OPTIMIZATION TECHNIQUES IN COMMUNICATION NETWORKS

By

XIAOYING ZHENG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2008

To my beloved husband, Junqi;

and to my parents.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Ye Xia, for his constant guidance, support, patience, and encouragement. I am privileged to have such a wonderful advisor, who is at all times enthusiastic, optimistic, patient, helpful, and encouraging. He gave me countless advice and insight during the course of my work, without which completing this dissertation would have been much more difficult.

I would like to express my appreciation to my committee, Professor Arunava Banerjee, P. Oscar Boykin, Shigang Chen and Randy Chow. Thanks to Professor Yuguang Fang, who has been working with me on the cross-layer design project.

I am thankful to the office-mates and friends that I met at the University of Florida, for supporting me over the years and for making my life at UF enjoyable. Special thanks to Chunglae Cho and Feng Chen, who have been working with me on the content distribution and cross-layer design projects, respectively.

I would like to thank my parents who have always loved me and encouraged me in my life. Thanks also go to my parents-in-law, my brother and his wife for their support.

Finally, I am particularly grateful to my darling husband, Junqi Zhao. We have been geographically separated by the Pacific Ocean through my entire Ph.D. life. In the past five years, he has been spending almost every morning and night accompanying me by the Webcam, which makes me emotionally strong and gives me courage to make my Ph.D. dream real. Without his endless love and support, this dissertation would not be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

OPTIMIZATION TECHNIQUES IN COMMUNICATION NETWORKS

By

Xiaoying Zheng

August 2008

Chair: Ye Xia
Major: Computer Engineering

Convex optimization techniques have found important applications in communications and signal processing. Recently, there has been a surge in research activities that apply the latest development in convex optimization to the design and analysis of communication systems. This research focuses on how to apply optimization techniques on both massive content distribution in high-speed internet core, and cross-layer design in wireless ad hod networks.

One of the important trends is that the Internet will be used to transfer content on more and more massive scale. Collaborative distribution techniques such as swarming and parallel download have been invented and effectively applied to end-user file-sharing or media-streaming applications, but mostly for improving end-user performance objectives. We consider the issues that arise from applying these techniques to content distribution networks for improving network objectives, such as reducing network congestion. In particular, we formulate the problem of how to make many-to-many assignment from the sending nodes to the receivers and allocate bandwidth for every connection, subject to the node capacity and receiving rate constraints. The objective is to minimize the worst link congestion over the network, which is equivalent to maximizing the distribution throughput, or minimizing the distribution time. The optimization framework allows us to jointly consider server load balancing, network congestion control, as well as the requirement of the receivers. We develop a special, diagonally-scaled gradient projection algorithm, which has a faster convergence speed, and hence, better scalability

with respect to the network size than a standard subgradient algorithm. We provide both a synchronous algorithm and a more practical asynchronous algorithm.

However, on the negative side, swarming traffic has made capacity shortage in the backbone networks a genuine possibility, which will be more serious with fiber-based access. The second problem addressed is how to conduct massive content distribution efficiently in the future network environment where the capacity limitation can equally be at the core or the edge. We propose a novel peer-to-peer technique as a main content transport mechanism to achieve efficient network resource utilization. The technique uses multiple trees for distributing different file pieces, which at the heart is a version of swarming. In this chapter, we formulate an optimization problem for determining an optimal set of distribution trees as well as the rate of distribution on each tree under bandwidth limitation at arbitrary places in the network. The optimal solution can be found by a distributed algorithm. The results of the chapter not only provide stand-alone solutions to the massive content distribution problem, but should also help the understanding of existing distribution techniques such as BitTorrent or FastReplica.

The joint optimal design of congestion control and wireless MAC-layer scheduling problem in multi-hop wireless networks has become a very active research area in the last few years. We solve this problem using a column generation approach with imperfect scheduling. We point out that the general subgradient algorithm has difficulty in recovering the time-share variables and experiences slower convergence. We first propose a two-timescale algorithm that can recover the optimal time-share values. Most existing algorithms have a component, called global scheduling, which is usually NP-hard. We apply imperfect scheduling and prove that if the imperfect scheduling achieves an approximation ratio ρ, then our algorithm converges to a sub-optimum of the overall problem with the same approximation ratio. By combining the idea of column generation and the two-timescale algorithm, we derive a family of algorithms that allow us to reduce the number of times the global scheduling is needed.

CHAPTER 1
INTRODUCTION

1.1 Research Motivation

With the progress of networking technologies, the development of the domain is shifting from providing connectivity to providing a multitude of services and applications with desirable quality and reliability. The shift was foreseen more than a decade ago. However, this vision then was to develop an integrated network, the ATM netwotk, capable of doing all the above. The recent reality has been that, the networking community tends to invent new physical or logical networks, or make other creative use of existing networks, for new applications. An evidence is the emergence of peer-to-peer (P2P) or other overlay networks over the last several years. These include BitTorrent [1], Kazaa [2] and Gnutella [3], and various content distribution networks. An overlay (including P2P) network is layered on top of a physical network, such as the Internet, and relies on the latter for reachability. The nodes of such a network are usually end systems running user-level applications. This gives the overlay network the flexibility to form different topologies and the ability to pool resources, such as storage capacity, computational power and communication bandwidth. In an effort to achieve new capabilities, improved performance, and higher reliability, researchers are often willing to challenge the traditional networking principles and practices.

One notable attempt is that multi-path or parallel transmission has been proposed for virtually all these networks, transcending the convention of single-path routing in the Internet. For instance, in the content-distribution application, each file or pieces of the file can be replicated at many nodes in the network. A user who wishes to retrieve the file can establish connections with multiple servers and schedule the download in parallel (See [4, 5] for examples.). This setup can often dramatically speed up the downloading process, enhance the performance, and improve the reliability.

An even more aggressive content distribution approach is to do it by packing multiple multicast trees. By multicast, the content is simultaneously delivered to all the users over each

link of the network only once, and the content gets duplicated only when the links split [6]. This fruitful way of viewing the many-to-many or collaborative approach not only achieves efficient network resource utilization but also helps the understanding the existing distribution techniques such as BitTorrent or FastReplica.

The study of the content distribution problem has led our research to the far more complicated wireless environment. Unlike the wired network case, the capacities of wireless links are not constant, but depend on many factors, such as power, transmission interference and ambient noise, medium contention and resolution (by scheduling), the state of the receivers (receiver numbers, locations, and mobility etc.), and random environmental factors. Conceptually, the study of the control algorithms should be the joint design over the flow rates, and the power allocation and scheduling strategies. Such network problem is currently a very popular theme in network research [7–19].

The methodological framework for all the three proposed problems is the theory and algorithms of network optimization. The unifying treatment of the three problems within the common theoretical and algorithmic framework is a distinguishing feature of this research. In addition, the network optimization theory and algorithms developed in the research is applicable to a large number of similar problems within or beyond this dissertation.

1.2 Why Optimization Framework?

Convex optimization techniques are widely used in communications and signal processing, because they can be used to analyze and interpret a system in a rigorous way, and often provide powerful numerical algorithms. Recently, there has been a surge in research activities that apply the latest development in convex optimization to the design and analysis of communication systems, which is both driven by the significant advances in the research of convex optimization over the last two decades, and the new demands in communications and networking areas [20, 21].

The optimization techniques have been applied to solve network flow problem as early as in [22–27]. In these work, the target problem is a linearly constrained optimization problem of the

form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & Ax \leq b. \end{aligned}$$

The gradient projection methods are often applied to solve the above problem. The difficulty of solving the primal problem directly by the gradient projection method is that the projection operation often involves another difficult optimization problem. The projection operation will be practical if the constraint set has a simple form, for instance, hyper-rectangle, Cartesian products of simplices, etc.; or has a special structure, for example, it expresses flow conservation at each node in a directed graph [23]. The Frank-Wolfe type [28] gradient projection methods only utilize the first derivative of the objective function, which have been extensively applied to multicommodity network flow problems [29], but the convergence rate of these methods is sublinear and therefore too slow for applications [23]. Bertsekas et al. propose to use the second derivative of the objective function in the gradient projection methods, which approximate the well-know Newton's method and often result in improved speed of convergence in the network flow problems [23, 24, 30].

The validity of applying the theory and algorithms of optimization to network problem has been most convincingly established by the successful line of research in optimal flow control started by Kelly [31] and Low [32] and followed by many other works [33–40]. The first papers recognize that, rather than vaguely understanding flow/congestion control as mechanisms to prevent receiver or network congestion, one can clearly describe the control objective and find the best solution by optimization or control algorithms, instead of non-optimal, heuristic solutions. Furthermore, one can broaden the goal as maximizing all users' utilities subject to the link capacity constraints (hence, there is no network congestion.). Let S denote the set of users, x_s be users s's flow rate, a concave function U_s be its utility function, and let $S(e)$ be the set of user

flows passing through link e. The (single-path) utility maximization problem is

$$\begin{aligned} \max \quad & \textstyle\sum_{s\in S} U_s(x_s) \\ \text{s.t.} \quad & \textstyle\sum_{s\in S(e)} x_s \le c_e, \quad \forall e \in E \\ & x_s \ge 0, \qquad \forall s \in S. \end{aligned}$$

This is a convex separable optimization problem with linear constraints. For such a problem, one can automatically derive a distributed algorithm by applying the general dual descent algorithm. The algorithm consists of a link part and a source part. There is one dual variable, p_e, associated with each link $e \in E$, corresponding to each capacity constraint. It has the interpretation as the price of the link. The path cost is $p^s = \sum_{e\in L(s)} p_s$, where $L(s)$ is the set of links used by the flow of s. In the k^{th} step of the iterative algorithm, the source algorithm is $x_s(k+1) = (U_s')^{-1}(p^s(k))$ and the link algorithm is $p_e(k+1) = [p_e(k) + \delta(\sum_{s\in P(e)} x_s - c_e)]_+$, where $\delta > 0$ is a chosen step size for the update, and $[\cdot]_+$ is the projection onto the non-negative domain.

The link algorithm has the straightforward interpretation that the link price increases when the aggregate rate crossing the link exceeds its capacity, and decreases otherwise. It is a local algorithm, requiring only local information. The source algorithm is distributed but requires the end-to-end path price. This is not a difficulty since network flow control protocol routinely passes control messages on the path, which are returned back to the source after a roundtrip. We see that a standard distributed optimization algorithm becomes the main ingredients of a TCP-style network congestion control algorithm. For different utility functions, there will be different "congestion control" algorithms. With the optimization theory, one can also prove the algorithm in fact works (converges), and show how fast the algorithm converges to the optimal solution. The range of valid parameters can also be identified and later turned for performance improvement when the algorithms is in operation. Many alternative distributed algorithms have also been explored for reasons such as improved convergence speed, lower implementation complexity, ease of deployment, and secondary objectives such as reducing the network queue sizes.

The success of applying optimization theory to network flow control has led to its application to far more complicated network environment, from the high-speed internet core [34, 35, 41–45] to wireless ad hoc networks [7–19].

1.3 Research Overview

Our first research problem is how to conduct massive content distribution efficiently in future network environment where the capacity limitation can equally be at the core or the edge. This problem is important because as the Internet is being used to transfer content on a more and more massive scale, the capacity shortage in the backbone networks has become a genuine possibility, which will be more serious with fiber-based access. Motivated by Kelly and Low's work, we will study the massive content distribution under the optimization framework in both chapter 2 and chapter 3.

In chapter 2, the content distribution is carried by end-to-end unicast traffic. The main problem is how to select a subset of the servers for each client and decide the transmission rate from each selected server to the client, so that the clients get their required bandwidth, the servers are not overloaded and the worst-case link congestion in the network is minimized. We first develop a subgradient algorithm, which works on the dual problem and is the most frequently used algorithm in the networking literature. However, since this problem is often ill-conditioned, our experience has shown that the subgradient algorithm converges slowly . We then develop a special diagonally scaled gradient projection algorithm operating on the primal problem, which tries to emulate the faster Newton's method and overcomes this difficulty.

In chapter 3, the content distribution is more aggressive and is carried by all possible multicast distribution trees. The difficulty is that the number of all possible distribution trees increases exponentially as the network increases. Fortunately, by the gradient projection algorithm, we avoid enumerating all possible trees and are able to identify and use the optimal distribution trees, and at the same time, allocate correct bandwidth on the selected trees. Furthermore, we introduce the column generation method, which introduces one more tree at a time and gradually expands the tree set. By combining the idea of column generation and the gradient projection algorithm,

we derive a family of algorithms that allow us to reduce the number of times the computation of minimum spanning tree is needed.

Motivated by the column generation approach, in chapter 4, we solve the problem of a joint optimal design of congestion control and wireless MAC-layer scheduling using a column generation approach with imperfect scheduling. We point out that the general subgradient algorithm has difficulty in recovering the time-share variables and experiences slower convergence. We first propose a two-timescale algorithm that can recover the optimal time-share values. Most existing algorithms have a component, called global scheduling, which is usually NP-hard. We apply imperfect scheduling and prove that if the imperfect scheduling achieves an approximation ratio ρ, then our algorithm converges to a sub-optimum of the overall problem with the same approximation ratio.

CHAPTER 2
OPTIMIZING NETWORK OBJECTIVES IN COLLABORATIVE CONTENT DISTRIBUTION

2.1 Introduction

With the deployment of high-speed access networks such as fiber-to-the-home (FTTH) or its variants, the Internet will be used to transport data on more and more massive scale. The current and future massive content includes high-definition movies and TV programs, large collection of multimedia data, and mountains of all automatically collected/sensed data such as environmental, scientific, or economic data. Beyond that, visionaries are already contemplating 3D super definition (643 Mbps) or 3D ultra definition (2,571 Mbps) TV, 3D telepresence, and tele-immersion in virtual worlds. Since the bandwidth mismatch between the network access and network core is expected to be sharply reduced, one can no longer assume virtually unlimited capacity in the future backbone network. Instead, one should not be surprised that, however large, the backbone capacity will be used up by future content. As an evidence, with its early adoption of FTTH, by 2005, Japan already saw 62% of its backbone network traffic being from residential users to users, which was consumed by content downloading or peer-to-peer (P2P) file sharing; the fiber users were responsible for 86% of the inbound traffic; and the traffic was rapidly increasing, by 45% that year [46].

An important networking problem addressed by this chapter is how to conduct massive content distribution efficiently in the future network environment where the capacity limitation can equally be at the core or the edge. Our work has applications in the following two-step content distribution process, which is prevalent today and is expected to become more important in the future for reducing wide-area network traffic. In the first step, the content is distributed over infrastructure networks, such as content distribution networks, ISP networks or IPTV networks. In the second step, the end users retrieve the content from one or more nearby content servers. In either step but particularly the first, collaborative distribution techniques are becoming very attractive. By collaborative distribution, we mean different nodes in a distribution session help each other to speed up the distribution or improve other performance

measures. A simple form of collaborative distribution is parallel download of a file from multiple nodes, which improves upon the single-server based approach. A more sophisticated form is known as *swarming*, in which each file is broken into many chunks and the nodes (peers) exchange the chunks with each other. One example of swarming is the popular BitTorrent [1]. Although swarming was originally invented in end-system file-sharing applications, it is really a fundamental distribution technique that can be employed by the operators of content distribution networks.

This chapter describes how to improve collaborative distribution techniques for achieving *network* objectives in content distribution networks. Since most of these techniques were designed for the end-user environment, targeting end-user performance objectives, they need considerable modification and improvement before they can be applied to content distribution networks and achieve important network performance objectives, such as low network congestion or high throughput. In particular, one common assumption of the current end-user systems is that the network is *access-limited*. As a result, they do not have built-in congestion control or bandwidth allocation mechanisms that can coordinate the entire distribution session and efficiently cope with internal network congestion. Instead, they either rely on the default TCP congestion control, working independently on each individual connection, or do not have any congestion control at all if UDP is used. As will be demonstrated in the chapter, they either cause unnecessarily heavy network congestion at parts of the network (due to poorly balanced network load), or miss the opportunity to achieve a shorter distribution time (or equivalently, a higher throughput) given the same network congestion level. The performance gap between what these systems can accomplish and the best possible can be very wide.

This chapter proposes a scheme that makes coordinated bandwidth assignment among different connections in the same distribution session so as to minimize the worst-case network congestion, or equivalently, maximize the distribution throughput. The coordination is achieved through fully distributed algorithms. For ease of discussion, we call the receiving nodes the

clients and the transmitting nodes the *servers*[1] . The scenario under investigation concerns a set of clients requesting chunks of a file (files) or streaming media from a set of servers. For simplicity, we assume all servers have the same content, but this assumption is not crucial[2] . Each client can make parallel download from multiple servers simultaneously. (See Fig. 2-1 as an example.) Our problem is to select a subset of the servers for each client and decide the transmission rate from each selected server to the client, so that the clients get their required bandwidth (e.g., for streaming requirement), the servers are not overloaded and the worst-case link congestion in the network is minimized.

Our problem formulation and solution follow the network optimization approach introduced by Kelly et al. [31] and Low et al. [32]. The problem contains a fractional server-selection problem. For instance, a client can get $1/3$ of its download from one server and $2/3$ from another server. If a connection is assigned a zero or near zero bandwidth, the client essentially has not selected the corresponding server. Our solution to the optimization problem leads to distributed algorithms that combine server assignment with congestion control (or equivalently, bandwidth allocation).

Our contributions are as follows. The results from this chapter will be useful for content distribution networks, ISPs and IPTV distribution networks. Up to orders of magnitude improvement in throughput (or reduction in congestion) is possible with our scheme. With respect to network optimization, our solution is a special gradient projection algorithm operating on the primal problem, instead of the subgradient algorithm, which works on the dual problem. For

[1] A node (peer) can be both a client and a server. Every node is a content distribution infrastructure node rather than an end system, although the problem formulation in this chapter does apply to end-system P2P file sharing.

[2] Our proposed scheme works the best if appropriate *source coding* is used, such as the Tornado code [4]. With source coding, the file chunks are coded and a receiver can reconstruct the entire file as long as it receives a sufficient number of chunks, irrespective of the identity of the chunks. Each server may contain an arbitrary collection of coded chunks. The nodes exchange coded chunks without the need of knowing what they are.

similar network flow problems, the latter is the most frequently used algorithm in the networking literature. For our problem, our experience has shown that the gradient projection algorithm has a faster convergence speed than the subgradient algorithm. The main reason is that the problem of minimizing the worst-case congestion is often ill-conditioned [47]. With the gradient projection algorithm, we are able to overcome this difficulty with diagonal scaling, which tries to emulate the faster Newton's algorithm. For improved practicality, we have also developed an asynchronous version of the algorithm. The correctness (i.e., convergence) of all versions of the algorithms has been proven. We also give important results on the convergence speed.

