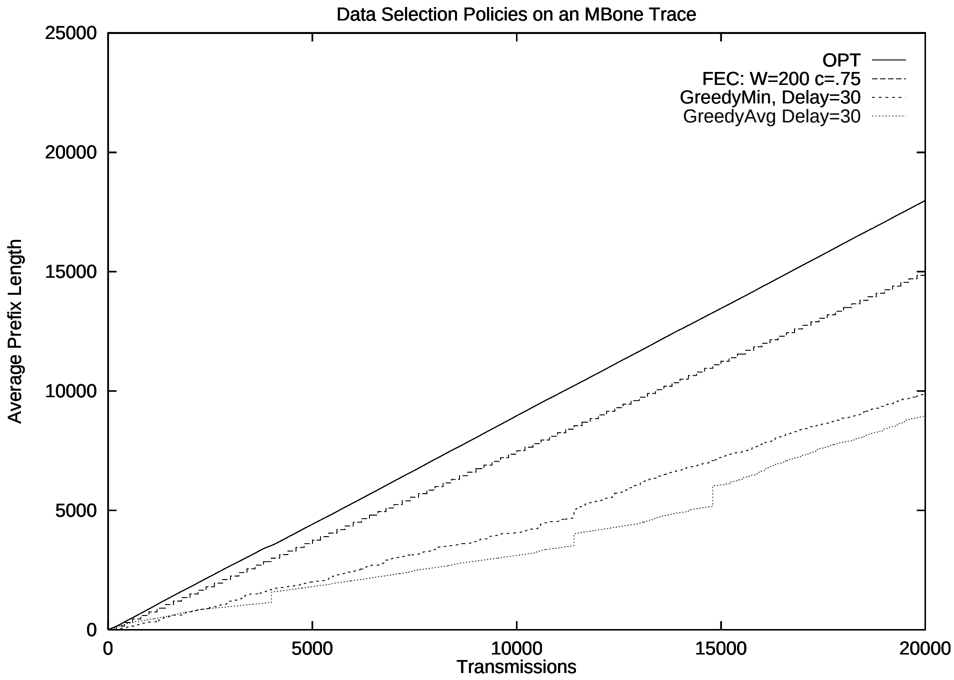

Figure 5.2: Multicast policies on an MBone Trace with 8 Receivers

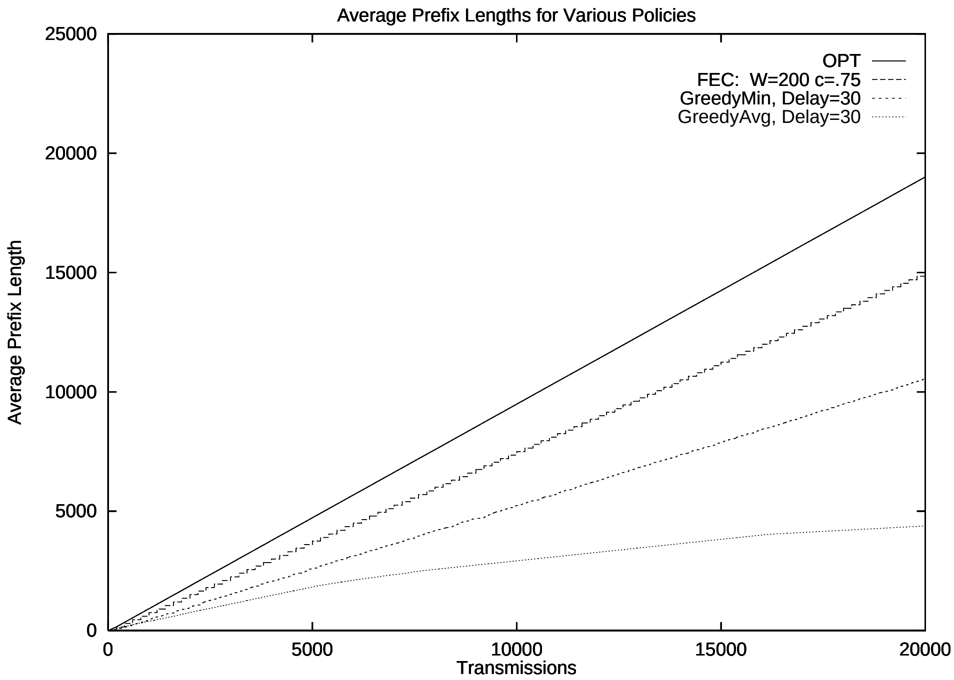

Figure 5.3: Simulated multicast with 12 receivers.

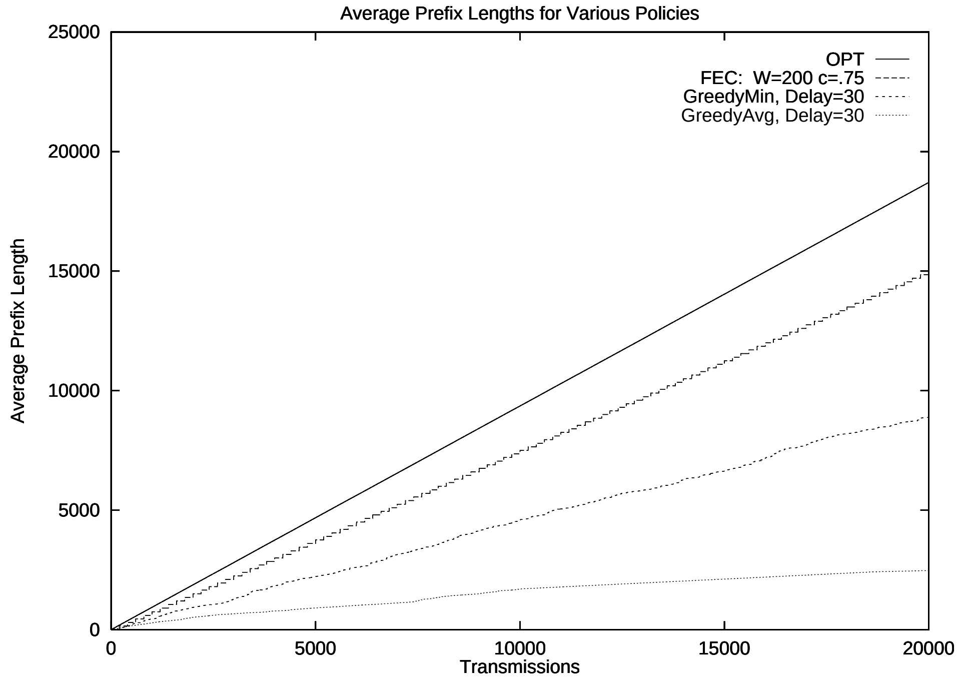

Figure 5.4: Simulated multicast with 24 receivers.

of the penalty is fixed as the number of receivers increases. This can be contrasted with the performance of the greedy, retransmission based strategies, whose performance is acceptable for very small group sizes and when packet loss rates are small, but whose performance degrades rapidly as group sizes scale and packet loss rates increase. Similar observations were made in related studies of the benefits of FEC for reliable multicast of Nonnenmacher et al [NBT 97]. Surprisingly, while one might expect the GreedyAvg policy to deliver a larger average prefix than the GreedyMin policy, in practice, the GreedyAvg policy performs much more poorly. An explanation for this phenomenon is the short-sightedness of the GreedyAvg policy, in that it frequently transmits words which benefit a single receiver immediately, but which do not contribute to the prefix of other receivers until much later in time. The GreedyMin policy on the other hand focuses attention on receivers which have fallen behind, and once those receivers catch up, subsequent transmissions increase the prefix length of many receivers simultaneously.

Another important benefit of using the FEC scheme worth emphasizing which is not depicted in the plots is the minimal feedback required to employ the policy in practice. If the network quality of service guarantee is realized, receivers need not transmit any feedback to the sender, and in the event that the service guarantee is not realized, the receiver need only inform the sender the quantity of additional redundant packets it requires to reconstruct the block. Without an underlying service guarantee, one might consider combining the FEC technique with a sampling technique in order to tune the parameters W and c dynamically in periods of bursty loss. This would enhance the scalability of the algorithm as the number of receivers grows very large and providing service guarantees is no longer realistic.

Chapter 6

Conclusions and Future Work

In this thesis, we have presented techniques for maximizing the utilization of a shared resource, network bandwidth. For the studies we performed, we focused on the largest consumers of this resource, bulk transfer applications, in which a large volume of data has to be reliably transmitted from a single source to one or more destinations. The first technique we considered established a rate-based allocation of end-to-end bandwidth to bulk connections so as to maximize the aggregate throughput of the connections, measured as the sum of the transmission rates. Starting from related work which proved tight bounds on how well a set of point-to-point connections could approximate this global objective using only local information, we explored the tradeoff between the amount of communication and the quality of the approximation. Our main result shows that a small (polylogarithmic) number of distributed communication rounds suffice to generate a near-optimal $(1+\epsilon)$ approximation to the optimum solution. Moreover, extensions of this distributed technique apply to allocations in which reliable multicast and layered multicast connections participate. Rate-based allocation techniques with provable performance guarantees provide an alternative to standard congestion control techniques, in which connections make dynamic, and often rapid adjustments to their transmission rates based on the presence or absence of perceived congestion along their route.