The chapter is organized as follows. In the remaining part of the introduction, we summarize the common notation used throughout the chapter. In Section 2.2, we introduce a linear optimization model and discuss its performance advantage compared with random server assignment together with UDP or TCP. This serves to further motivate our optimization-based problem formulation. In Section 2.3, we describe a fully distributed subgradient algorithm for the linear problem. In Section 2.4, we approximate the linear problem using a barrier-function approach. We then develop a gradient projection algorithm. Next, we extend the gradient projection algorithm to a diagonally-scaled version. Finally, we provide an asynchronous version of the algorithm. In Section 2.6, we present experimental results to compare the performance of the subgradient algorithm and the gradient projection algorithm without and with scaling. In Section 2.7, additional related work is reviewed. The conclusion is in Section 2.8.

Notation. Let the network be represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of (directed) arcs. An arc $e \in E$ can also be denoted by the corresponding ordered node pair (i, j), where $i, j \in V$ are distinct nodes. An arc represents a directed communication link. The link capacity is denoted by c_{ij} for $(i, j) \in E$, or c_e for $e \in E$, interchangeably.

Let $S \subseteq V$ be the set of servers and $C \subseteq V$ be the set of clients. S and C may overlap. Let p_{ij} be the allowed path, for instance, the shortest path, from server i to client j. We regard

$p_{ij} \subseteq E$ as the collection of edges on the path. Let P be the set of all paths p_{ij}, for any $i \in S$ and any $j \in C$.

The flow rate (i.e., traffic rate) from server i to client j is denoted by x_{ij}, or x_p for $p = p_{ij} \in P$, interchangeably. Let $y_e = \sum_{p_{ij} \ni e} x_{ij}$ denote the flow rate through link e for any $e \in E$. The utilization of link e, a measure of link congestion, is denoted by μ_e, and $\mu_e = y_e/c_e$. Let $z_i = \sum_{j \in C} x_{ij}$ denote the total sending rate of server i for any $i \in S$.

In our notation, all vectors are column vectors. Let $||x||$ denote the usual Euclidean norm of a vector x, and $\langle \cdot, \cdot \rangle$ denote the usual vector inner product. For any vector x, we denote by $[x]_+$ the positive part of x (i.e., the vector obtained by replacing each negative component of x with zero). For any matrix E, we denote its induced norm by $||E||$, which is equal to $\max_{||x||=1} ||Ex||$, and we denote by $[E]_{ij}$ the entry on row i and column j. Let $\text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$ denote the $n \times n$ diagonal matrix whose diagonal elements are the λ_i's. For any finite set S, we denote by $|S|$ the cardinality (number of elements) of S.

2.2 Problem Formulation and Motivation

In this section, we will give a linear programming formulation of the problem with the objective of minimizing the worst-case link congestion, under server and client side capacity (or required bandwidth) constraints. The main objective of the chapter is to solve the problem using distributed algorithms. Before doing that, to further motivate the problem formulation, we first solve the problem using centralized approach and demonstrate the possible performance gain against existing popular heuristic algorithms.

2.2.1 Optimal Flow Assignment Problem

Suppose the total capacity of server i is $K_i > 0$ for each $i \in S$, and suppose the total required receiving rate at client j is $Q_j > 0$ for each $j \in C$. For the problem to be feasible, let us assume throughout this chapter,

$$\sum_{i \in S} K_i \geq \sum_{j \in C} Q_j. \tag{2–1}$$

In addition, let us suppose the route between each server-client pair is fixed, for instance, by the shortest path routing. Let μ denote the worst link utilization (i.e., the highest link utilization

over all links). Our objective is to minimize μ, which can also be viewed as network-wide load balancing. The complete problem is written as follows.

$$\textbf{Min-Congestion}: \min \quad \mu$$

$$\text{s.t.} \quad \sum_{p_{ij} \ni e} x_{ij} \leq c_e \mu, \quad \forall e \in E \tag{2–2}$$

$$\sum_{j \in C} x_{ij} \leq K_i, \quad \forall i \in S \tag{2–3}$$

$$\sum_{i \in S} x_{ij} = Q_j, \quad \forall j \in C \tag{2–4}$$

$$x_{ij} \geq 0, \quad \forall i \in S, \ \forall j \in C. \tag{2–5}$$

Condition (2–2) says that the link utilization (i.e., congestion level) at each link e, $(\sum_{p_{ij} \ni e} x_{ij})/c_e$, must be no greater than μ. Here, $\sum_{p_{ij} \ni e} x_{ij}$ is the aggregate bandwidth of all connections (paths) crossing link e. Condition (2–3) is the server capacity constraint: The aggregate sending rate out of server i cannot exceed its total capacity, K_i. For some applications such as media streaming, a minimum receiving rate is required. Condition (2–4) is the client required bandwidth constraint: The aggregate receiving rate at client j should satisfy the requested receiving rate, Q_j. There is no loss of generality to write $\sum_{i \in S} x_{ij} = Q_j$ instead of $\sum_{i \in S} x_{ij} \geq Q_j$ for the minimum rate requirement, since if an optimal solution exists for the Min-Congestion problem, there must exist an optimal solution for which the equality holds. Condition (2–4) may also be interpreted as requiring saturation of the clients' capacity or the downlink capacity at the clients' access links.

Remark 1: The problem formulation can be generalized to minimizing the sum of the link cost functions $\sum_{e \in E} h_e(\mu_e)$, where h_e is the cost function of link e and μ_e is the utilization of link e. Rather than overly emphasizing the most congested link, this formulation allows different degrees of emphasis on the congestion levels at different links. The objective function may also incorporate the server objectives, $-\sum_{i \in S} U_i(\sum_{j \in C} x_{ij})$, where U_i is server i's utility function. The solution algorithms in this chapter still apply after straightforward modification.

Remark 2: More generally, each client may have a different set of allowed servers, and each server may have a different set of allowed clients. In this case, we write S_j as the set of allowed

servers for client j; and we write C_i as the set of allowed clients for server i. (See Fig. 2-1 as an example.) The formulation can be modified accordingly and the structure of the solution algorithms remains the same.

Remark 3: Fig. 2-1 shows a moderately complex collaborative distribution example, where nodes may serve both as clients and servers to relay the content. The server-client dependency graph needs not be acyclic. For instance, we can add (directed) edges from node 6 to nodes 1 and 4. A swarming session such as those in BitTorrent typically corresponds to cyclic graphs. Our formulation and solutions apply to the cyclic case. But, side arrangement is needed to ensure that, in any cycle, the data from node a to b is not the same as the data from b back to a. They each should have different information content (e.g., different parts of the file).

The Min-Congestion problem is related to, but different from, the well-known maximum concurrent flow (MCF) problem [48]. Similar to the MCF problem, the Min-Congestion problem has an equivalent throughput-maximization formulation as follows. Let $\gamma = 1/\mu$, and $\tilde{x}_{ij} = x_{ij}/\mu$.

$$\begin{aligned}
\textbf{Max-Throughput}: \max \quad & \gamma \\
\text{s.t.} \quad & \textstyle\sum_{p_{ij} \ni e} \tilde{x}_{ij} \le c_e, \quad \forall e \in E \\
& \textstyle\sum_{j \in C} \tilde{x}_{ij} \le K_i \gamma, \quad \forall i \in S \\
& \textstyle\sum_{i \in S} \tilde{x}_{ij} = Q_j \gamma, \quad \forall j \in C \\
& \tilde{x}_{ij} \ge 0, \qquad \forall i \in S,\ \forall j \in C.
\end{aligned}$$

The maximization problem basically asks what the maximum scaling factor γ^* is, so that when the rates of the servers and clients are scaled by γ^*, none of the link capacity is exceeded. The variable γ is called the *throughput*. It is this interpretation, maximizing the throughout, that sometimes is a more important reason for our problem formulation, since maximizing the throughput is the same as minimizing the content distribution time.

2.2.2 Motivation: Performance Gain

To further motivate the above problem formulation, we first experimentally demonstrate the amount of performance gain that can be expected from our network optimization approach, as compared to existing heuristics. Random server assignment is an often used heuristic scheme, as discussed in the related work section (See Section 2.7.). In this scheme, each client selects a random number of servers. To have fair comparison, we also require that conditions (2–3) and (2–4) are satisfied. This complicates the actual sever-selection procedure a little. But the details are not essential.

Comparison with random server assignment and UDP. We first consider the case where UDP is the transport protocol. This is a realistic scenario. Practical systems such as digital fountain [49] combine source coding with UDP as the distribution mechanism for multicast content distribution, to avoid many technical difficulties of using TCP in the multicast situation. Our experiments are conducted on random networks. The precise model for the random networks is described in Section 2.6. We only point out that the usual transit-stub network model is inappropriate for our purpose, since we are dealing with content distribution networks or ISP networks instead of trying to capture the provider-customer network relationship of the Internet.

In the random server assignment scheme with UDP, each client requests the same flow rate from each randomly selected server. The random behavior of the system is simulated. The Min-Congestion problem is solved optimally using the Gnu Linear Programming Kit (glpk-4.9). We conduct experiments on a network with 1000 nodes, 16144 directed links, and 500 clients. We vary the number of servers from 100 to 500. The client side receiving rate is normalized to be $Q_j = 1.0$ for all $j \in C$, and all servers have the same K_i value.

Fig. 2-2 shows the link utilization μ at the most congested link for each scheme. We see that the μ of the random assignment is thousands times larger than the optimal one. Thus, it is very beneficial to apply the optimization solution. In fact, one can create network examples where the worst-case link congestion in the random assignment scheme is arbitrarily worse than the optimal scheme. The reason is that the random assignment scheme with UDP has no mechanism to cope

with network congestion, whereas the optimization scheme considers server assignment and network congestion jointly and optimally.

Comparison with random server assignment under TCP. In applications where TCP-styled congestion control can be used, the network congestion problem can be solved. To understand the resulting performance, we use the network simulator *ns-2* to simulate parallel download under TCP-Reno congestion control. We conduct experiments on a network with 50 nodes, 496 links, 10 servers and 40 clients. The server capacity and required receiving rate at each client are $K_i = 4.0$ and $Q_j = 1.0$, respectively, for all $i \in S$ and $j \in C$. In the *ns-2* simulation, a TCP connection is established between every server-client pair. We leave it to TCP to determine how much bandwidth is assigned to each server-client pair[3] .

Fig. 2-3 (a) shows that, when TCP is applied, the worst-case link utilization oscillates around 0.1, representing a factor of 30 increase over the optimal result, which is 0.0033. Under the optimal bandwidth allocation, the maximum throughput is $1/0.0033 = 300$, implying that the network can handle the traffic rate corresponding to $K_i = 1200$ and $Q_j = 300$. For the case with TCP, our experiments show that TCP at most can handle the traffic rate corresponding to $K_i = 400$ and $Q_j = 100$.[4] In conclusion, TCP ultimately overcomes congestion but the resulting throughput can be far worse than the optimal throughput.

2.3 Subgradient Method

In seeking suitable solutions to our models, our main objective is to find distributed algorithms, because they naturally become network control algorithms. One of the theoretical tools for this is convex separable optimization, as has been amply demonstrated by [31, 32, 34, 35, 41–43] for network control and resource allocation problems. A convex separable

[3] Although the network is lightly loaded, TCP still matters in determining the final bandwidth allocation of the server-client pairs.

[4] The interaction of TCP connections is complex when congestion occurs. The worst-case link utilization being equal to 0.1 for $K_i = 4$ and $Q_j = 1$ (light load) does not imply that TCP can handle no more than $K_i = 40$ and $Q_j = 10$.

optimization problem on $\Re^n$ has the form $\min_x \sum_{i=1}^n f_i(x_i)$, where f_i is a convex function for each i, subject to the linear constraints, $Ax = b$, where A is a matrix, $x = (x_1, ..., x_n)^T$, and b is a constant vector. Note that each f_i is a single-variable function of each individual x_i. Nonlinear convex separable objective function together with linear constraints makes it possible to decompose the optimization problem into distributed sub-problems. The subgradient algorithm yields a formal procedure for deriving distributed algorithms [47].

We will use the subgradient approach on the Min-Congestion problem, in a way similar to Madan and Lall [50]'s solution to a sensor network lifetime maximization problem. The idea is that the problem remains the same if the objective function is replaced by μ^2. We then have a nonlinear convex separable problem, and we expect the problem can be decomposed into a number of sub-problems, which can be carried out by each node distributively, hopefully using only local information or information from a small number of nodes. After some computation, it can be observed that, in the distributed algorithm, each node still needs to communicate with all other nodes for part of the computation. To make the algorithm completely distributed and decentralized, the problem is reformulated into the following problem.

For any $e = (k, l) \in E$, let N_e be the set of "neighboring" links of e, defined as follows.

$$\begin{aligned} N_e = &\{e' : e \text{ and } e' \text{ are consecutive links on some routing path}\} \\ &\cup \{e' = (k, l') \in E : \text{if } k \in S\} \cup \{e' = (k', l) \in E : \text{if } l \in C\}. \end{aligned}$$

$$\begin{aligned} \text{Primal: } \min \quad & \textstyle\sum_{e\in E} \mu_e^2 + \epsilon \sum_{i\in S} \sum_{j\in C} x_{ij}^2 \\ s.t. \quad & \textstyle\sum_{p_{ij}\ni e} x_{ij} \le c_e \mu_e, && \forall e \in E && (2\text{–}6) \\ & \textstyle\sum_{j\in C} x_{ij} \le K_i, && \forall i \in S && (2\text{–}7) \\ & \textstyle\sum_{i\in S} x_{ij} = Q_j, && \forall j \in C && (2\text{–}8) \\ & \mu_e = \mu_{e'}, && \forall e \in E, \forall e' \in N_e && (2\text{–}9) \end{aligned}$$

$$x_{ij} \geq 0, \quad \forall i \in S, \forall j \in C$$

$$\mu_e \geq 0, \quad \forall e \in E.$$

The extra constraint (2–9) and the definition of N_e guarantee that all μ_e are identical for all $e \in E$. The particular definition of N_e is to remove many redundant constraints of the form (2–9) and to make the resulting distributed algorithm simpler. The second term in the optimization objective is a regularization term. The new problem is identical to model 1 when $\epsilon = 0$.

To derive the distributed algorithm, the idea is to write down the Lagrange function and then find the dual function by minimizing the Lagrange function with respect to the primal variables. In essence, the subgradient algorithm is similar to the gradient (steepest ascent) algorithm for the dual problem. In general, the gradient for the dual problem does not necessarily exist. The subgradient is a generalization of the gradient and always exists. Part of the decomposition comes from the step of minimizing the Lagrange function. In our case, let the vectors w, α, β and v be the dual variables associated with the constraints (2–6), (2–7), (2–8) and (2–9), respectively. Let $L(x, \mu, \alpha, \beta, w, v)$ be the Lagrange function and let $g(\alpha, \beta, w, v)$ be the dual function. Then,

$$\begin{aligned} g(\alpha, \beta, w, v) &= \inf_{x \geq 0, \mu \geq 0} L(x, \mu, \alpha, \beta, w, v) \\ &= -\sum_{i \in S} K_i \alpha_i - \sum_{j \in C} Q_j \beta_j + \sum_{e \in E} \inf_{\mu_e \geq 0} (\mu_e^2 + (-c_e w_e - \sum_{e' \in N_e} v_{ee'} + \sum_{e' \in N_e} v_{e'e}) \mu_e) \\ &\quad + \sum_{i \in S} \sum_{j \in C} \inf_{x_{ij} \geq 0} (\epsilon x_{ij}^2 + (\alpha_i + \beta_j + \sum_{e \in p_{ij}} w_e) x_{ij}). \end{aligned} \tag{2–10}$$

The dual problem is,

$$\begin{aligned} \text{Dual: } \max \quad & g(\alpha, \beta, w, v) \\ s.t. \quad & w_e \geq 0, \quad \forall e \in E \\ & \alpha_i \geq 0, \quad \forall i \in S. \end{aligned}$$

Notice the sub-problems in (2–10) resulting from the decomposition. The solutions to the decomposed minimization problems will be written as follows.

$$x_{ij}^* = \text{argmin}_{x_{ij}\geq 0}\{\epsilon x_{ij}^2 + (\alpha_i + \beta_j + \sum_{e\in p_{ij}} w_e)x_{ij}\} \tag{2–11}$$

$$\mu_e^* = \text{argmin}_{\mu_e\geq 0}\{\mu_e^2 + (-c_e w_e - \sum_{e'\in N_e} v_{ee'} + \sum_{e'\in N_e} v_{e'e})\mu_e\}. \tag{2–12}$$

The subgradient algorithm is then applied to the dual problem. By a very general theory (Proposition 6.1.1 in [29]), the dual problem here is differentiable and the subgradient is equal to the gradient. Furthermore, the subgradient is related to the constraints of the primal problem. The components of the subgradient of g corresponding to w_e, α_i, β_j, $v_{ee'}$ for $e \in E$, $i \in S$, $j \in C$, and $e \in E, e' \in N_e$ are,

$$\frac{\partial g_e}{\partial w_e} = \sum_{p_{ij}\ni e} x_{ij}^* - c_e\mu_e^*, \tag{2–13}$$

$$\frac{\partial g_i}{\partial \alpha_i} = \sum_{j\in C} x_{ij}^* - K_i, \tag{2–14}$$

$$\frac{\partial g_j}{\partial \beta_j} = \sum_{i\in S} x_{ij}^* - Q_j, \tag{2–15}$$

$$\frac{\partial g_{ee'}}{\partial v_{ee'}} = -\mu_e^* + \mu_{e'}^*. \tag{2–16}$$

Finally, the actual subgradient algorithm at iteration step t is,

$$\begin{aligned}
x_{ij}^{(t)} &= x_{ij}^*(\alpha^{(t)}, \beta^{(t)}, w^{(t)}, v^{(t)}) \\
\mu_e^{(t)} &= \mu_e^*(\alpha^{(t)}, \beta^{(t)}, w^{(t)}, v^{(t)}) \\
w_e^{(t+1)} &= [w_e^{(t)} + \delta_t(\sum_{p_{ij}\ni e} x_{ij}^{(t)} - c_e\mu_e^{(t)})]_+ \\
\alpha_i^{(t+1)} &= [\alpha_i^{(t)} + \delta_t(\sum_{j\in C} x_{ij}^{(t)} - K_i)]_+ \\
\beta_i^{(t+1)} &= \beta_i^{(t)} + \delta_t(\sum_{i\in S} x_{ij}^{(t)} - Q_j) \\
v_{ee'}^{(t+1)} &= v_{ee'}^{(t)} + \delta_t(\mu_{e'}^{(t)} - \mu_e^{(t)}),
\end{aligned}$$

where $[\,]_+$ denotes the projection onto the non-negative orthant, and $\delta_t > 0$ is a scalar step size. Note that α_i and β_j can be computed by server i and client j, respectively, and $v_{ee'}$ can be computed by the node connecting edges e and e', all based on local information only. The link cost, w_e, can be computed based on the aggregate rate passing through link e and μ_e, both being local information. From (2–11), the path flow x_{ij} can be computed using the accumulated link costs along the path from server i to client j, and α_i, β_j from the server and client. This can be accomplished by an end-to-end feedback control protocol. From (2–12), the computation of μ_e needs only $\sum_{e' \in N_e} v_{ee'}$ and $\sum_{e' \in N_e} v_{e'e}$, where e' is a neighboring link of e. Hence, the subgradient algorithm is completely decentralized. As a final comment, according to (2–11), the rate x_{ij} is adjusted based on both the path congestion information and the costs that reflect the rate violation at the server and the client. Hence, the coupling of congestion control and server-client assignment is embodied in (2–11). In Section 2.6, the simulation results will show that the subgradient algorithm does not achieve good scalability and converges slowly compared with the gradient projection algorithm stated in the next section.

2.4 Barrier Method with Gradient Projection

In this and the next section, we show how to solve the Min-Congestion problem using distributed optimization algorithms, which naturally become network control algorithms. The solution is a special gradient projection algorithm, which has better convergence properties and simpler interpretation than the more often used subgradient algorithm.

2.4.1 Nonlinear Approximation of the Min-Congestion Problem with Barrier

We will first convert the Min-Congestion problem into a nonlinear convex separable form, using a similar approximation as in [50]. For each $e \in E$, let $\mu_e = \sum_{p_{ij} \ni e} x_{ij}/c_e$, which is the utilization of link e. Let $\vec{\mu}$ denote the vector (μ_e), $\forall e \in E$. Let $||\vec{\mu}||_\infty = \max_{e \in E} \mu_e$ be the max-norm, and $||\vec{\mu}||_q = (\sum_{e \in E} \mu_e^q)^{1/q}$ be the q-norm, for $q > 0$. The objective of minimizing the worst link congestion can be written as $\min ||\vec{\mu}||_\infty$. As $q \to \infty$, $||\vec{\mu}||_q \to ||\vec{\mu}||_\infty$. Hence, the objective of $\min ||\vec{\mu}||_\infty$ can be approximated by $\min ||\vec{\mu}||_q$, for some reasonably large q. The optimal solution under the latter objective is the same as that under $\min ||\vec{\mu}||_q^q$. We now have

converted the Min-Congestion problem into a nonlinear convex separable problem, to which the subgradient algorithm can be applied.

But, more is needed to apply the aforementioned specialized gradient projection algorithm. We write the server side capacity constraints in (2–3) as a barrier function. Then, the Min-Congestion problem can be approximated by the following problem.

$$\begin{aligned}
\min \quad & ||\vec{\mu}||_q^q - \epsilon \textstyle\sum_{i\in S} \ln(K_i - \sum_{j\in C} x_{ij}) && && \text{(2–17)}\\
\text{s.t.} \quad & \textstyle\sum_{p_{ij}\ni e} x_{ij} = c_e \mu_e, && \forall e \in E && \text{(2–18)}\\
& \textstyle\sum_{i\in S} x_{ij} = Q_j, && \forall j \in C && \text{(2–19)}\\
& x_{ij} \geq 0, && \forall i \in S,\ \forall j \in C.
\end{aligned}$$

The term in the objective function, $-\epsilon \sum_{i\in S} \ln(K_i - \sum_{j\in C} x_{ij})$, serves as a barrier function associated with the constraint $\sum_{j\in C} x_{ij} \leq K_i$, for $i \in S$. The parameter, $\epsilon > 0$, needs to be small enough for the approximation to be accurate.

Condition (2–18) is not really constraint but definition of what μ_e is in terms of (x_{ij}), which can be substituted into the objective function. The above problem now has only constraints of the type in (2–19) and the non-negativity constraint. Constraints of this type are called intersections of simplices. A convex problem on the intersection of simplices admits a special gradient projection algorithm, which will be explained.

We will next simplify the description of the problem for easy manipulation and will also remove certain technical difficulty. Let x denote the vector $(x_{ij}), \forall i \in S, \forall j \in C$ (or equivalently, (x_p), for any $p \in P$). For each link e, let y_e be the total traffic flow through e (i.e., $y_e = \sum_{p_{ij}\ni e} x_{ij}$). Let y denote the vector $(y_e), \forall e \in E$. In addition, define $z_i = \sum_{j\in C} x_{ij}$, which is the total sending rate from server i, and let z denote the vector $(z_i), \forall i \in S$. Denote by $\mathcal{X} \subset \Re^{|P|}$ the compact set of all feasible flow rates.

$$\mathcal{X} = \{x \mid x_{ij} \geq 0, \forall i \in S, \forall j \in C; \sum_{i\in S} x_{ij} = Q_j, \forall j \in C\}. \qquad \text{(2–20)}$$

Let G_1 and G_2 denote the $|E| \times |P|$ link-path incidence matrix and the $|S| \times |P|$ server-path incidence matrix associated with the given paths P. That is, $[G_1]_{ep} = 1$ if link e lies on path p, and $[G_1]_{ep} = 0$ otherwise; $[G_2]_{ip} = 1$ if server i uses path p, and $[G_2]_{ip} = 0$ otherwise. Hence, for any flow rate $x \in \mathcal{X}$, we have $y = G_1 x$ and $z = G_2 x$.

For each link $e \in E$, define the link cost functions in terms of path flow vector or the link flow rate, respectively, as

$$f_e^1(x) = (\frac{\sum_{p_{ij} \ni e} x_{ij}}{c_e})^q \tag{2–21}$$

and

$$\hat{f}_e^1(y_e) = (\frac{y_e}{c_e})^q. \tag{2–22}$$

Define the cost functions of the network in terms of the path flow vector or the link flow vector, respectively, as $f^1(x) = \sum_{e \in E} f_e^1(x)$ and $\hat{f}^1(y) = \sum_{e \in E} \hat{f}_e^1(y_e)$. The two cost functions are related by $f^1(x) = \hat{f}^1(y) = \hat{f}^1(G_1 x)$.

Next, consider each summand of the barrier function $\ln(K_i - z_i)$, where $z_i = \sum_{j \in C} x_{ij}$. The function $\ln(K_i - z_i)$ is only defined on the interval $[0, K_i)$. This creates some technical difficulty in developing an algorithm, since we need to worry about the feasibility of the flow rate vector with respect to the server capacity constraint during each step of the algorithm iteration. We will resolve this difficulty by extending the definition of the function $\ln(K_i - z_i)$ to the interval $[0, \infty)$, for all $i \in S$. For convenience, let us replace ϵ by ϵ^q. For any server $i \in S$, define the new barrier function in terms of the path flow vector or the server sending rate, respectively, as

$$f_i^2(x) = \begin{cases} -\epsilon^q \ln(K_i - \sum_{j \in C} x_{ij}) & \text{if } \sum_{j \in C} x_{ij} \leq K_i - \rho, \\ \epsilon^q((\sum_{j \in C} x_{ij} - (K_i - \rho) + \frac{1}{2\rho})^2 - (\frac{1}{4\rho^2} + \ln \rho)) & \text{if } \sum_{j \in C} x_{ij} > K_i - \rho, \end{cases} \tag{2–23}$$

and

$$\hat{f}_i^2(z_i) = \begin{cases} -\epsilon^q \ln(K_i - z_i) & \text{if } z_i \leq K_i - \rho, \\ \epsilon^q((z_i - (K_i - \rho) + \frac{1}{2\rho})^2 - (\frac{1}{4\rho^2} + \ln \rho)) & \text{if } z_i > K_i - \rho, \end{cases} \tag{2–24}$$

where $0 < \rho < 1$ is a small constant. The point is that $\hat{f}_i^2(z_i)$ thus defined is continuously differentiable on $[0, \infty)$. The overall barrier functions in terms of the path flow vector or the

sending rate vector are $f^2(x) = \sum_{i \in S} f_i^2(x)$ and $\hat{f}^2(z) = \sum_{i \in S} \hat{f}_i^2(z_i)$, respectively. The two functions are related by $f^2(x) = \hat{f}^2(z) = \hat{f}^2(G_2 x)$.

Finally, we can write the approximation to the Min-Congestion problem as follows, which we will solve distributedly.

$$\min \quad f(x) = f^1(x) + f^2(x) = \hat{f}^1(G_1 x) + \hat{f}^2(G_2 x) \tag{2–25}$$

$$\text{s.t.} \qquad x \in \mathcal{X}. \tag{2–26}$$

2.4.2 Gradient Projection Algorithm

For optimization problems with the simplex constraint of the form in (2–26), the optimality condition is especially simple [47]. Furthermore, there exists a special gradient projection algorithm [51] [52]. In this subsection, we describe the optimality condition and the algorithm. In the next subsection, we give the convergence results.

Useful First and Second Derivatives.

Throughout, we will assume $q \geq 2$ unless otherwise mentioned. We will need various first and second derivatives related to the function f. The derivative of f with respect to x_{ij} (the path flow rate) is given by

$$\frac{\partial f(x)}{\partial x_{ij}} = \sum_{e \in p_{ij}} (\hat{f}_e^1)'(\sum_{p_{kl} \ni e} x_{kl}) + (\hat{f}_i^2)'(\sum_{l \in C} x_{il}), \tag{2–27}$$

and the second derivative of f with respect to x_{ij} and x_{kl} is given by

$$\frac{\partial^2 f(x)}{\partial x_{ij} \partial x_{kl}} = \sum_{e \in p_{ij} \cap p_{kl}} (\hat{f}_e^1)''(\sum_{p_{kl} \ni e} x_{kl}) + \begin{cases} (\hat{f}_i^2)''(\sum_{l \in C} x_{il}) & \text{if } i = k \\ 0 & \text{otherwise.} \end{cases} \tag{2–28}$$

The following facts are important for the development of fully distributed algorithms.

1. $\frac{\partial f(x)}{\partial x_{ij}}$ is known as the first derivative cost of path p_{ij} at x.

2. As shown in (2–27), $\frac{\partial f(x)}{\partial x_{ij}}$ is equal to sum of the first derivative costs of the links on path p_{ij} plus the first derivative cost of server i. [5]
3. Each link cost and server cost can be computed locally by each link and server, using only local information.
4. To compute the cost of path p_{ij}, client j can collect all link costs on the path and the cost of server i, using an end-to-end control protocol.
5. In view of (2–28), similar statements as 2) – 4) can be said about the second derivative, $\frac{\partial^2 f(x)}{\partial^2 x_{ij}}$.

The computation of (2–27) and (2–28) requires the first and second derivatives of $\hat{f}_e^1$ and $\hat{f}_i^2$. The first derivatives of $\hat{f}_e^1$ and $\hat{f}_i^2$ are given by

$$(\hat{f}_e^1)'(y_e) = qc_e^{-q}y_e^{q-1}, \tag{2–29}$$

$$(\hat{f}_i^2)'(z_i) = \begin{cases} \frac{\epsilon^q}{K_i - z_i} & \text{if } z_i \leq K_i - \rho \\ 2\epsilon^q(z_i - (K_i - \rho) + \frac{1}{2\rho}) & \text{if } z_i > K_i - \rho. \end{cases} \tag{2–30}$$

The second derivatives of $\hat{f}_e^1$ and $\hat{f}_i^2$ are given by

$$(\hat{f}_e^1)''(y_e) = q(q-1)c_e^{-q}y_e^{q-2}, \tag{2–31}$$

$$(\hat{f}_i^2)''(z_i) = \begin{cases} \frac{\epsilon^q}{(K_i - z_i)^2} & \text{if } z_i \leq K_i - \rho \\ 2\epsilon^q & \text{if } z_i > K_i - \rho. \end{cases} \tag{2–32}$$

Optimality Condition.

Since the feasible set $\mathcal{X}$ is a convex set and the objective function is a convex function, we can characterize an optimal solution x^* to the problem (2–25)-(2–26) by the following optimality

[5] Subsequently, the cost of a path, a link or a server will refer to the first derivative cost, unless otherwise mentioned.

condition.

$$\sum_{j\in C}\sum_{i\in S}\frac{\partial f(x^*)}{\partial x_{ij}}(x_{ij}-x^*_{ij})\geq 0, \forall x\geq 0 \text{ with } \sum_{i\in S}x_{ij}=Q_j, \forall j\in C. \tag{2–33}$$

This condition is equivalent to, for any $i\in S$ and $j\in C$ (See [47].),

$$x^*_{ij}>0 \text{ only if } [\frac{\partial f(x^*)}{\partial x_{kj}}\geq \frac{\partial f(x^*)}{\partial x_{ij}}, \forall k\in S]. \tag{2–34}$$

Important insight is contained in (2–34). It says that, *in an optimal solution, a client j only selects those servers (i.e., receives positive traffic rates from them) that have the minimum path cost*. Here, the path for server i refers to the one from server i to client j (i.e., p_{ij}). The path cost under x refers to $\frac{\partial f(x)}{\partial x_{ij}}$, as given in (2–27), which contains both link costs for the links on the path and the server cost. Under x, for each client j, let us define the server that has the smallest path cost to client j by $s_j(x)$. That is,

$$s_j(x)=\text{argmin}_{i\in S}\{\frac{\partial f(x)}{\partial x_{ij}}\}. \tag{2–35}$$

If there are multiple such servers, an arbitrary one among them is chosen.

The Synchronous Gradient Projection Algorithm.

Based on a similar algorithm in [51], we apply the synchronous iterative gradient projection algorithm listed in Algorithm 1 to solve the problem (2–25)-(2–26).

Algorithm 1 Synchronous Gradient Projection Algorithm

$$x(t+1)=\alpha(t)\bar{x}(t)+(1-\alpha(t))x(t), \tag{2–36}$$

where, for all $j\in C$, $i\neq s_j(x(t))$,

$$\bar{x}_{ij}(t)=[x_{ij}(t)-\delta(\frac{\partial f(x(t))}{\partial x_{ij}(t)}-\frac{\partial f(x(t))}{\partial x_{s_j(x(t)),j}(t)})]_+ \tag{2–37}$$

and

$$\bar{x}_{s_j(x(t)),j}(t)=Q_j-\sum_{i\in S, i\neq s_j(x(t))}\bar{x}_{ij}(t). \tag{2–38}$$

Here, $\delta > 0$ is a scalar step size, and $\alpha(t)$ in (2–36) is a scalar on $[\underline{a}, 1]$, for some $\underline{a}$, $0 < \underline{a} \leq 1$. Given the current $x(t)$, (2–37) and (2–38) compute the end point, $\bar{x}(t)$, of a feasible direction, entry by entry. For each client j, there are two cases.

case 1 If a server i is not the chosen minimum-cost one (with the index $s_j(x(t))$), the flow rate on path p_{ij} will be reduced, unless it has zero flow already.

case 2 If a server i is the chosen minimum-cost one, the flow rate on p_{ij} is increased so that the total rates of all paths (from all servers) for client j is equal to the demanded rate Q_j.

Note that the description in case 2 ensures that $\bar{x}(t)$ is feasible (i.e., in $\mathcal{X}$). Since $x(t)$ is also feasible, by (2–36), the new rate vector $x(t+1)$, which is on the line segment between $x(t)$ and $\bar{x}(t)$, is feasible. Hence, if we start with a feasible rate vector $x(0)$ in $\mathcal{X}$, then $x(t)$ is in $\mathcal{X}$ for all t.[6]

Also note that server i and link e can compute the server cost $(\hat{f}_e^1)'(y_e)$ and the link cost $(\hat{f}_i^2)'(z_i)$ according to (2–29) and (2–30), respectively, all based on the local aggregate rate passing through the server or the link. The path cost, $\frac{\partial f}{\partial x_{ij}}$, can be computed by client j based on the link costs and the server cost along the path from server i to client j, according to (2–27). Hence, the gradient projection algorithm is completely decentralized.

2.4.3 Analysis of Convergence

In this subsection, we will adapt the results from [53] to find an upper bound on δ that guarantees the global convergence of the synchronous gradient projection algorithm to an optimal solution of the problem (2–25)-(2–26). Furthermore, when the sequence $\{x(t)\}$ gets near an optimal solution to which it converges, the convergence speed is linear (i.e., geometric). Some technical condition or minor reformulation of the problem are needed for some of these results to hold.

[6] (2–36) can be replaced with a more general update $x(t+1) = A(t)\bar{x}(t) + (I - A(t))x(t)$, where $A(t)$ is a $|S||C| \times |S||C|$ diagonal matrix with diagonal entries in the interval $[\underline{a}, 1]$, for some $\underline{a}$, $0 < \underline{a} \leq 1$. To ensure feasibility of $x(t+1)$ in $\mathcal{X}$, it is required that $\sum_{i \in S, j \in C} a_{ij}(t)(\bar{x}_{ij}(t) - x_{ij}(t)) = 0$, where $a_{ij}(t)$ is a corresponding diagonal entry of $A(t)$.

Assume that there is at least one feasible flow assignment. Define a set $\mathcal{X}_0 = \{x \in \mathcal{X} \mid f(x) \leq \eta\}$ for some scalar η such that $\mathcal{X}_0$ is nonempty (contains at least one feasible assignment). Since $\mathcal{X}_0$ is compact, f must attain a minimum on this set. Hence, there is an $x^* \in \mathcal{X}_0$ satisfying $f(x^*) = f^*$, where $f^* = \min_{x \in \mathcal{X}_0} f(x)$. We call any such x^* an optimal assignment, and we denote by $\mathcal{X}^*$ the set of optimal assignments. (There may be more than one optimal assignment since, although $\hat{f}^1$ and $\hat{f}^2$ are strictly convex, $f = f^1 + f^2$ is usually not.) That is,

$$\mathcal{X}^* = \{x \in \mathcal{X}_0 \mid f(x) = \min_{x \in \mathcal{X}_0} f(x)\}$$

We state the convergence results for algorithm (2–36) - (2–38). The main proofs are adapted from the proofs in [53]. More details about the proofs are given in Appendix 2.9.

Theorem 1 (Global Convergence)**.** *There is a $\delta_1 > 0$ such that for any δ, $0 < \delta < \delta_1$, every limit point of $\{x(t)\}$ generated by the synchronous gradient projection algorithm (2–36)-(2–38) with $x(0) \in \mathcal{X}_0$ is optimal.*

The constant δ_1 can be chosen as in Theorem 2. The proof is given in Appendix 2.9.

All our experimental results on random networks have shown that the algorithm actually converges to an optimal point and does so quite fast. We next show that, under additional conditions, the algorithm indeed converges to an optimal point. Furthermore, it converges linearly (i.e., geometrically) once the sequence $\{x(t)\}$ gets sufficiently near to an optimum.

- $A1$: At every optimum, the utilization of every link is strictly positive.

The condition $A1$ is not a stringent one. It can be naturally satisfied or forced to be satisfied. If a link is not used at all in the end, it probably would not have been deployed or activated in the first place. Alternatively, it may be easy to spot those links that will not get used in the final solution, possibly based on past knowledge. Then, these links can be excluded from the algorithm.

An important fact is that, under $A1$, the second derivative $(\hat{f}_e^1)''(y_e^*)$ is strictly positive for every $e \in E$ at any optimal x^* and $y^* = G_1 x^*$ (See (2–31).). As a result, the diagonal entries of $\nabla^2 \hat{f}^1$ are bounded away from zero by some positive scalar at an optimal x^*. By continuity, the

above is true in a neighborhood, say $\mathcal{N}$, near x^*. Furthermore, the diagonal entries of $\nabla^2 \hat{f}^2(G_2 x)$ are always bounded away from zero by some positive scalar for all $x \in \mathcal{X}_0$ (See (2–32).). We can then apply the result in [53] and conclude local geometric convergence in the neighborhood $\mathcal{N}$ of x^*, as well as global convergence. The precise statement is technical. We first need the following lemma.