The second technique we addressed focuses on the on-line scheduling problem of determining what data to place in each packet of a bulk transfer to maximize the effective throughput at the receivers throughout the transmission. The policies we develop for this problem can be used in conjunction with the rate-based allocation policies we develop, or in conjunction with any other mechanism for regulating the rate at which connections inject packets into the network. Factors shaping the development of the Internet motivate the design of policies which deliver high performance even as the impact of other users in the network causes transmission rates to fluctuate, round-trip times to vary widely and packets to be lost. For the unicast case, one of our main contributions is a randomized data selection policy which delivers provably high performance with respect to competitive measures on arbitrary event sequences. This result further implies that data selection policies need not be tuned for ideal performance with a given rate allocation policy – the algorithm we present will deliver near-optimum performance when coupled with *any* rate allocation policy. This provides further motivation for the modular separation we advocate between policies which regulate the transmission *rate* and policies which regulate the *content* of transmissions.

For multicast applications, we show that the use of coding techniques, and forward error correction in particular, is a prerequisite for a multicast data selection policy to deliver provably high performance. The use of forward error correction has an additional benefit in that it can dramatically reduce the number of transmissions needed to complete a reliable bulk transfer to multiple receivers when packet loss is present. Moreover, the gains which are realized over strategies which use packet retransmission grow with the number of receivers in the multicast group.

6.1 Future Work

For the problem of rate allocation, there are several avenues along which to perform further exploration. One consideration is the discrepancy between the performance predicted by the analysis and the performance realized by the actual implementation. For the examples we have tested, the algorithm realizes a considerably closer approximation ratio than the target ratio. For a given setting of ϵ, realization of an $(1+\frac{\epsilon}{10})$ approximation is quite common. This has clear implications for the specification of ϵ in practice, and it would be nice to tighten the analysis or exhibit examples on which the current analysis is tighter. A related problem regards the distributed running time – for all examples on which we have run the algorithm, the dual feasible solutions achieve the desired approximation ratio after only 1/10th of the run is complete. A natural question would be to ask if there is a way to exploit this behavior and improve the running time, either by taking less conservative steps while increasing primal variables, or by improving the mechanism which correlates values of primal and dual variables.

Another direction would be to consider generalizing the objective functions over which we perform the optimization. The weighted throughput version of the problem can be viewed as a profit maximization problem on pay-per-use networks in which connections are willing to pay a fixed amount for each unit of end-to-end rate which they are allocated. But in many economic settings, of which this is an example, the consumer is likely to derive diminishing marginal benefit for each additional unit of commodity beyond some threshold. This motivates study of a related non-linear optimization problem in which the objective function reflects the demand curves of the connections.

Along implementation-based lines, a promising direction would be to deploy the allocation policy we have described here in existing networks which support rate-based allocation. An important aspect of deployment would be to expand on the heuristics we describe for dynamic adjustment of the allocation as bulk connections arrive and depart, and to define the integrated service mechanisms needed to allow bulk connections to coexist with interactive and streaming applications. In practice, it might also be necessary to guarantee some minimal level of service to bulk transfers which are present but do not contribute to the throughput maximizing allocation.

For data selection, deploying the policies in existing network applications is considerably easier, as it can be done at the application level, without modifying existing protocols. The challenge behind the implementation of reliable bulk multicast with forward error correction is to judiciously choose the right settings for the block size and the amount of redundancy. Ideally, one would hope to keep W relatively small to minimize the delay of reconstructing each message block and to keep c large, but small enough to tolerate a bursty period of high packet loss. This can be complicated by the fact that the optimal values for the settings may change over time. Another research possibility for reliable multicast is to incorporate forward error correction with other approaches to minimize the number of retransmissions, such as local recovery. Finally, there are several other multicast applications, such as reliable layered multicast and unreliable streaming applications, for which one could quantify reasonable data selection policy performance measures. For real-time streaming applications, where receivers are not concerned with the completion time of the broadcast, but only the quality of service during the broadcast. A possibility for modifying our performance metric for this application would be to consider the fraction of recently transmitted packets that arrived at the receiver, since earlier transmissions are not typically relevant to current performance. The design of data selection policies which perform well with respect to these measures would be another interesting research direction.