Lemma 1. *Suppose $q > 2$ and $A1$ holds. There exists a scalar $\hat{\sigma}_1 > 0$ and a closed ball $\hat{\mathcal{X}} \subseteq \mathcal{X}_0$ around an optimal x^* such that*

$$\langle G_1 x^* - G_1 x, \nabla \hat{f}^1(G_1 x^*) - \nabla \hat{f}^1(G_1 x) \rangle \geq \hat{\sigma}_1 ||G_1 x^* - G_1 x||^2$$

for all $x \in \hat{\mathcal{X}}$. When $q = 2$, the above inequality holds for all $x \in \mathcal{X}$.

Proof. See Appendix 2.9. □

Define

$$F = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}. \tag{2–39}$$

Theorem 2 (Local Geometric Convergence Rate)**.** *Suppose $q > 2$ and $A1$ holds. Let δ satisfy $0 < \delta \leq \delta_1$. Suppose for some $t_0 \geq 0$, $x(t_0) \in \hat{\mathcal{X}}$. Then, the sequence $\{x(t)\}_{t \geq t_0}$ generated by the synchronous gradient projection algorithm (2–36)-(2–38) converges to an element of $\mathcal{X}^*$ and the convergence rate is linear (i.e., geometric) in the sense that for all $t \geq t_0$,*

$$f(x(t+1)) - f^* \leq (1 - D_5 \delta)(f(x(t)) - f^*). \tag{2–40}$$

Furthermore, when $q = 2$, the above conclusion holds all $t \geq 0$ with an initially feasible x.

The constants and parameters are as follows. $\delta_1 = \underline{a}/(L|S|)$, and $L > 0$ is the upper bound of the norm of $\nabla^2 f$ over $\mathcal{X}_0$; $D_5 = \underline{a}/(D_4 + \delta_1)$, $D_4 = ((5L+1)(D_3)^2 + 1 + 2\delta_1 + 6L(\delta_1)^2/\underline{a})|S|$ and $D_3 = D \max\{1, \delta_1\}$ for some $D > 0$. Moreover, D is bounded above by $D_1(D_1 + (\sqrt{|S|} + 1)\hat{L}||F^T||)/\hat{\sigma}$, where $D_1 = \max\{||H^{-1}|| \mid H$ an invertible submatrix of $F\}$. $\hat{L}_1$ is a positive scalar such that the diagonal entries of $\nabla^2 \hat{f}^1(G_1 x)$ are bounded above by $\hat{L}_1$ for all $x \in \mathcal{X}_0$. The positive scalar $\hat{\sigma}_1$ and the set $\hat{\mathcal{X}}$ are as defined in Lemma 1. $\hat{\sigma}_2 \leq \hat{L}_2$ are any two positive scalars

such that the diagonal entries of $\nabla^2 \hat{f}^2(G_2 x)$ *lie inside* $[\hat{\sigma}_2, \hat{L}_2]$ *for all* $x \in \mathcal{X}_0$. $\hat{\sigma} = \min\{\hat{\sigma}_1, \hat{\sigma}_2\}$ *and* $\hat{L} = \max\{\hat{L}_1, \hat{L}_2\}$.

Proof. The proof is outlined in Appendix 2.9. □

Theorem 3 (Global Convergence - Stronger Version)**.** *Suppose* $q > 2$ *and A*1 *holds. For* $0 < \delta < \delta_1$, *the sequence* $\{x(t)\}$ *generated by the synchronous gradient projection algorithm (2–36)-(2–38) with* $x(0) \in \mathcal{X}_0$ *converges to an optimal point.*

Proof. By Theorem 1, the sequence $\{x(t)\}$ will get near an optimal solution. By Theorem 2, once it gets sufficiently near that optimum, it converges to the optimum at a geometric speed. □

We suspect that the algorithm almost always enjoys *global* geometric convergence in random networks of reasonable size and traffic. However, one may not claim this theoretically. But, the original Min-Congestion problem can be approximated slightly differently, and it can be shown that the reformulated problem always enjoys global geometric convergence. Note that $\min ||\vec{\mu}||_\infty$ has the same solution as $\min ||\vec{\mu} + \vec{\kappa}||_\infty$, where $\kappa_e = \kappa$ for any constant $\kappa > 0$, $\forall e \in E$. We then consider the approximation of $\min ||\vec{\mu} + \vec{\kappa}||_\infty$. More precisely, we replace the term $||\vec{\mu}||_q^q$ by $\min ||\vec{\mu} + \vec{\kappa}||_\infty$ in the objective function in (2–17) and keep the same constraints. With this modification, $\hat{f}_e^1(y_e) = (\frac{y_e}{c_e} + \kappa)^q$ and $(\hat{f}_e^1)''(y_e) = q(q-1)(\frac{y_e}{c_e} + \kappa)^{q-2}$, for all $e \in E$. Then, the diagonal entries of $\nabla^2 \hat{f}^1(G_1 x)$ are bounded away from zero by some positive scalar for all $x \in \mathcal{X}_0$. We can then apply the result in [53] and conclude *global* geometric convergence.

Another possible reformulation is to define $\hat{f}_e^1(y_e) = \exp(\alpha y_e / c_e + \kappa)$ for some $\alpha > 0$ and $\kappa > 0$. The objective of minimizing $||\vec{\mu}||_\infty$ is approximated by minimizing $\sum_{e \in E} \exp(\alpha y_e / c_e + \kappa)$. Global geometric convergence can also be claimed.

2.5 Diagonally Scaled Algorithm

Geometric convergence in theory may still be slow for practical problems. As Section 2.6 will show, the gradient projection algorithm in (2–36) - (2–38) does not work well enough in terms of practical convergence speed when the network becomes large. In this section, we will discuss the cause of the problem and give a more scalable variant of the algorithm.

2.5.1 Ill-Conditioned Problem

When the power, q, is large, the problem (2–25)-(2–26) is ill-conditioned because of the poor relative scaling of the optimization variables. By this we mean that single unit changes of different variables have disproportionate effects on the cost. For instance, in the network of Fig. 2-4 (The numbers around the links are the link capacities.), the effect on the function f due to one unit increment of x_{11} can be very different from the effect due to one unit increment of x_{21}. This can be seen from the partial derivatives of f with respect to x_{11} or x_{21} in the following example. This type of situation corresponds to a large condition number of the Hessian of f. (See page 71 of [47] for a relevant discussion.)

Let us look at two particular iteration steps of the algorithm. Assume $K_i = 1.5$ for all $i \in S$ and $Q_j = 1.0$ for all $j \in C$. One of the optimal assignments of the Min-Congestion problem is $x = (1, 0.5, 0.05, 0, 0.5, 0.95)^T$. Here, the flow rates are first ordered according to the servers and then according to the clients. With $\epsilon = 1e-04$ and $\rho = 1e-05$ in (2–23) and (2–24), the step size upper bound δ_1 is about $5.6e+06$ (See Theorem 2.). Assume the flow assignment in the current iteration is $x = (0, 1, 0.5, 1, 0, 0.5)^T$. The most congested path is p_{21} and we need to decrease the flow on this path according to the algorithm in (2–36)-(2–38). The first derivatives of f with respect to x_{11} and x_{21} are 3.3×10^{-41} and 1.6×10^{-9}, respectively. Hence, by (2–37), at the current step, the change of the flow on p_{21} is about 0.009, which is reasonable in size. Now assume the current flow assignment is $x = (1, 0, 0.5, 0, 1, 0.5)^T$. The most congested path is p_{13}. The first derivatives of f with respect to x_{13} and x_{23} are 6.8×10^{-18} and 5.0×10^{-31}, respectively. Considering the link capacities, it is easy to see that path p_{13} is much more congested than p_{23} and we need to decrease the flow on p_{13}. But with the step size δ_1, the change of the flow on p_{13} is only 3.8×10^{-11}, which is too small to decrease x_{13} from 0.5 toward 0.05 substantially. Hence, the step size cannot lead to fast convergence for all cases of $x(t)$.

2.5.2 Diagonally Scaled Gradient Projection Algorithm

The ill-condition can be significantly alleviated by changing the units in which the optimization variables are expressed. This amounts to diagonal scaling of the variables and yields the

diagonally scaled gradient projection method as below (See [47].). Such scaling of the variables can also be interpreted as using different step sizes for different (unscaled) variables. The scaled algorithm is as follows.

$$x(t+1) = \alpha(t)\bar{x}(t) + (1-\alpha(t))x(t), \tag{2–41}$$

where for all $j \in C$, $i \neq s_j(x(t))$,

$$\bar{x}_{ij}(t) = [x_{ij}(t) - \delta \cdot d_{ij}^{-1}(t) \cdot (\frac{\partial f(x(t))}{\partial x_{ij}(t)} - \frac{\partial f(x(t))}{\partial x_{s_j(x(t)),j}(t)})]_+ \tag{2–42}$$

with

$$d_{ij}(t) = \frac{\partial^2 f(x(t))}{\partial x_{ij}^2(t)} + \frac{\partial^2 f(x(t))}{\partial x_{S_j(x(t)),j}^2(t)} - 2\frac{\partial^2 f(x(t))}{\partial x_{ij}(t)\partial x_{S_j(x(t)),j}(t)}$$

and

$$\bar{x}_{s_j(x(t)),j}(t) = Q_j - \sum_{i \in S, i \neq s_j(x(t))} \bar{x}_{ij}(t). \tag{2–43}$$

Here, $\frac{\partial f(x(t))}{\partial x_{ij}(t)}$, $\frac{\partial^2 f(x(t))}{\partial x_{ij}^2(t)}$ and $s_j(x(t))$ are given in (2–27), (2–28), and (2–35), and $\alpha(t)$ is as described for the non-scaled gradient projection algorithm in Section 2.4. The only change from Algorithm 1 is the scaling factor $d_{ij}^{-1}(t) = \frac{1}{d_{ij}(t)}$, which is motivated as a way to approximate the well-known Newton's algorithm. Note that the computation of d_{ij} is also decentralized. Each server i and link e can compute $(\hat{f}_e^1)''(y_e)$ and $(\hat{f}_i^2)''(z_i)$ according to (2–31) and (2–32), respectively, all based on the local aggregate rate passing through the server or the link. d_{ij} can be computed by client j based on the accumulated values along the paths from server i to client j, and from server $S_j(x)$ to client j according to (2–31) (2–32). Hence, the scaled algorithm is completely decentralized.

2.5.3 Asynchronous Algorithm

Time synchrony is usually difficult to maintain in a large network. In this section, an asynchronous version of the scaled gradient algorithm will be developed and its convergence result will be presented.

The development will capture two issues that may possibly make the asynchronous algorithm incorrect: the asynchronous operations of distributed network elements and delayed

measurement of data due to network delay. Specifically, the asynchronous algorithm differs from the synchronous algorithm in that each link, server and client is only required to make an update (iteration) at least once every B_1 time units, and the information used in the update may be outdated by as much as B_2 time units, where B_1 is a positive integer and B_2 is a nonnegative integer. In the case of $B_1 = 1$ and $B_2 = 0$, the asynchronous algorithm reduces to the synchronous algorithm. Our asynchronous model and the convergence result follow the approach in [54] [53].

In the asynchronous algorithm, we replace (2–42) with

$$\bar{x}_{ij}(t) = [x_{ij}(t) - \delta \cdot \hat{d}_{ij}^{-1}(t) \cdot (\lambda_{ij}(t) - \lambda_{s_j(x(t)),j}(t))]_+, \tag{2–44}$$

and set $\bar{x}_{s_j(x(t)),j}(t)$ according to (2–43), where for any client j, $s_j(x(t))$ satisfies $\lambda_{s_j(x(t)),j}(t) = \min_{i \in S} \lambda_{ij}(t)$. Here, $\lambda_{ij}(t)$ and $\hat{d}_{ij}(t)$ are some estimates of $\frac{\partial f}{\partial x_{ij}(t)}$ and $d_{ij}(t)$, respectively, which are, in general, inexact due to asynchrony and delays in obtaining measurements.

In correcting the assumption of perfect synchronization of computation, we only assume that the time between consecutive updates is bounded. More precisely, for each server i, client j and link e, let the subsets of updating times be $T^i \subseteq \{0, 1, 2, \cdots\}$, $T^j \subseteq \{0, 1, 2, \cdots\}$ and $T^e \subseteq \{0, 1, 2, \cdots\}$. Assume these subsets satisfy $\{t, t+1, \cdots, t+B_1-1\} \cap T^i \neq \emptyset$, $\{t, t+1, \cdots, t+B_1-1\} \cap T^j \neq \emptyset$ and $\{t, t+1, \cdots, t+B_1-1\} \cap T^e \neq \emptyset$ for all t. That is, for each client, server or link, at least one update occurs in every B_1 consecutive time slots.

We now describe the process by which $\lambda_{ij}(t)$ and $\hat{d}_{ij}(t)$ are formed, and the update is carried out. For each link e and each $t \in T^e$, we set $\lambda_e(t)$, the estimator of $(\hat{f}_e^1)'$, to be

$$\lambda_e(t) = \sum_{\tau=t-B_2}^{t} h_e(t,\tau)(\hat{f}_e^1)'(y_e(\tau)), \tag{2–45}$$

where, for every t, $h_e(t,\tau)$ are nonnegative coefficients summing to one. That is, $\lambda_e(t)$ is the weighted sum of the B_2+1 true measurements of $(\hat{f}_e^1)'(y_e(\tau))$ on the B_2+1 time slots at or prior to t. Hence, $\lambda_e(t)$ is a weighted moving average of $(\hat{f}_e^1)'(y_e(t))$. Similarly, $\hat{d}_e(t)$, the estimator of

$(\hat{f}_e^1)''$, is

$$\hat{d}_e(t) = \sum_{\tau=t-B_2}^{t} h_e(t,\tau)(\hat{f}_e^1)''(y_e(\tau)). \tag{2–46}$$

For each $t \notin T^e$, we set $\lambda_e(t) = \lambda_e(t-1)$ and $\hat{d}_e(t) = \hat{d}_e(t-1)$. That is, no update occurs on the time slots not in T^e.

Similarly, for each server i and each $t \in T^i$, we set $\lambda_i(t)$, the estimator of $(\hat{f}_i^2)'$, to be

$$\lambda_i(t) = \sum_{\tau=t-B_2}^{t} h_i(t,\tau)(\hat{f}_i^2)'(z_i(\tau)), \tag{2–47}$$

and $\hat{d}_i(t)$, the estimator of $(\hat{f}_i^2)''$, to be

$$\hat{d}_i(t) = \sum_{\tau=t-B_2}^{t} h_i(t,\tau)(\hat{f}_i^2)''(z_i(\tau)), \tag{2–48}$$

where, for each fixed t, $h_i(t,\tau)$ are nonnegative coefficients summing to one. For each $t \notin T^i$, we set $\lambda_i(t) = \lambda_i(t-1)$ and $\hat{d}_i(t) = \hat{d}_i(t-1)$.

In the above computations of the moving averages, $(\hat{f}_e^1)'$, $(\hat{f}_i^2)'$, $(\hat{f}_e^1)''$ and $(\hat{f}_i^2)''$ are given by (2–29) - (2–32).

At time $t \in T^j$, for each client j, $\lambda_{ij}(t)$ and $\hat{d}_{ij}(t)$ are naturally estimated by

$$\lambda_{ij}(t) = \sum_{e \in p_{ij}} \lambda_e(t) + \lambda_i(t), \tag{2–49}$$

and

$$\hat{d}_{ij}(t) = \hat{d}_i(t) + \hat{d}_{S_j(x(t))}(t) + \sum_{e \in \hat{p}_{ij}(t)} \hat{d}_e(t), \tag{2–50}$$

where

$$\hat{p}_{ij}(t) = (p_{ij} \cup p_{S_j(x(t)),j}) / (p_{ij} \cap p_{S_j(x(t)),j}).$$

$\lambda_{ij}(t)$ and $\hat{d}_{ij}(t)$ are each an aggregate of the measurements on the end-to-end path p_{ij} and the server i. There is no need to model the delay in collecting these measurements on the path, since the delay has already been modeled in each individual measurement itself in (2–45)-(2–48). For each $t \notin T^j$, we set $\lambda_{ij}(t) = \lambda_{ij}(t-1)$ and $\hat{d}_{ij}(t) = \hat{d}_{ij}(t-1)$.

Also at time $t \in T^j$, for each client j, $\bar{x}_{ij}(t)$ is updated according to (2–44) and (2–43). Given $x_{ij}(t)$ and $\bar{x}_{ij}(t)$ for all $i \in S$, client j derives the next flow assignment, $x_{ij}(t+1)$ for all $i \in S$, according to (2–41). At each $t \notin T^j$, we set $\bar{x}_{ij}(t) = \bar{x}_{ij}(t-1)$. In this case, $x(t+1) = x(t)$.

The asynchronous algorithm is summarized in Algorithm 2. We assume that there is an end-to-end control protocol operating on each path from a server to a client, if the path is carrying a positive flow. The function of this protocol is to carry the measurement information from each link and server to the relevant clients that use the link and server.

Algorithm 2 Asynchronous Scaled Gradient Projection Algorithm

Link e's algorithm:

- Link e measures the total traffic rate at the link on each time interval and keeps a memory of B_2 past measurements.
- At each update time $t \in T^e$, link e computes $\lambda_e(t)$ and $\hat{d}_e(t)$ according to (2–45) and (2–46), respectively.
- Link e participates in the control protocol and communicates $\lambda_e(t)$ and $\hat{d}_e(t)$ to the clients that use link e.

Server i's algorithm:

- Server i measures the total sending rate on each time interval and keeps a memory of B_2 past measurements.
- At each update time $t \in T^i$, server i computes $\lambda_i(t)$ and $\hat{d}_i(t)$ according to (2–47) and (2–48), respectively.
- Server i participates in the control protocol and communicates $\lambda_i(t)$ and $\hat{d}_i(t)$ to the clients that receive flow from it.

Client j's algorithm:

- At each update time $t \in T^j$, client j receives $\lambda_e(t)$ and $\hat{d}_e(t)$ from the links that it uses, and $\lambda_i(t)$ and $\hat{d}_i(t)$ from the servers that it uses, with the help of the control protocol.
- At each update time $t \in T^j$, client j computes $\lambda_{ij}(t)$ and $\hat{d}_{ij}(t)$ according to (2–49), (2–50), respectively. Then, it computes $\bar{x}_{ij}(t)$ for all $i \in S$ according to (2–44) and (2–43). Client j chooses new rates $x_{ij}(t+1)$ for all $i \in S$ according to (2–41) and communicates the changed flow rates to the servers.