Bibliography

[AA 94] B. Awerbuch and Y. Azar. Local Optimization of Global Objectives: Competitive Distributed Deadlock Resolution and Resource Allocation. In *Proc. of the 35th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 240-249, 1994.

[AAB 97] B. Awerbuch, Y. Azar, and Y. Bartal. Local Multicast Rate Control with Globally Optimum Throughput. Unpublished manuscript, 1997.

[AAFKLL 96] D.M. Andrews, B. Awerbuch, A. Fernandez, J. Kleinberg, F.T. Leighton, Z. Liu. Universal Stability Results for Greedy Contention-resolution Protocols. In *Proc. 37th IEEE Symposium on Foundations of Computer Science*, pp. 380-389, 1996.

[AAFPW 92] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-line Load Balancing with Applications to Machine Scheduling and Virtual Circuit Routing. In *Proc. of the 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pp. 164-173, October 1992.

[AAP 93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput Competitive On-line Routing. In *Proc. of the 34th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 32-40, November 1993.

[ABBLR 97] M. Adler, Y. Bartal, J. Byers, M. Luby and D. Raz. A Modular Analysis of Network Transmission Protocols. *Proc. of the 5th Israeli Symp. on Theory of Computing and Systems,* June 1997.

[ABELS 96] A. Albanese, J. Bloemer, J. Edmonds, M. Luby and M. Sudan. Priority Encoding Transmission. In *IEEE Transactions on Information Theory* (special issue on coding theory), Volume 42, No. 6, November 1996, pp. 1737-1744.

[AMK 97] E. Amir, S. McCanne and R. Katz. Receiver-driven Bandwidth Adaptation for Light-weight Sessions. In *ACM Multimedia '97*, November 1997.

[AMO 96a] Y. Afek, Y. Mansour and Z. Ostfeld. Convergence Complexity of Optimistic Rate Based Flow Control Algorithms. In *Proc. of 28th ACM Symposium on Theory of Computing*, pp. 89-98, 1996.

[AMO 96b] Y. Afek, Y. Mansour and Z. Ostfeld. Phantom: A Simple and Effective Flow Control Scheme. In *Proc. of ACM SIGCOMM '96*, pp. 169-182, August 1996.

[AS 97a] B. Awerbuch and Y. Shavitt. Converging to Approximated Max-Min Flow Fairness in Logarithmic Time. Johns Hopkins Tech Report, 1997.

[AS 97b] B. Awerbuch and T. Singh. Online Algorithms for Selective Multicast and Maximal Dense Trees. In *Proc. of the 29th ACM Symposium on Theory of Computing*, pp. 354-362, 1997.

[BBR 97] Y. Bartal, J. Byers and D. Raz. Global Optimization using Local Information with Applications to Flow Control. In *Proc. of the 38th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 303-312, 1997.

[BG 87] D. Bertsekas and R. Gallagher. *Data Networks.* Prentice Hall, 1987.

[BF 95] F. Bonomi and K. Fendick. The Rate Based Flow Control for Available Bit Rate ATM Service. *IEEE Networks*, pp. 24-39, March/April 1995.

[BKRSW 95] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan and D. Williamson. Adversarial Queueing Theory. In *Proc. 28th ACM Symposium on Theory of Computing*, 1996.

[BOP 94] L. S. Brakmo, S. W. O'Malley and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proc. ACM SIGCOMM '94*, pp. 24-35, 1994.

[BPSK 96] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *Proc. ACM SIGCOMM '96*, pp. 256-269, 1996.

[Cha 94] A. Charny. An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback. Technical Report MIT/LCS/TR-601, MIT Laboratory for Computer Science, April 1994.

[CLZ 87] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: A High Throughput Transport Protocol. *Computer Communications Review*, 17(5):353-359, 1987.

[DKS 89] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proc. ACM SIGCOMM '89*, pp. 3-12, 1989,

[FF 96] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *Computer Communications Review*, 26(3), July 1996.

[FJLMZ 95] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proc. ACM SIGCOMM '95*, pp. 342-356, August 1995.

[FJ 93] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. In *IEEE/ACM Transactions on Networking,* 1(4):397-413, August 1993.

[FJ 94] S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. In *ACM Transactions on Networking,* 2(2):122-136, April 1994.

[Hoe 96] J. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *Proc. ACM SIGCOMM '96*, pp. 270-280, 1996.

[Hui 96] C. Huitema. The Case for Packet Level FEC. In *Proc. of IFIP 5th Int'l Workshop on Protocols for High Speed Networks,* Sophia Antipolis, France, October 1996.

[Jac 88] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM '88*, pp. 314-329, 1988.