Theorem 4. *(Convergence Result) Suppose the following conditions hold: The function $\hat{f}_e^1$ of each link $e \in E$ and the function $\hat{f}_i^2$ of each server $i \in S$ are defined on $[0, \infty)$, real valued (finite), convex and continuously differentiable; the derivatives of $\hat{f}_e^1$ and $\hat{f}_i^2$ are both Lipschitz continuous on any bounded interval; $\hat{d}_{ij}(t)$ is bounded below and above by some positive constants, for all $i \in S$ and $j \in C$, for all t; $\underline{a} > 0$; and the step size δ is chosen small enough.*

Then, $f(x(t))$ generated by the asynchronous diagonally scaled algorithm converges to f^* and any limit point of $\{x(t)\}$ is a minimizing point. Moreover, $x_{ij}(t) - \bar{x}_{ij}(t)$ converges to zero for all server-client pair i and j.

For problem (2–25)-(2–26), the conditions of Theorem 4 are all met. Hence, we have the following conclusion.

Corollary 1. *For problem (2–25)-(2–26), $f(x(t))$ generated by the asynchronous diagonally scaled algorithm converges to f^* and any limit point of $\{x(t)\}$ is a minimizing point. Moreover, $x_{ij}(t) - \bar{x}_{ij}(t)$ converges to zero for all server-client pair i and j.*

2.6 Performance Evaluation

In this section, we present experimental results that compare the convergence of the subgradient algorithm, and the gradient projection algorithm without and with scaling.

Since we are dealing with content distribution networks or ISP networks, we will not use the typical transit-stub network model, which is more suitable to capture the provider-customer network relationship in the Internet. Instead, all experiments in the chapter are conducted on the following class of uniform random networks. The parameters are the number of nodes n and the number of directed links m. Let $p = \frac{m/2}{n(n-1)/2}$ be the probability that a pair of nodes are connected. Then, for every pair of nodes, a Bernoulli trial with probability p is made. If the outcome of the trial is 1, then the two nodes are connected by two directed links in both directions. Otherwise, they are not connected. The number of connected pairs is $\text{Binomial}(\frac{n(n-1)}{2}, p)$ distributed, and its expected value is $m/2$. Hence, the expected number of directed links is m. If the resulting network is not connected, we repeat the same procedure. The capacity of each directed link is uniformly distributed on $[0.1, 1000]$. The servers and clients are randomly distributed over the network.

2.6.1 Gradient Projection vs. Subgradient Algorithm

The experiments are conducted on a randomly generated network with 50 nodes, 496 directed links, 10 servers and 40 clients. We set the client-side receiving rate to be $Q_j = 1.0$ for all $j \in C$, and set $K_i = 6.0$ for all $i \in S$.

In the subgradient algorithm, the step size δ is chosen manually. Fig. 2-5 shows that after about 600 iterations (not 150 iterations as it might appear in the figure, because the algorithm still needs to satisfy the uniform client side receiving rate as Fig. 2-6 (b) shows), μ converges to the optimal value. Fig. 2-6 (a) shows the average flow rates from the servers or to the clients. After about 50 iterations, they each become close to their final values, 4.0 and 1.0, respectively. This means that the rate of traffic entering and exiting the network at each node becomes stable quickly. From iteration step 50 to 150, the algorithm is devoted to balancing the link utilization. Fig. 2-6 (b) shows the standard deviation of the client rates. It is less than 6% of the average rate after 50 iterations. The algorithm spends about 600 iterations to make $Q_j = 1.0$ for each client j.

In the gradient projection algorithm, we choose constant values for ϵ, ρ and δ. In Fig. 2-7 (a), as q grows, the optimal value μ^* for the approximate problem (2–25)-(2–26) becomes increasingly close to the optimum of the Min-Congestion problem. Considering the scale of the vertical axis, the two optimal values are never too different for all $q \geq 2$. Again, the non-zero ϵ prevents them from becoming exactly the same. Fig. 2-7 (b) shows that, for $q = 10$, it takes 600 iterations for the objective value to converge, but only 100 iterations for it to come very close to the optimum. The convergence is faster than the subgradient algorithm. Note that unlike the subgradient algorithm, the gradient projection algorithm always guarantees that $Q_j = 1.0$ for each client j at each iteration.

2.6.2 Scaled vs. Unscaled Algorithm

The diagonally scaled gradient projection method is surprisingly effective for our problem. With the same network of 50 nodes, Fig. 2-8 (a) shows that the scaled algorithm converges to the optimal value after 350 iterations, which is slightly faster than the unscaled algorithm. However, the subgradient algorithm and the unscaled gradient projection algorithm do not work well for larger networks. We next conduct experiments on a slightly larger network with 100 nodes, 1606 links, 20 servers and 60 clients. We set $K_i = 4.5$ and $Q_j = 1.0$. As Fig. 2-9 (a) shows, the subgradient algorithm starts to stablize at iteration 200, but, towards a value $\mu = 0.00096$, which is about 2.0 times of the optimum of the Min-Congestion problem and cannot be improved by

much with more iterations. In Fig. 2-9 (b), the unscaled gradient projection algorithm produces a better solution than the subgradient algorithm, but costs 10000 iterations to get close to the optimum. In both algorithms, the poor performance is most likely due to the fact that the problem is ill-conditioned. As an evidence, when we apply the scaled projection algorithm to this network of 100 nodes, Fig. 2-8 (b) shows that it only takes 250 iterations to reach the optimum, which is very close to the optimum of the Min-Congestion problem.

The scaled algorithm achieves good convergence results for much larger networks. We conduct experiments on a network with 1000 nodes, 16036 links, 150 servers, and 750 clients. This network is much beyond what the subgradient algorithm or the unscaled gradient projection algorithm can solve. We set $K_i = 10.0$ for all $i \in S$, $Q_j = 1.0$ for all $j \in C$, and $q = 10$. Other parameters are chosen manually. Fig. 2-10 (a) shows that, after 2000 iterations, the algorithm converges nicely. In contrast, for the unscaled gradient algorithm or the subgradient algorithm, extensive degree of tuning of the step size still will not make the algorithm work for such a network size. It is also worth noting that this network is even challenging for the simplex method (for the linear Min-Congestion problem), which takes more than $44,000$ iterations and a lot of computing power to reach the optimum. Fig. 2-10 (b) shows that the barrier function $f^2(x)$ bounds the aggregate server sending rates very well, as it is supposed to. In the example of the figure, the sending rate is close to but below the maximum allowed sending rate, 10.

2.7 Additional Related Work

References to convex network optimization have been given throughout the chapter. This section discusses other related work.

Many P2P file sharing/distribution systems have been proposed, most of which are end-user based. In these systems, peers usually form a mesh network and exchange file chunks collaboratively. Examples of media-streaming systems include PROMISE [55], GridMedia [56], PRO [57], and PRM [58]. Examples of bulk data distribution systems include BitTorrent [1], FastReplica [59], SplitStream [60], Bullet′ [61], ChunkCast [62], and CoBlitz [63]. With a few exceptions, these P2P systems are mostly concerned with performance experienced by individual

nodes instead of network performance, whereas the central concern of this chapter is network performance.

The literature on server/node selection is vast. We will focus on recent works in the context of P2P or content distribution systems, which handle node selection in a variety of ways[7] . Nearly all of them are heuristic approaches. In SplitStream [60] and FastReplica [59], the selection is essentially random. Other systems employ a peer ranking function, and every node tries to select those peers with high ranking. A typical strategy to accomplish this is based on sampling: Each node initially selects some random peers, but dynamically probes other peers and gradually switches to those with better ranking over the course of transmission. BitTorrent [1], Bullet′ [61], ChunkCast [62] and CoBlitz [63] all use some form of this strategy. An often used ranking function is nodal load, such as in CoBlitz. Other ranking functions include the round-trip time (RTT) such as in ChunkCast, and the degree of content overlap such as in Bullet [64]. More relevant to this chapter, BitTorrent, Bullet′ and Slurpie [65] use the sending or receiving bandwidth to or from a peer as the ranking function. The performance of such a scheme is difficult to understand. But, it has been shown that BitTorrent works well for access-limited networks [66].

The next two are content distribution systems that pay attention to network performance. Julia [67] assumes the underlying P2P network is locality-aware. Each peer exchanges file chunks with neighbors in differential amount, more with closer neighbors. This reduces the total *work*, which is defined as the weighted sum of the total network traffic during the entire distribution process, where the weights are the distances travelled by bits. In [68], the server-to-client assignment is determined by minimizing the weighted cost of the total traffic during a media-streaming process under fixed server and client bandwidth constraints, where the weights

[7] Not all systems frame or handle this problem explicitly. But all should have at least an implicit selection algorithm.

are the RTT from the server to the client. This problem is similar to our problem, but the simplex method is used as the algorithmic solution, which is centralized.

In [69], several single-client server-selection problems are formulated under the optimization framework. The server capacity is the limited resource in these problems. In more traditional node-selection literature, [70] presents a dynamic selection scheme based on instantaneous measurement of the RTT and available bandwidth. Similar approaches are also reported in [71–73].

2.8 Conclusion

In this chapter, we attempt to improve key aspects of the collaborative distribution techniques so that they can be applied to content distribution over backbone or infrastructure networks. In this application, improvement in network performance is the main objective. Motivated by this objective, we formulate collaborative distribution as an optimization problem that minimizes the worst-case network congestion, or equivalently, maximizes the distribution throughput. Within the optimization framework, we jointly consider the server capacity constraint, the client receiving rate requirement, and the network capacity constraint. Our approach is different from most earlier P2P systems, which either do not consider the network constraint or do not make conscious bandwidth allocation. Our formulation allows us to derive the combined server-client assignment and congestion control (bandwidth allocation) algorithm. The optimization solutions naturally become network algorithms because they are fully distributed and decentralized.

There exist two classes of solutions to the optimization problem, the subgradient algorithm and a special gradient projection algorithm. The chapter focuses on developing the gradient projection algorithm, which is technical. We compare the two solutions carefully, mainly with respect to the convergence speed. The gradient projection algorithm achieves a locally linear convergence rate in theory; the simulation experiments have demonstrated that it converges faster than the subgradient algorithm in practice. However, neither of them works well enough on large networks due to the fact that our problem is ill-conditioned. Hence, we have developed a

diagonally scaled gradient projection algorithm, which achieves much better scalability. Both the synchronous and asynchronous algorithms are provided and the convergence results are stated.

Our approach makes the assumption that the network nodes can actually implement the functionalities required by the distributed algorithm. In practice, it is certainly difficult to deploy these functions at every network node, especially at the core routers. However, for an ISP network or a managed content distribution network, the deployment difficulty is drastically reduced since the network is usually much smaller (up to 10,000s of nodes instead of millions) and the network operator has the authority to deploy whatever features they desire to the network nodes. Furthermore, we can see at least three ways to make the deployment easier. First, the distributed algorithm may be implemented at a subset of the network nodes that are not the core routers, for instance, the gateway nodes of edge networks, or the nodes that interface with slower links. These nodes are more likely to become bandwidth bottlenecks; they are also more likely to have sufficient computing power to implement the algorithm, since they do not handle the largest traffic volume. Second, overlay servers can be attached to the network nodes where the algorithm needs to be implemented. The servers can implement the algorithm at the application layer on behalf of the network nodes. Third, if traffic is routed on the overlay network, as is usually the case for content distribution networks, the network nodes are themselves overlay nodes, which can implement the algorithm. Our algorithm can also be useful for specialized networks such as research and education networks and wireless mesh networks. It is easier to deploy new algorithms in these networks because they tend to be small, and hence, more upgradeable.

We can compare our server-selection criterion (based on the first-derivative path cost) with the metrics considered by other approaches discussed in Section 2.7. In our formulation, the capacity limitations of the servers and clients are naturally captured by the constraints of the optimization problem. Our formulation does not consider the RTT, which matters to the response time of short transactions but does not necessarily affect long-time average bandwidth, a much

more important factor for large content distribution[8] . If closer servers are more likely to lead to higher bandwidth, our formulation will tend to select closer servers for each client. In this chapter, we do not consider the degree of content overlap between the clients and servers during server selection. This is not needed if source coding is employed. Otherwise, how to incorporate the content overlapping aspect into the optimization formulation is an interesting problem.

The problem formulated in this chapter is still among the simplest. In the future, one may consider the problem with heterogeneous contents, possibly under environmental uncertainties. One may also investigate the case of multiple paths per server-client pair instead of the fixed shortest path routing. Finally, in the case of streaming content, future formulation may also need to take into account the timing and sequencing constraints.

2.9 Proof of Convergence Results for the Synchronous Algorithm (Algorithm 1)

We will show that under a strict convexity assumption on the functions $\hat{f}^1$ and $\hat{f}^2$, the sequence of flow assignments generated by the gradient projection algorithm (2–36) - (2–38) converges in the space of path flows to an element x^* of $\mathcal{X}^*$, the set of optimal flow rates, and achieves a local geometric convergence rate in the neighborhood of x^* (under some technical conditions).

The proof for the geometric convergence rate will closely follow [53]. One key difference is that, in [53], it requires that the diagonal entries of $\nabla^2 \hat{f}^1(G_1 x)$ and $\nabla^2 \hat{f}^2(G_2 x)$ are bounded away from zero by some positive scalars for all $x \in \mathcal{X}_0$. If this were true, [53] actually shows *global* geometric convergence. In our case, while $\nabla^2 \hat{f}^2(G_2 x)$ satisfies this requirement for all $x \in \mathcal{X}_0$, but $\nabla^2 \hat{f}^1(G_1 x)$ does not. However, under the condition $A1$ in Section 2.4.3, we can prove Lemma 1. The key is that the diagonal entries of $\nabla^2 \hat{f}^1(G_1 x)$ are bounded away from zero by some positive scalar in some neighborhood of an optimal x^*. With some more modification of the argument in [53], this will lead to the conclusion of a local geometric convergence rate.

[8] If TCP is used instead of our bandwidth allocation algorithm, then RTT does affect the achievable throughput.

Proof. (of Lemma 1) Let $y^* = G_1 x^*$, where x^* is an optimum, and let $y = G_1 x$. Assume $q > 2$. By $A1$, $\nabla^2 \hat{f}^1(y^*) > 0$. By continuity, there is a closed ball $\hat{\mathcal{X}} \subseteq \mathcal{X}_0$ around x^*, such that $\nabla^2 \hat{f}^1(G_1 x) > 0$ for all $x \in \hat{\mathcal{X}}$. Let $\hat{\sigma}_1 > 0$ be a minimum of $\nabla^2 \hat{f}^1(G_1 x)$ on $\hat{\mathcal{X}}$.

Let $\hat{\mathcal{Y}} = \{G_1 x \mid x \in \hat{\mathcal{X}}\}$. $\hat{\mathcal{Y}}$ is a closed and convex set since $\hat{\mathcal{X}}$ is closed and convex. Furthermore, $\nabla \hat{f}^1(y) \geq \hat{\sigma}_1$ for all $y \in \hat{\mathcal{Y}}$. Let $x \in \hat{\mathcal{X}}$. By the mean value theorem, there exists some $t \in [0, 1]$ such that

$$\begin{aligned} & \nabla \hat{f}^1(G_1 x^*) - \nabla \hat{f}^1(G_1 x) \\ = & \quad \nabla \hat{f}^1(y^*) - \nabla \hat{f}^1(y) \\ = & \quad \nabla^2 \hat{f}^1(ty^* + (1-t)y)^T (y^* - y). \end{aligned}$$

Since y and y^* are in the convex set $\hat{\mathcal{Y}}$, so is $ty^* + (1-t)y$. Therefore for all $x \in \hat{\mathcal{X}}$,

$$\begin{aligned} & \langle G_1 x^* - G_1 x, \nabla \hat{f}^1(G_1 x^*) - \nabla \hat{f}^1(G_1 x) \rangle \\ = & \quad \langle y^* - y, \nabla \hat{f}^1(y^*) - \nabla \hat{f}^1(y) \rangle \\ = & \quad \langle y^* - y, (\nabla^2 \hat{f}^1(ty^* + (1-t)y))^T (y^* - y) \rangle \\ \geq & \quad \textstyle\sum_{e \in E} \hat{\sigma}_1 (y_e^* - y_e)^2. \\ = & \quad \hat{\sigma}_1 ||G_1 x^* - G_1 x||^2. \end{aligned}$$

When $q = 2$, $\nabla^2 \hat{f}^1(y) = \text{diag}[\frac{q(q-1)}{c_1^q}, \frac{q(q-1)}{c_2^q}, \cdots, \frac{q(q-1)}{c_{|E|}^q}]$, for any $y = G_1 x$, where $x \in \mathcal{X}$. Let $\hat{\sigma}_1 = \min_{e \in E} \frac{q(q-1)}{c_e^q}$. We again have for all $x \in \mathcal{X}$,

$$\begin{aligned} & \langle G_1 x^* - G_1 x, \nabla \hat{f}^1(G_1 x^*) - \nabla \hat{f}^1(G_1 x) \rangle \\ \geq & \quad \hat{\sigma}_1 ||G_1 x^* - G_1 x||^2. \end{aligned}$$

□

Now we construct an expanded graph $\hat{G} = (\hat{N}, \hat{E})$ by adding some pseudo-links to the original graph G. At each server $i \in S$, attach one pseudo-node $\hat{n}_i$ with a directed pseudo-link $\hat{e}_i$ to server i. The capacity of this pseudo-link is K_i. Instead of thinking $\hat{f}_i^2(z_i)$ as a barrier function,

we regard $\hat{f}_i^2(z_i)$ as the link cost charged by each pseudo-link, where z_i is the link flow rate on the pseudo-link. Define the link cost function $\hat{f}_e$ on the expanded graph as

$$\hat{f}_e(\cdot) = \begin{cases} \hat{f}_e^1(\cdot) & \text{if } e \text{ is a physical link} \\ \hat{f}_i^2(\cdot) & \text{if } e \text{ is a pseudo-link directed to server } i. \end{cases}$$

and

$$\hat{f}(\cdot) = \sum_{e \in \hat{E}} \hat{f}_e(\cdot).$$

Denote the matrix $F = ((G_1)^T, (G_2)^T)^T$. The objective function in (2–25) can be written as

$$f(x) = \hat{f}^1(G_1 x) + \hat{f}^2(G_2 x) = \hat{f}(Fx).$$

Corollary 2. *When $q > 2$ and $A1$ holds, there exists a positive scalar $\hat{\sigma} = \min\{\hat{\sigma}_1, \hat{\sigma}_2\}$ and a closed ball $\hat{\mathcal{X}} \subseteq \mathcal{X}_0$ around an optimal x^* such that*

$$\langle Fx^* - Fx, \nabla \hat{f}(Fx^*) - \nabla \hat{f}(Fx) \rangle \geq \hat{\sigma} ||Fx^* - Fx||^2$$

for all $x \in \hat{\mathcal{X}}$. $\hat{\sigma}_1$ is a positive scalar defined in Lemma 1. $\hat{\sigma}_2$ is a positive scalar such that the diagonal entries of $\nabla^2 \hat{f}^2(G_2 x)$ are bounded below by $\hat{\sigma}_2$ for all $x \in \mathcal{X}$. When $q = 2$, the above inequality holds for all $x \in \mathcal{X}$.

For any feasible x, define $\phi(x) = \min_{x^* \in \mathcal{X}^*} ||x - x^*||$, and denote by $m(x)$ the vector in $\Re^{|P|}$ whose p^{th} component is $\min_{i' \in S} \frac{\partial f(x)}{\partial x_{i'j}}$ for all $p = P_{ij} \in P$.