[JB 88] V. Jacobson and R. Braden. TCP Extensions for Long-Delay Paths, October 1988. RFC 1072.

[Jaf 81] J. Jaffe. Bottleneck Flow Control. In *IEEE Transactions on Commmunication,* COM-29, 1(7), pp. 954-962, July 1981.

[Kar 96] H. Karloff. *Linear Programming.* Birkhauser, 1996.

[Kes 91] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Proc. ACM SIGCOMM '91*, pp. 3-16.

[L 87] N. Linial. Distributive Graph Algorithms - Global Solutions from Local Data. In *Proc. of the 28th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 331-335, 1987.

[LMSSS 97] M. G. Luby, M. Mitzenmacher, M. Amin Shokrollahi, D. A. Spielman, and V. Stemann. Practical Loss-Resilient Codes. In *Proc. of 29th ACM Symposium on Theory of Computing*, pp. 150-159, 1997.

[LN 93] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *Proc. of 25th ACM Symposium on Theory of Computing*, pp. 448-457, 1993.

[LTWW 95] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). In *IEEE/ACM Transactions on Networking,* Vol. 2, No. 1, pp. 1-15, Feb. 1995.

[MSZ 96] Q. Ma, P. Steenkiste, and H. Zhang. Routing High-bandwidth Traffic in Max-min Fair Share Networks. In *Proc. ACM SIGCOMM '96*, pp. 206-217, 1996.

[MS 77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes,* North Holland, Amsterdam, 1977.

[MM 96] M. Mathis and J. Mahdavi. Forward Acknowledgment: Refining TCP Congestion Control. In *Proc. of ACM SIGCOMM '96*, pp. 281-291, 1996.

[McC 96] S. McCanne. *Scalable Compression and Transmission of Internet Multicast Video,* PhD Thesis, University of California, Berkeley, UCB/CSD-96-928, December 1996.

[MJV 96] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In Proc. of ACM SIGCOMM '96, pp. 117-130, 1996.

[Mos 84] J. Mosley. *Asynchronous Distributed Flow Control Algorithms.* PhD Thesis. MIT, June 1984.

[NB 96] J. Nonnenmacher and E. W. Biersack. Reliable Multicast: Where to Use Forward Error Correction. In em Proc. of IFIP 5th Int'l Workshop on Protocols for High Speed Networks, pp. 134-148, Sophia Antipolis, France, October 1996. Chapman and Hall.

[NBT 97] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *Proc. of ACM SIGCOMM '97*, 1997.

[Pax 96] V. Paxson. End-to-End Routing Behavior in the Internet. In *Proc. ACM SIGCOMM '96*, pp. 25-38, 1996.

[Pax 97] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics,* PhD Thesis, University of California, Berkeley, 1997.

[PSLB 97] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). To appear in *IEEE Journal on Selected Areas in Communications, a special issue on Network Support for Multipoint Communications.*

[PF 95] V. Paxson and S. Floyd. Wide Area Traffic: The Failure of Poisson Modeling. In *IEEE/ACM Trans. on Networking*, 3(3):226-244, June 1995.

[PST 94] S. Plotkin, D. Shmoys and E. Tardos. Fast Approximation Algorithms for Fractional Packing and Covering Problems. *Technical Report ORIE-999 of the School of Operations Research and Industrial Engineering, Cornell Univer*, 1995. A preliminary version of this work appeared in Proc. of 25th ACM Symp. on Theory of Computing, 1993.

[PY 93] C. Papadimitriou and M. Yannakakis. Linear Programming without the Matrix. In *Proc. of 25th ACM Symposium on Theory of Computing*, pp. 121-129, 1993.

[Riz 97] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. In *Computer Communication Review*, April 1997.

[RV 97] L. Rizzo and L. Vicisano. A Reliable Multicast data Distribution Protocol Based on Software FEC Techniques. In *Proc. of HPCS '97*, Greece, June 1997.

[ST 85] D. Sleator and R. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2):202-208, 1985.

[Vic 97] L. Vicisano. Notes on a Cumulative Layered Organization of Data Packets Across Multiple Streams with Different Rates. Unpublished manuscript, November 1997.

[WTSW 95] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. In *Proc. ACM SIGCOMM '95*, pp. 100-113, Cambridge, MA, August 1995.

[YKT 96] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network, In *Proceedings of IEEE Global Internet '96,* London, November 1996.

[YM 93] R. Yavatkar and L. Manoj. Optimistic Strategies for Large-scale Dissemination of Multimedia Information. In *Proceedings of ACM Multimedia '93*, August 1993.