Theorem 5. *Suppose $q > 2$ and suppose $A1$ holds. There exists a scalar $D > 0$ and a closed ball $\hat{\mathcal{X}} \subseteq \mathcal{X}_0$ around x^* such that*

$$\phi(x) \leq D||x - [x - \delta(\nabla f(x) - m(x))]_+|| / \min\{1, \delta\}$$

for all $\delta > 0$ and all $x \in \hat{\mathcal{X}}$. Moreover, D is bounded above by $D_1(D_1 + (\sqrt{|S|} + 1)\hat{L}||F^T||)/\hat{\sigma}$, $D_1 = \max\{||H^{-1}|| \mid H \text{ an invertible submatrix of } F\}$. $\hat{\sigma}$ is a positive scalar defined in Corollary 2. $\hat{L}$ is a positive scalar such that the diagonal entries of $\nabla^2 \hat{f}(Fx)$ are bounded above by $\hat{L}$ for all $x \in \hat{\mathcal{X}}$.

Proof. The proof basically follows [53]. [53] requires that the diagonal entries of $\nabla^2 \hat{f}(Fx)$ are bounded away from zero by some positive scalar for all $x \in \mathcal{X}_0$. It then uses this property of $\nabla^2 \hat{f}(Fx)$ to show that

$$\langle Fx^* - Fx, \nabla \hat{f}(Fx^*) - \nabla \hat{f}(Fx) \rangle \geq \hat{\sigma} ||Fx^* - Fx||^2$$

for all $x \in \mathcal{X}_0$. But, by Corollary 2, we are able to have the above inequality in a neighborhood x^*. The rest of the proof is nearly identical to that in [53]. □

The bounds $\hat{L}$ and $\hat{\sigma}$ are well defined for our specific $\hat{f}^1$ and $\hat{f}^2$ in the compact set $\hat{\mathcal{X}}$. Furthermore, if $q = 2$ or if we can modify our $\hat{f}^1$ to make the diagonal entries of its Hessian bounded below by a positive scalar for all $x \in \mathcal{X}_0$ (See the comment at the end of Section 2.4.3 on how to do this.), Theorem 5 will hold for all $x \in \mathcal{X}_0$ instead of only for x near an optimum.

Let $\delta_1 = \underline{a}/(L|S|)$, we have the following three lemmas. The only change from their counterparts in [53] involves substitution of appropriate constants.

Lemma 2. *For $0 < \delta \leq \delta_1$, we have for all t that $x(t) \in \mathcal{X}_0$ and*

$$f(x(t+1)) - f(x(t)) \leq -(\frac{\underline{a}}{\delta |S|} - \frac{L}{2}) ||x(t) - \bar{x}(t)||^2. \tag{2–51}$$

Lemma 3. *For $0 < \delta \leq \delta_1$, we have for all t that*

$$f(\bar{x}(t)) - f^* \leq \frac{5L+1}{2} \phi(x(t))^2 + (\frac{3L}{2} + \frac{1}{\delta} + \frac{1}{2\delta^2}) ||x(t) - \bar{x}(t)||^2. \tag{2–52}$$

Lemma 4. *For $0 < \delta \leq \delta_1$, we have for all t that*

$$f(x(t+1)) - f^* \leq \underline{a}[f(\bar{x}(t)) - f^*] + (1 - \underline{a})[f(x(t)) - f^*] + L(2 - \frac{\underline{a}}{2}) ||x(t) - \bar{x}(t)||^2. \tag{2–53}$$

Upon combining the preceding three lemmas and Theorem 5, Theorem 2 can be shown similarly as in [53].

Global Convergence without Condition $A1$.

We next prove Theorem 1.

Proof. From Lemma 2, for $0 < \delta \leq \delta_1$ and for all t that $x(t) \in \mathcal{X}_0$, we have

$$f(x(t+1)) - f(x(t)) \leq -(\frac{\underline{a}}{\delta|S|} - \frac{L}{2})||x(t) - \bar{x}(t)||^2.$$

With the constant stepsize $0 < \delta < \delta_1 = \underline{a}/L|S|$, the right-hand side of the above relation is nonpositive. Hence, if $\{x(t)\}$ has a limit point, the left-hand side tends to 0. The algorithm (2–36)-(2–38) can be denoted as a function $A(x)$ (i.e., $x(t+1) = A(x(t))$). Therefore, $||x(t) - \bar{x}(t)|| \to 0$, which implies that for every limit point $\tilde{x}$ of $\{x(t)\}$ we have $\tilde{x} = A(\tilde{x})$. It is easy to show that, if $\tilde{x} = A(\tilde{x})$, then, for every $i \in S$ and $j \in C$, we have

$$\tilde{x}^*_{ij} > 0 \text{ only if } [\frac{\partial f(\tilde{x})}{\partial x_{kj}} \geq \frac{\partial f(\tilde{x})}{\partial x_{ij}}, \forall k \in S]. \tag{2–54}$$

which is exactly the optimality condition in (2–34). (Proposition 2.3.2 and Example 2.1.2 in [47]). □

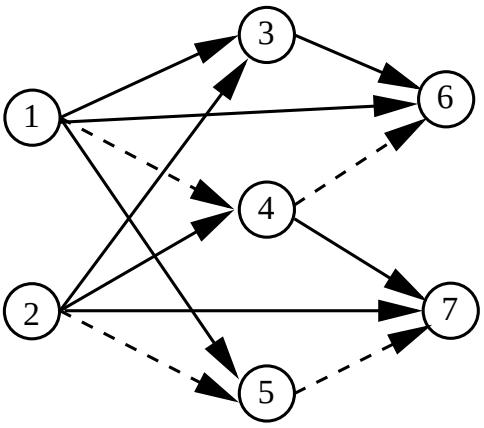


Figure 2-1. A server-client dependency graph. The solid and dashed lines indicate server-to-client relationship. The solid lines indicate the final server selection results. For instance, node 6 has the server set $S_6 = \{1, 3, 4\}$. It ends up selecting servers 1 and 3. Node 1 has the client set $C_1 = \{3, 4, 5, 6\}$.

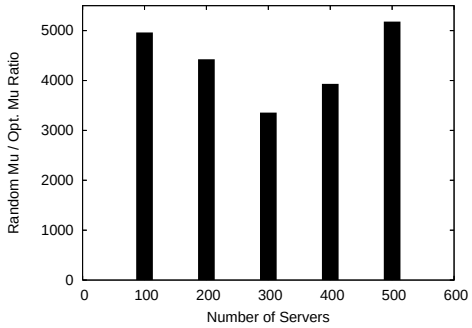
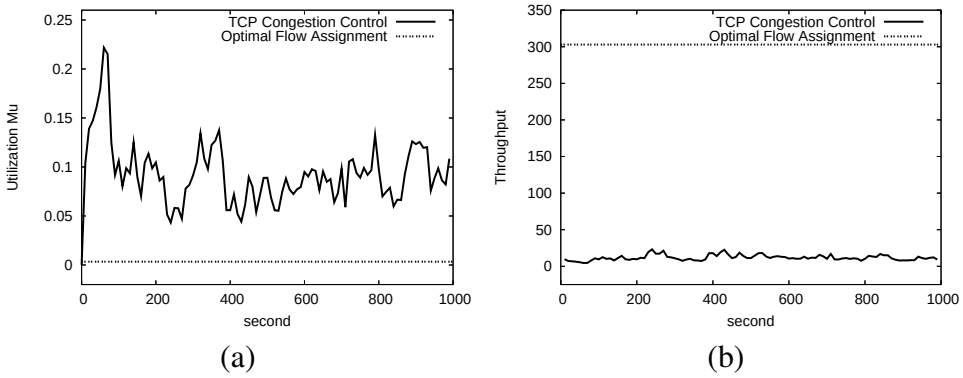


Figure 2-2. Ratio of worst-case link utilization: random vs. optimal flow assignment. The networks have 1000 nodes, 16144 links and 500 clients.



(a) (b)

Figure 2-3. Parallel download under TCP-Reno congestion control vs. optimal flow assignment. The networks have 50 nodes, 496 links, 10 servers and 40 clients. $K_i = 4.0$, $Q_j = 1.0$: (a) Worst-case link utilization; (b) Network throughput.

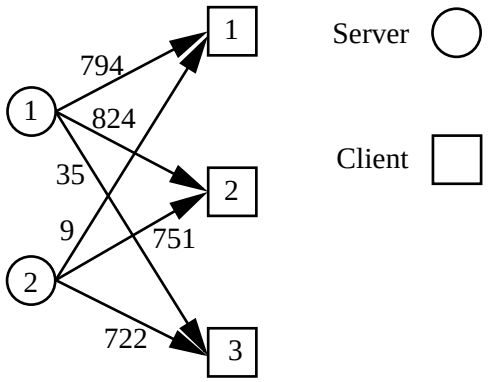


Figure 2-4. Network that leads to an ill-conditioned problem: $q = 10$.

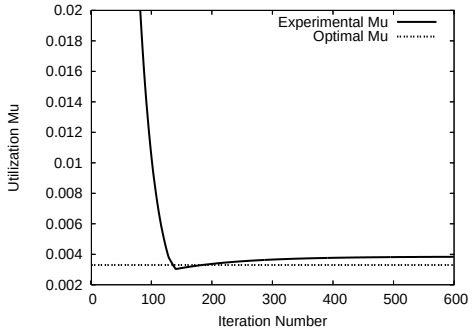
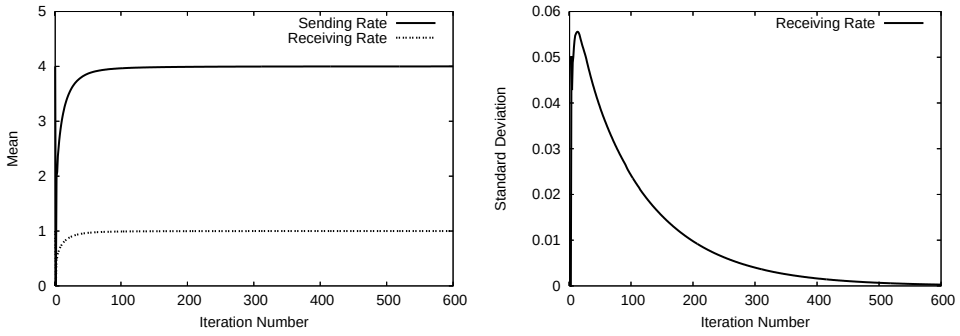
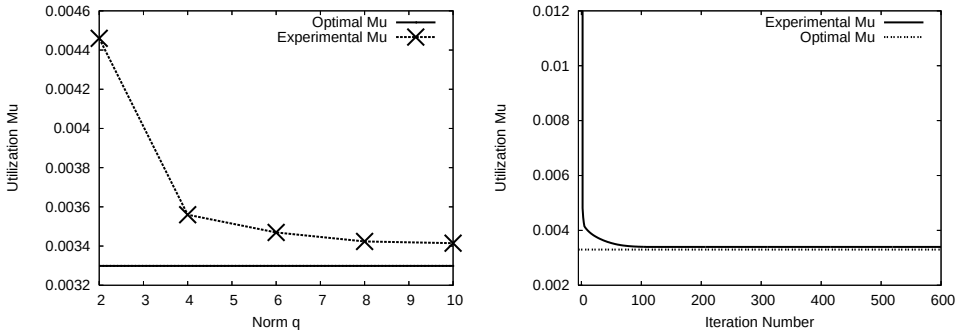


Figure 2-5. Convergence of the subgradient algorithm.



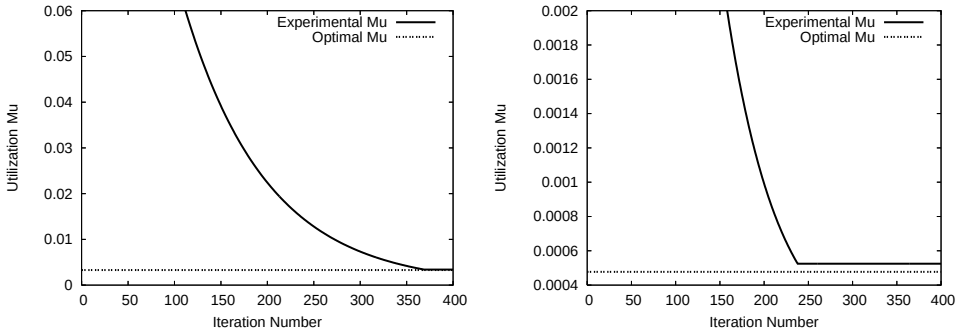
(a) (b)

Figure 2-6. Flow rate convergence of the subgradient algorithm: $K_i = 6.0, Q_j = 1.0$.



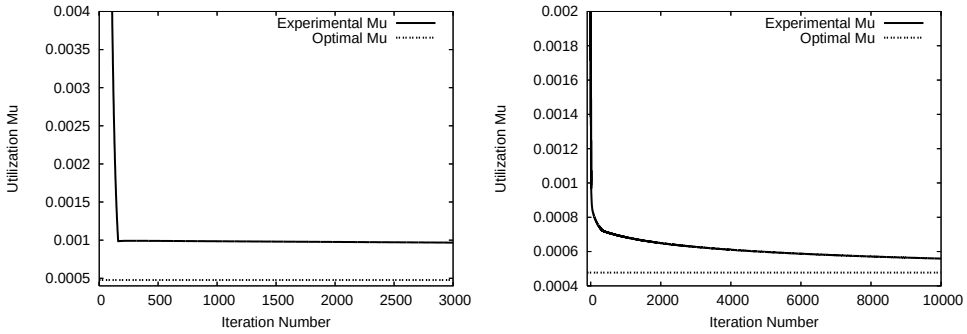
(a) (b)

Figure 2-7. Convergence of gradient projection algorithm: (a) Optimal utilization with different q; (b) Convergence of the algorithm. $q = 10$.



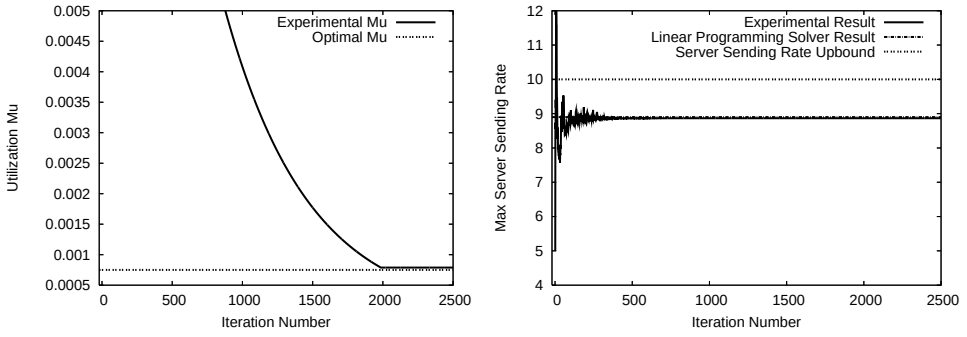
(a) (b)

Figure 2-8. Convergence of diagonally scaled gradient projection algorithm: (a) 50 nodes, 496 links, 10 servers and 40 clients. $q = 10$; (b) 100 nodes, 1606 links, 20 servers and 60 clients. $q = 10$.



(a) (b)

Figure 2-9. Poor performance of subgradient and unscaled gradient projection algorithms. 100 nodes, 1606 links, 20 servers and 60 clients. $K_i = 4.5$, $Q_j = 1.0$: (a) Subgradient algorithm; (b) Gradient projection algorithm. $q = 10$.



(a) (b)

Figure 2-10. Diagonally scaled gradient projection method. The network has 1000 nodes, 16036 links, 150 servers, 750 clients: (a) Convergence of the algorithm; (b) Converge of sending rate. $K_i = 10$.

CHAPTER 3
OPTIMAL PEER-TO-PEER TECHNIQUE FOR MASSIVE CONTENT DISTRIBUTION

3.1 Introduction

One of the distinct trends is that the Internet is being used to transfer content on a more and more massive scale. The grass root initiative for massive content distribution by the user and research community has been a peer-to-peer (P2P) networking technique known as *swarming*. In a swarming session, the file to be distributed is broken into many chunks at the original source, which are then spread out across the peers in a fashion that the sets of chunks at different peers are substantially different. Subsequently, the peers can exchange the chunks with each other to speed up the distribution process. The above description of swarming does not specify the precise manner at which the chunks are initially spread out or the manner at which the peers exchange them later. In fact, many different ways of swarming have been proposed, such as BitTorrent [1], FastReplica [59, 74], Bullet [61, 64], Chunkcast [62], CoBlitz [63], and Julia [67]. The most popular one among them is the BitTorrent protocol.

Swarming enables content providers with poor capacity to reach a large number of audience, allows rapid deployment of a large distribution system with minimum infrastructure support, and is increasingly used in mainstream and critical network services. On the negative side, swarming traffic has made capacity shortage in the backbone networks a genuine possibility, which will become more serious with fiber-based access[1] . As an example, with its early adoption of FTTH, by 2005, Japan already saw 62% of its backbone network traffic being from residential users to users, which was consumed by content downloading or P2P file sharing; the fiber users were responsible for 86% of the inbound traffic; and the traffic was rapidly increasing, by 45% that year [46].

[1] At the present, telecom companies are aggressively rolling out fiber-to-the home (FTTH) or its variants. Verizon is building a nationwide FTTH network in the US, to be completed by 2010. Japan had 5.6 million FTTH subscribers by June, 2006, is on an exponential upward ramp to have 30 million by 2010 [75]. The speed of the access fiber is currently at 100 Mbps or lower, heading to 1 Gbps by 2020 and is likely to reach 10 Gbps thereafter [76].

The problem addressed in this chapter is how to conduct massive content distribution efficiently in the future network environment where the capacity limitation can equally be at the core or the edge. The proposed solution is a class of improved swarming techniques, known as *optimal swarming*. In this chapter, swarming is viewed not just as a casual technique for end-user file-sharing applications. The benefits of swarming rest upon the fact it is an advanced form of networking mechanism, more advanced and more powerful than all previous distribution schemes, including IP or application-level multicast (e.g. [77]), network cache systems and existing content distribution networks (e.g., Akamai [78]). As will be seen, swarming can be thought as distributing content on *multiple* multicast trees. When done properly, it provides the most efficient utilization of the network capacity, a remedy for backbone congestion, or gives the fastest distribution.

For illustration of the main ideas in this chapter, consider the toy example in Fig. 3-1. The numbers associated with the links are their capacities. Suppose a large file is split into many chunks at source node 1. We wish to find the fastest way to distribute all chunks to receivers 2 and 3. [2] We impose no restriction on how peers can help each other in the distribution process. Let us focus on a fixed chunk and consider how it can be distributed to the receivers. With some thoughts, it can be argued that, when the delay is not modeled, the path should be a tree rooted at the source and covering both receivers. All possible distribution trees are shown in Fig. 3-2. The question becomes how to assign the chunks to different distribution trees so that the distribution time is minimized, subject to the link capacity constraint. For this simple example, it is easy to see that distributing the chunks in 1:2 ratio on the second and third tree, while leaving the first unused, is optimal.

The ideas contained in the toy example are also given in [79]. That work focuses on how to compute the maximum throughput, which leads to the fastest distribution. Our contribution in

[2] As we will show, the objective of minimizing the distribution time is equivalent to maximizing the distribution throughput, which is also equivalent to minimizing the network congestion.

this chapter is on developing distributed algorithms to identify and use the optimal distribution trees, and at the same time, allocate correct bandwidth on the selected trees. Furthermore, we introduce the column generation method, which introduces one more tree at a time and gradually expands the tree set. By combining column generation and the aforementioned tree packing algorithm, we in fact have a whole family of algorithms. On one side of the spectrum, we have a pure column generation algorithm; on the other side, we have a pure tree packing algorithm. In between, we have a mixed algorithm that introduces new extreme points at varying degree of frequency, thus balancing various aspects of the algorithm (e.g., performance and complexity).

Our approach illustrated by the toy example can be contrasted with existing P2P distribution systems or techniques [1, 59, 61–64, 67, 74]. Being originally designed for end-system file-sharing applications, the current swarming systems leave much room for improvement. Most systems only select and solve part of the problem and do so with poorly understood heuristics. Not enough is known on how well they work or how much improvement remains possible. They have user-centric performance objectives, but mostly ignore the important network-centric objectives, such as minimizing network congestion. The bandwidth bottleneck is often assumed at the access links, rather than throughout the network. Our solution will be able to automatically adapt to capacity constraint anywhere in the network. Furthermore, within our framework, we are able to solve four separate problems jointly and optimally: peer selection (from which peer to download), chunk selection (which chunks to download from a peer), distribution tree selection and bandwidth allocation. In contrast, other P2P systems solve one or two of these problems in isolation, typically using ad-hoc approaches.

The chapter is organized as follows. The models and problem formulations are given in Section 3.2. The distributed algorithm is given in Section 3.3. In Section 3.4, we present the column generation approach, combine it with the distributed algorithm described in Section 3.3. In Section 3.5, we discuss practical issues in applying our algorithm to realistic settings, such as scalability and coping with network dynamics and churn. In Section 3.6, we evaluate

the performance of our algorithm, including a comparison with BitTorrent and FastReplica. In Section 3.7, we discuss additional related work. The conclusion is drawn in Section 3.8.

3.2 Problem Description

We will start with a formulation for optimal content distribution on a generic network. It turns out the problem is difficult on an arbitrary network. However, for overlay content distribution, the problem is far easier. We will give formulations for two possible scenarios of overlay distribution.

3.2.1 Optimal Multicast Tree Packing

Let the network be represented by $G = (V, E)$, where V is the set of nodes and E is the set of links. The capacity associated with each link $e \in E$ is c_e. The utilization of link e, a measure of link congestion, is denoted by μ_e. We define a multicast session as a group of nodes (members) exchanging the same file. In a session, some members own some distinct chunks of a large file (The case of overlapping content at different nodes requires a minor extension, which will be discussed in Section 3.5.1.) and we call those members *sources* of the session. A reasonable assumption about a session is that all members in the session are interested in the file, and at the end of file distribution, every member in the session will have a complete copy of the file.

Let M be the set of all multicast sessions. For each session $m \in M$, let $V^{(m)} \subseteq V$ represent the set of members in session m, and let $S^{(m)} \subseteq V^{(m)}$ be the set of sources in session m. For each source $s \in S^{(m)}$, let $L_s^{(m)}$ be the total size of the file chunks at source s for session m. Let the set of all possible multicast trees spanning all members in the session rooted at source $s \in S^{(m)}$ be denoted by $T_s^{(m)}$. A multicast tree may contain nodes not in the session, in which case the tree is called a *Steiner tree*. In the case where all nodes on the tree belong to the session, the multicast tree is called a *spanning tree*, meaning it spans the multicast session (rather than the whole network V). For the i^{th} tree $t_{s,i}^{(m)} \in T_s^{(m)}$, where the order of indexing is arbitrary, denote $z_{s,i}^{(m)}$ to be the sending rate on tree $t_{s,i}^{(m)}$ from the root s.

A straightforward objective is to minimize the overall downloading time for all multicast sessions, which is to minimize the worst downloading time associated with any source s in any session m. With some thought, we see that no difference is made in terms of the achievable downloading time if we assume that all sources in all sessions finish their distribution at the same time. We can then minimize this common duration t. Let the total rate on a link $e \in E$ be denoted by x_e. It is equal to the sum of all the rates on all the trees passing through link e, across all sessions and all sources. That is,

$$x_e = \sum_{m \in M} \sum_{s \in S^{(m)}} \sum_{i: e \in t_{s,i}^{(m)}} z_{s,i}^{(m)}. \tag{3–1}$$

The optimization problem is as follows.

$$\min \quad t \tag{3–2}$$

$$\text{s.t.} \quad t \sum_{i=1}^{|T_s^{(m)}|} z_{s,i}^{(m)} = L_s^{(m)}, \ \ \forall s \in S^{(m)}, \ \forall m \in M \tag{3–3}$$

$$x_e \leq c_e, \quad \forall e \in E \tag{3–4}$$

$$z_{s,i}^{(m)} \geq 0, \quad \forall i = 1, \cdots, |T_s^{(m)}|, \forall s \in S^{(m)}, \forall m \in M. \tag{3–5}$$

Condition (3–3) says that, if one looks at all the multicast trees rooted at a source s for a session m, the sum of the distribution rates on all these trees, multiplied by the distribution time, should be equal to the total size of all the file chunks stored at source s for session m. This means that every bit of the file stored at s must be transmitted exactly once. A moment of thinking reveals that nothing is gained by transmitting the same bit more than once. Condition (3–4) is the link capacity constraint. At each link e, the flow rate on the link should be no greater than the link capacity, c_e.

It turns out the above problem is equivalent to a minimizing-congestion problem. This is immediate if we define $y_{s,i}^{(m)} = t z_{s,i}^{(m)}$ and make the substitution of variables. But, we will do this a little differently for ease of interpretation. Let $z_s^{(m)} = \sum_{i=1}^{|T_s^{(m)}|} z_{s,i}^{(m)}$ be the total sending rate at a source node s of session m. Select a set of constants $\{r_s^{(m)}\}$, each being proportional to $L_s^{(m)}$

with the same constant proportional factor. Each $r_s^{(m)}$ is understood as a rate. Consider a feasible solution $\{z_{s,i}^{(m)}\}$ and t. By (3–3), $z_s^{(m)}$ is proportional to $L_s^{(m)}$, the total size of the chunks at s for session m. Then, $z_s^{(m)} = \gamma r_s^{(m)}$, for some constant $\gamma > 0$. Next, define $\mu = 1/\gamma$. We then make the substitution of variables by $t = \frac{\mu L_s^{(m)}}{r_s^{(m)}}$, and redefine $z_{s,i}^{(m)}$ to be $\mu z_{s,i}^{(m)}$. We get the following minimizing-congestion formulation.

$$\min \quad \mu \tag{3–6}$$

$$\text{s.t.} \quad \sum_{i=1}^{|T_s^{(m)}|} z_{s,i}^{(m)} = r_s^{(m)}, \ \ \forall s \in S^{(m)}, \ \forall m \in M \tag{3–7}$$

$$x_e \leq \mu c_e, \quad \forall e \in E \tag{3–8}$$

$$z_{s,i}^{(m)} \geq 0, \quad \forall i = 1, \cdots, |T_s^{(m)}|, \forall s \in S^{(m)}, \forall m \in M. \tag{3–9}$$

In the above formulation, $r_s^{(m)}$ can be understood as the demanded rate, and μ is the maximum link utilization, which also measures the worst link congestion. The problem is to minimize the worst link congestion subject to the fulfillment of all demanded rates.

Thus, we have two equivalent views of optimal multicast tree packing. In the first view, the objective is to minimize the overall distribution time (or maximize the distribution throughput) while satisfying the link capacity constraint. In the second view, the objective is to best balance the network load while satisfying the rate demand for all sources and all sessions.

Another minor reformulation will be helpful later. Let μ_e stand for the utilization of link e, and let $\vec{\mu}$ denote the vector of μ_e over all links. Let $||\vec{\mu}||_\infty$ denote the maximum norm (i.e., $||\vec{\mu}||_\infty = \max_{e \in E} \mu_e$). The above minimizing-congestion formulation is equivalent to the following.

$$\min \quad ||\vec{\mu}||_\infty \tag{3–10}$$

$$\text{s.t.} \quad \sum_{i=1}^{|T_s^{(m)}|} z_{s,i}^{(m)} = r_s^{(m)}, \ \ \forall s \in S^{(m)}, \ \forall m \in M \tag{3–11}$$

$$x_e = \mu_e c_e, \quad \forall e \in E \tag{3–12}$$

$$z_{s,i}^{(m)} \geq 0, \quad \forall i = 1, \cdots, |T_s^{(m)}|, \forall s \in S^{(m)}, \forall m \in M. \tag{3–13}$$

The optimization problem proposed so far is equivalent to the problem of packing Steiner trees [80, 81], which is computationally intractable. Fortunately, for P2P content distribution, the problem becomes simpler. The main reason is that the overlay network of each session consists of exactly those nodes in the session. Given such an overlay network, any Steiner tree that covers all nodes of the session is in fact a spanning tree. We will show later that the algorithm used to solve the optimization problem involves a minimum-cost spanning tree problem in each iteration. Should some Steiner node exist, it would have involved a minimum-cost Steiner tree problem. The former is far more tractable than the latter NP-hard problem. One should be reminded that, although the computation for the overlay network case is far easier, the achievable performance is sub-optimal since the overlay edges are determined by the fixed underlay routing.

Unlike the inflexible underlay routing, the bandwidth of the overlay edge may have alternatives. Next, we consider two cases, both of which can be useful.

3.2.2 Fixed Overlay Link Bandwidth

In this case, the bandwidth of each overlay link is a fixed constant[3] determined by a bandwidth allocation scheme external to our problem. For instance, the bandwidth may be determined by the end-to-end TCP control, or by bandwidth allocation algorithms that enforce other allocation policies such as the max-min fairness [82]. We assume the overlay nodes know about the bandwidth on each overlay link. For instance, in the case of TCP, the overlay link bandwidth can be measured. When the overlay link bandwidth is fixed, different distribution sessions become decoupled. We then have $|M|$ totally independent overlay networks. The original optimization problem becomes separated to $|M|$ identical but independent optimization

[3] By fixed overlay bandwidth, we do not mean the bandwidth may not vary over time. We simply mean that the overlay link bandwidth is not determined by our algorithm and is known at the time of running the algorithm.

problems. The solution to this problem will require running algorithms only at the overlay nodes, making deployment easy.

We illustrate this by focusing on one of the overlay networks, which corresponds to one session. Let $\hat{G} = (\hat{V}, \hat{E})$ represent the overlay network. For all other notations, since there is no danger of confusing them with earlier definitions, we will re-define them. The bandwidth associated with each overlay link $e \in \hat{E}$ is c_e, which is allocated already and is a constant. The utilization of overlay link e is denoted by μ_e. Assume $S \subseteq \hat{V}$ is the set of sources. Let T_s represent the set of all possible (overlay) multicast trees rooted at source s spanning all overlay nodes. Let $t_{s,i} \in T_s$ be the i^{th} (overlay) multicast tree and $z_{s,i}$ be the associated sending rate on tree $t_{s,i}$. The rate on the overlay link $e \in \hat{E}$ is

$$x_e = \sum_{s \in S} \sum_{i: e \in t_{s,i}} z_{s,i} \tag{3–14}$$

The optimization problem is now

$$\begin{aligned} &\min ||\vec{\mu}||_\infty \\ &\text{s.t.} \sum_{i=1}^{|T_s|} z_{s,i} = r_s, \ \ \forall s \in S \\ &x_e = \mu_e c_e, \ \ \forall e \in \hat{E} \\ &z_{s,i} \geq 0, \ \ \forall i = 1, \cdots, |T_s|, \ \ \forall s \in S. \end{aligned} \tag{3–15}$$

3.2.3 Optimally Allocated Overlay Bandwidth

In this subsection, we consider an alternative scenario with better performance. Instead of relying on TCP to allocate the overlay bandwidth, leading to the partition of the overall network into multiple independent overlay networks, we will incorporate overlay bandwidth allocation into the optimization problem. Note that different sessions are coupled together by the sharing of the underlay links. The solution to this problem will require cooperation from the physical links. But, it is possible to modify the problem slightly and run the algorithm at only bottleneck links, such as the inter-ISP slow links.

Let $\hat{G}^{(m)} = (\hat{V}^{(m)}, \hat{E}^{(m)})$ represent the *overlay network* for each session m. For each overlay link $\hat{e} \in \hat{E}^{(m)}$, the notation $e \in \hat{e}$ for some underlay link $e \in E$ means that link e is on overlay link $\hat{e}$, which is itself an underlay path. For each session m, let $T_s^{(m)}$ be the set of all possible spanning trees on $\hat{G}^{(m)}$ rooted at source s , and $t_{s,i}^{(m)}$ be the i^{th} tree in $T_s^{(m)}$. Then, the total rate on a physical link $e \in E$, x_e, is given by

$$x_e = \sum_{m \in M} \sum_{s \in S^{(m)}} \sum_{\hat{e} \in \hat{E}^{(m)}: e \in \hat{e}} \sum_{i: \hat{e} \in t_{s,i}^{(m)}} z_{s,i}^{(m)}. \tag{3–16}$$

The optimization problem is exactly written as in (3–10)-(3–13).

3.3 Distributed Algorithm: Diagonally Scaled Gradient Projection

Note that $\min ||\vec{\mu}||_\infty$ has the same solution as $\min ||\vec{\mu} + \vec{\kappa}||_\infty$, where $\vec{\kappa} = (\kappa, \ldots, \kappa)$ for some small constant $\kappa \geq 0$. More precisely, we replace the objective function $||\vec{\mu}||_\infty$ by $||\vec{\mu} + \vec{\kappa}||_\infty$ in (3–10) and keep the same constraints. $\vec{\kappa}$ serves as a regularization term. The strictly positive vector $\vec{\kappa}$ (i.e., $\kappa > 0$) guarantees a global geometric convergence rate of our gradient projection algorithm; if $\vec{\kappa} = \vec{0}$, we can only claim that our gradient projection algorithm converges to one optimal solution globally.

3.3.1 Fixed Overlay Link Bandwidth

The goal of minimizing $||\vec{\mu} + \vec{\kappa}||_\infty$ is to balance the network load. The same objective can be achieved by minimizing $\sum_{e \in \hat{E}} \hat{f}_e(x_e)$, where $\hat{f}_e$ is some convex increasing function on $x_e \geq 0$. Such an objective function discourages large link rate. One such function is the q norm $||\vec{\mu} + \vec{\kappa}||_q = (\sum_{e \in \hat{E}} (\mu_e + \kappa)^q)^{1/q}$. In fact, $||\vec{\mu} + \vec{\kappa}||_\infty$ can be approximated by $||\vec{\mu} + \vec{\kappa}||_q$: as $q \to \infty$, $||\vec{\mu} + \vec{\kappa}||_q \to ||\vec{\mu} + \vec{\kappa}||_\infty$. We will assume $q \geq 2$ throughout. Since $\min ||\vec{\mu} + \vec{\kappa}||_q$ is equivalent to $\min ||\vec{\mu} + \vec{\kappa}||_q^q$, after substitution of μ_e with $\frac{x_e}{c_e}$, (3–15) becomes

$$\min \sum_{e \in \hat{E}} (\frac{x_e}{c_e} + \kappa)^q \tag{3–17}$$

$$\text{s.t.} \sum_{i=1}^{|T_s|} z_{s,i} = r_s, \quad \forall s \in S \tag{3–18}$$

$$z_{s,i} \geq 0, \quad \forall i = 1, \cdots, |T_s|, \quad \forall s \in S. \tag{3–19}$$

For optimization problems with the simplex constraint of the form (3–18), the optimality condition is especially simple [47]. It has been shown in [52] and [51] that there exists a special gradient projection algorithm. For our case, the gradient projection algorithm can also be easily extended to an equally simple scaled version. The latter overcomes the issue that our problem may be ill-conditioned, and hence, drastically improves the algorithm's convergence time. Our computational experiences have shown that the scaled gradient algorithm is much faster than the unscaled algorithm or the subgradient algorithm. The latter is often used in network optimization problems.

Another difficulty is the large number of possible spanning trees, and hence, the large number of variables. Fortunately, the algorithm does not maintain all possible spanning trees. The following steps take place for every source at the overlay network level. The algorithm starts out with one or few spanning trees. In each iteration, a cost is assigned to each (overlay) link to reflect the current link congestion. Then, a minimum-cost spanning tree can be computed. The algorithm shifts an appropriate amount of traffic (rate) from each currently maintained spanning tree to the minimum-cost tree. The new minimum-cost tree enters the current collection of spanning trees. Some previous spanning tree may leave the collection if its distribution rate is reduced to zero.

We next illustrate some details. Let $T = \bigcup_{s \in S} T_s$ be the collection of all multicast trees rooted at any source. Let z be the vector $(z_{s,i})$ where $s \in S, i = 1, \ldots, |T_s|$, with an arbitrary indexing order for the sources. In problem (3–17), let the feasible set defined by (3–18) and (3–19) be denoted by $\mathcal{Z}$.

For each overlay link $e \in \hat{E}$, recall that x_e is the aggregate flow rate it carries. Let x be the vector $(x_e)_{e \in \hat{E}}$. Let H denote the $|\hat{E}| \times |T|$ link-tree incidence matrix associated with the trees in T (i.e. $[H]_{et} = 1$ if link e lies on tree t; and $[H]_{et} = 0$ otherwise). Obviously, $x = Hz$. Now define $\hat{f}_e(x_e) = (x_e/c_e + \kappa)^q$ and $\hat{f}(x) = \sum_{e \in \hat{E}} \hat{f}_e(x_e)$. The objective function, denoted by $f(z)$, is given by

$$f(z) = \hat{f}(Hz) = \sum_{e \in \hat{E}} \hat{f}_e(x_e) = \sum_{e \in \hat{E}} (\frac{x_e}{c_e} + \kappa)^q, \tag{3–20}$$

and (3–17) can be written as

$$\begin{aligned} \min \quad & f(z) = \hat{f}(Hz) \\ \text{s.t.} \quad & z \in \mathcal{Z}. \end{aligned} \tag{3–21}$$

The derivative of the objective function $f(z)$ with respect to $z_{s,i}$ is given by

$$\frac{\partial f(z)}{\partial z_{s,i}} = \sum_{e \in t_{s,i}} \frac{\partial \hat{f}(x_e)}{\partial x_e} = \sum_{e \in t_{s,i}} \frac{q}{c_e} (\frac{x_e}{c_e} + \kappa)^{q-1}. \tag{3–22}$$

Note that $\frac{\partial \hat{f}(x_e)}{\partial x_e}$ is the so-called first-derivative link cost of link e [47, 51]. It reflects the current congestion level at link e. $\frac{\partial f(z)}{\partial z_{s,i}}$ is the first-derivative cost of the tree $t_{s,i}$, which is equal to the sum of the first-derivative costs of the links on the tree. It reflects the current congestion level of the tree $t_{s,i}$. The first-derivative tree cost is an important quantity. We will see later that our algorithm is to shift flows from trees with higher costs to the minimum-cost tree. The second derivative of $f(z)$ will be used in the scaling of the algorithm. With respect to $z_{s,i}$ and $z_{s,j}$, it is given by

$$\frac{\partial^2 f(z)}{\partial z_{s,i} \partial z_{s,j}} = \sum_{e \in t_{s,i} \cap t_{s,j}} \frac{q(q-1)}{c_e^2} (\frac{x_e}{c_e} + \kappa)^{q-2}. \tag{3–23}$$

For each $s \in S$, let i_s be the index of a minimum-cost tree rooted at s (i.e., with s as the source). That is,

$$i_s(z) = \text{argmin}_{\{i: t_{s,i} \in T_s\}} \{\frac{\partial f(z)}{\partial z_{s,i}}\}. \tag{3–24}$$

If there are multiple minimum-cost trees, we choose an arbitrary one. Since the feasible set $\mathcal{Z}$ is a convex set and the objective function is a convex function, we can characterize an optimal solution z^* to the problem (3–17) by the following optimality condition.

$$\sum_{s \in S} \sum_{i: t_{s,i} \in T_s} \frac{\partial f(z^*)}{\partial z_{s,i}} (z_{s,i} - z^*_{s,i}) \geq 0, \ \forall z \in \mathcal{Z}. \tag{3–25}$$

This optimality condition can be equivalently written as, for any source $s \in S$,

$$z^*_{s,i} > 0 \text{ only if } [\frac{\partial f(z^*)}{\partial z_{s,j}} \geq \frac{\partial f(z^*)}{\partial z_{s,i}}, \forall t_{s,j} \in T_s]. \tag{3–26}$$

That is, for every source, only those trees with the minimum first-derivative cost carry positive amount of flow. This intuitively suggests that, in the algorithm, we should shift flow to the minimum-cost trees from other trees.

It turns out this is exactly what the gradient projection algorithm does. We will develop the gradient projection algorithm following the proposal in [51] to solve the problem (3–17). But we will add diagonal scaling to speed up the algorithm's convergence time.

The equality constraint in (3–18) implies that, for each source, one of the variables depends completely on the rest of the variables. We can eliminate this variable and have a problem with fewer variables. To be concrete, at a feasible vector z, let's eliminate the variable z_{s,i_s} for each $s \in S$. Define a new objective function $g(\hat{z})$ on $\mathbb{R}^{|T|-|S|}$, where $\hat{z}$ consists of the remaining $z_{s,i}$'s after z_{s,i_s} is eliminated for each s. Without loss of generality, suppose, for each source s, i_s corresponds to the tree with the largest index (i.e., $i_s = |T_s|$). Also suppose the sources are indexed from 1 to $|S|$. Then,

$$\hat{z} = (z_{1,1}, \ldots, z_{1,|T_1|-1}; z_{2,1}, \ldots, z_{2,|T_2|-1}; \ldots; z_{|S|,1}, \ldots, z_{|S|,|T_{|S|}|}).$$

We will call the domain of g where $\hat{z}$ lies the reduced domain. We let

$$\begin{aligned} g(\hat{z}) = & f(z_{1,1}, \ldots, z_{1,|T_1|-1}, r_1 - \sum_{j=1}^{|T_1|-1} z_{1,j}; \\ & z_{2,1}, \ldots, z_{2,|T_2|-1}, r_2 - \sum_{j=1}^{|T_2|-1} z_{2,j}; \quad \ldots; \\ & z_{|S|,1}, \ldots, z_{|S|,|T_{|S|}-1|}, r_{|S|} - \sum_{j=1}^{|T_{|S|}|-1} z_{|S|,j}). \end{aligned}$$

The optimization problem in (3–17)- (3–19) is equivalent to

$$\min g(\hat{z}) \tag{3–27}$$

$$\text{s.t. } \hat{z} \geq 0. \tag{3–28}$$

This problem can be solved by the gradient projection algorithm.

$$\hat{z}(k+1) = [\hat{z}(k) - \delta(k)\nabla g(\hat{z}(k))]_+, \tag{3–29}$$

where $\delta(k)$ is a positive step size and $[\,]_+$ is the projection operator on $\hat{z} \geq 0$. In this case, $[y]_+$ just means that, if y_i is a component of y, we take $\max(y_i, 0)$ as the corresponding component of the vector $[y]_+$. The key is to compute $\nabla g(\hat{z}(k))$. Let $i_s(k)$ be a short hand for $i_s(z(k))$. It is easy to show, for $s \in S$ and $i \neq i_s(k)$,

$$\frac{\partial g(\hat{z}(k))}{\partial z_{s,i}} = \frac{\partial f(z(k))}{\partial z_{s,i}} - \frac{\partial f(z(k))}{\partial z_{s,i_s(k)}}. \tag{3–30}$$

The first derivatives are given in (3–22).

The algorithm in (3–29) is actually the constrained steepest-descent algorithm. It is well-known that the steepest-descent algorithm can be slow if the optimization problem is ill-conditioned. It happens that the minimizing-congestion type of network problems is often ill-conditioned. In our case, the problem becomes more ill-conditioned when the parameter q in the q norm becomes larger. An ultimate solution to an ill-conditioned problem is Newton's algorithm. However, Newton's algorithm is generally very complex since it requires the inverse of the Hessian matrix of the objective function. For large problems, this computation is generally impractical. We will next develop the diagonally scaled gradient algorithm, which is a much simpler alternative and a good approximation of Newton's algorithm. The scaled gradient projection algorithm can be written as

$$\hat{z}(k+1) = [\hat{z}(k) - \delta(k)D(k)\nabla g(\hat{z}(k))]_+, \tag{3–31}$$

where $D(k)$ is a positive definite matrix. For diagonal scaling, $D(k)$ is chosen to be a diagonal matrix.

$$D(k) = \text{diag}[(d_{s,i}(k))^{-1}]_{s\in S, i\neq i_s(k)}. \tag{3–32}$$

That is, the diagonal entry corresponding to $z_{s,i}(k)$ is chosen to be $(d_{s,i}(k))^{-1}$. For each $s \in S$ and $i \neq i_s(k)$, the value of $d_{s,i}(k)$ is chosen to be

$$d_{s,i}(k) = \frac{\partial^2 g(\hat{z}(k))}{\partial z_{s,i}^2}. \tag{3–33}$$

This way, the matrix $D(k)$ approximates the inverse of the Hessian of g at $\hat{z}(k)$. For each $s \in S$ and $i \neq i_s(k)$, the second derivative of g is further given by

$$\frac{\partial^2 g(\hat{z}(k))}{\partial z_{s,i}^2} = \frac{\partial^2 f(z(k))}{\partial z_{s,i}^2} + \frac{\partial^2 f(z(k))}{\partial z_{s,i_s(k)}^2} - 2\frac{\partial^2 f(z(k))}{\partial z_{s,i} \partial z_{s,i_s(k)}}. \tag{3–34}$$

The second derivatives are given in (3–23).

We can now collect different pieces of the development above and formally give the diagonally scaled gradient projection algorithm in the original domain where z lies. A slight generalization is present in (3–35).

Diagonally Scaled Gradient Projection Algorithm

$$z(k+1) = \alpha(k)\bar{z}(k) + (1 - \alpha(k))z(k) \tag{3–35}$$

$$\bar{z}_{s,i}(k) = \begin{cases} [z_{s,i}(k) - \delta(k) \cdot (d_{s,i}(k))^{-1} \cdot (\frac{\partial f(z(k))}{\partial z_{s,i}} - \frac{\partial f(z(k))}{\partial z_{s,i_s(k)}})]_+, & \text{if } i \neq i_s(k); \\ r_s - \sum_{1 \leq j \leq |T_s|, j \neq i_s(k)} \bar{z}_{s,j}(k), & \text{if } i = i_s(k), \end{cases} \tag{3–36}$$

with

$$d_{s,i}(k) = \sum_{e \in t_{s,i} \cup t_{s,i_s(k)} \setminus t_{s,i} \cap t_{s,i_s(k)}} \frac{q(q-1)}{c_e^2} (\frac{x_e(k)}{c_e} + \kappa)^{q-2}. \tag{3–37}$$

In (3–35), $\alpha(k)$ is a scalar on $[\underline{a}, 1]$, for some $\underline{a}$, $0 < \underline{a} \leq 1$. (3–35) says that the new rate vector at the $(i+1)^{th}$ iteration, $z(k+1)$, is on the line segment between $z(k)$ and $\bar{z}(k)$.

The main part of the algorithm is expression (3–36), which computes the end point of a feasible direction, $\bar{z}(k)$, entry by entry. There are three cases.

case 1: If a tree $t_{s,i}$ is not the chosen minimum-cost tree (with index $i_s(k)$) and $t_{s,i}$ has a positive flow, its rate will be reduced (more precisely, if $t_{s,i}$ is a minimum-cost tree with positive flow rate but not the chosen minimum-cost tree, its rate will keep the same).

case 2: If a tree $t_{s,i}$ is not the chosen minimum-cost tree and the tree has zero flow rate, then the rate stays at 0.

case 3: If $t_{s,i}$ is the chosen minimum-cost tree, the rate of the tree is increased so that the total rates of all trees rooted at s will be equal to the demanded rate r_s.

Note that the description in case 3 ensures that $\bar{z}(k)$ is feasible (in $\mathcal{Z}$). Since $z(k)$ is also feasible, by (3–35), the new rate vector $z(k+1)$ is feasible. Hence, if we start with a feasible solution in $\mathcal{Z}$, $z(k)$ is in $\mathcal{Z}$ for all k. [4]

What remains to be said is how much the rate is reduced in case 1. Note that the expression $\frac{\partial f(z(k))}{\partial z_{s,i}} - \frac{\partial f(z(k))}{\partial z_{s,i_S(k)}}$ is the difference in the first-derivative cost between the tree $t_{s,i}$ and the chosen minimum-cost tree, and the difference is always non-negative. Intuitively, the amount of reduction should be proportional to this difference. Indeed, if we ignore the factor $(d_{s,i}(k))^{-1}$ in (3–36), the rate reduction is proportional to the cost difference with the proportional constant (step size) $\delta_s(k) > 0$.

The factor $(d_{s,i}(k))^{-1}$ does the diagonal scaling, which can effectively deal with our ill-conditioned problem. The scaling factor $(d_{s,i}(k))^{-1}$ can be understood as allowing different components of the vector z to use different step sizes. Note that, in the expression for $d_{s,i}(k)$ in (3–37), which corresponds to the i^{th} tree, the sum is over the non-overlapping links between the i^{th} tree and the $i_s(k)^{th}$ tree, the latter being the minimum-cost tree.

The algorithm in (3–35)-(3–37) is a distributed one. In order to compute the tree cost, $\frac{\partial f(z)}{\partial z_{s,i}}$ in (3–22), and the scaling factor, $(d_{s,i}(k))^{-1}$ in (3–37), each link e can independently compute its corresponding term based on the local aggregate rate, x_e, passing through the link. Then, the tree cost and the scaling factor can be accumulated by the source s based on the link values along the tree. To find the minimum-cost tree $i_s(k)$, each source needs to compute the minimum-cost

[4] (3–35) can be replaced with a more general update $z(k+1) = A(k)\bar{z}(k) + (I - z(k))z(k)$, where $A(k)$ is a $\sum_{s\in S}|T_s| \times \sum_{s\in S}|T_s|$ diagonal matrix with diagonal entries in the interval $[\underline{a}, 1]$, for some $\underline{a}$, $0 < \underline{a} \leq 1$. To ensure feasibility of $z(k+1)$ in $\mathcal{Z}$, it is required that $\sum_{1\leq i\leq|T_s|} a_{s,i}(k)(\bar{z}_{s,i}(k) - z_{s,i}(k)) = 0$, where $a_{s,i}(k)$ is a corresponding diagonal entry of $A(k)$.

directed spanning tree (MDSP). Both centralized and distributed algorithms exist for computing the MDSP [83] [84] [85]. Both achieve $O(n^2)$ time complexity for a complete graph with n nodes. In the distributed version, the amount of information exchanged is also $O(n^2)$. In our implementation, each source collects the (overlay) link costs from all the receivers and uses a centralized algorithm to compute the MDSP. Other than that, the gradient algorithm is completely decentralized.

In addition to fast convergence, another strength of this gradient algorithm lies in that it avoids the enumeration of all possible spanning trees. The source only needs to manage the set of active multicast trees (i.e., those trees with positive flows). At each iteration, the source computes a new minimum-cost tree. A non-active tree will not become active unless it is the minimum-cost tree. The source only adjusts the flow rates among the set of active trees. The set of active trees usually is not very large if the algorithm converges fast, since, at each iteration, at most one more tree becomes active. For the original linear model (3–15), there are at most $|\hat{E}| + |S|$ active trees in any extreme point solution. But since this gradient algorithm is a kind of interior point method, strictly speaking, $|\hat{E}| + |S|$ is not really an upper bound. Nevertheless, it should give a rough sense on what the bound might be.

We stress that the reason to apply the scaling factor is to counter the ill-conditioned problem when q is large. In an ill-conditioned problem, single-unit changes of different variables have disproportionate effects on the cost (e.g., objective) change. For convergence, the step size in the iterative algorithm must be tuned according to the variables that cause large cost changes. However, such a step size can be too small for other variables, and as a result, they hardly change from iteration to iteration. The diagonally scaled algorithm essentially re-scales the variables so that single-unit changes in the scaled variables have similar effect on the cost objective. For our problem, the scaling has the simple interpretation that different trees use different step sizes, each roughly being proportional to a power of the worst link utilization on the tree. The resulting scaled algorithm is far superior to the plain gradient projection algorithm.

3.3.2 Optimally Allocated Overlay Bandwidth

The problem described in Section 3.2.3 can be worked out in a similar way, leading to a scaled gradient projection algorithm. Substitute μ_e with $\frac{x_e}{c_e}$, the problem becomes

$$\min \quad \sum_{e\in E}(\frac{x_e}{c_e}+\kappa)^q \tag{3–38}$$

$$\text{s.t.} \quad \sum_{i=1}^{|T_s^{(m)}|} z_{s,i}^{(m)} = r_s^{(m)}, \ \ \forall s \in S^{(m)}, \ \forall m \in M \tag{3–39}$$

$$z_{s,i}^{(m)} \geq 0, \quad \forall i = 1, \cdots, |T_s^{(m)}|, \forall s \in S^{(m)}, \forall m \in M. \tag{3–40}$$

Let $T = \bigcup_{m\in M}^{s\in S^{(m)}} T_s^{(m)}$ be the collection of all multicast trees rooted at any source s for any session m. Let z be the vector $(z_{s,i}^{(m)})$ where $s \in S^{(m)}, m \in M, i = 1, \ldots, |T_s^{(m)}|$, with an arbitrary indexing order for the sources. In problem (3–38)-(3–40), let the feasible set defined by (3–39) and (3–40) be denoted by $\mathcal{Z}$.

Let $\hat{E} = \bigcup_{m\in M} \hat{E}^{(m)}$ be the collection of all overlay links in all sessions. Let $\hat{H}$ denote the $|\hat{E}| \times |T|$ overlay link-tree incidence matrix associated with the trees in T (i.e. $[\hat{H}]_{\hat{e}t} = 1$ if overlay link $\hat{e}$ lies on tree t; and $[\hat{H}]_{\hat{e}t} = 0$ otherwise). Recall E is the set of underlay links, let H denote the $|E| \times |\hat{E}|$ underlay link-overlay link incidence matrix associated with the overlay links in $\hat{E}$ (i.e. $[H]_{e\hat{e}} = 1$ if underlay link e lies on overlay link (underlay path) $\hat{e}$; and $[H]_{e\hat{e}} = 0$ otherwise).

For each underlay link $e \in E$, recall that x_e is the aggregate flow rate it carries. Let x be the vector $(x_e)_{e\in E}$. It is easy to see $x = H\hat{H}z$. Note that an underlay link e might carry multiple copies of the same file chunk distributed by one tree t. Define $\hat{f}_e(x_e) = (x_e/c_e + \kappa)^q$ and $\hat{f}(x) = \sum_{e\in E} \hat{f}_e(x_e)$. The objective function, denoted by $f(z)$, is given by

$$f(z) = \hat{f}(H\hat{H}z) = \sum_{e\in E} \hat{f}_e(x_e) = \sum_{e\in E}(\frac{x_e}{c_e}+\kappa)^q, \tag{3–41}$$

and (3–38) can be written as

$$\begin{aligned} \min \quad & f(z) = \hat{f}(H\hat{H}z) \\ \text{s.t.} \quad & z \in \mathcal{Z}. \end{aligned} \tag{3–42}$$

The derivative of the objective function $f(z)$ with respect to $z_{s,i}^{(m)}$ is given by

$$\frac{\partial f(z)}{\partial z_{s,i}^{(m)}} = \sum_{\hat{e} \in t_{s,i}^{(m)}} \sum_{e \in \hat{e}} \frac{\partial \hat{f}(x_e)}{\partial x_e} = \sum_{\hat{e} \in t_{s,i}^{(m)}} \sum_{e \in \hat{e}} \frac{q}{c_e} (\frac{x_e}{c_e} + \kappa)^{q-1}. \tag{3–43}$$

For each $s \in S^{(m)}$ in session m, let $i_s^{(m)}$ be the index of a minimum-cost tree rooted at s (i.e., with s as the source). That is,

$$i_s^{(m)}(z) = \text{argmin}_{\{i: t_{s,i}^{(m)} \in T_s^{(m)}\}} \{\frac{\partial f(z)}{\partial z_{s,i}^{(m)}}\}. \tag{3–44}$$

Let $i_s^{(m)}(k)$ be a short hand of $i_s^{(m)}(z(k))$.

Diagonally Scaled Gradient Projection Algorithm

$$z(k+1) = \alpha(k)\bar{z}(k) + (1 - \alpha(k))z(k) \tag{3–45}$$

$$\bar{z}_{s,i}^{(m)}(k) = \begin{cases} [z_{s,i}^{(m)}(k) - \delta_s^{(m)}(k) \cdot (d_{s,i}^{(m)}(k))^{-1} (\frac{\partial f(z(k))}{\partial z_{s,i}^{(m)}} - \frac{\partial f(z(k))}{\partial z_{s,i_s^{(m)}(k)}^{(m)}})]_+, & \text{if } i \neq i_s^{(m)}(k); \\ r_s^{(m)} - \sum_{1 \leq j \leq |T_s^{(m)}|, j \neq i_s^{(m)}(k)} \bar{z}_{s,j}^{(m)}(k), & \text{if } i = i_s^{(m)}(k), \end{cases} \tag{3–46}$$

with

$$d_{s,i}^{(m)}(k) = \sum_{\hat{e} \in t_{s,i}^{(m)} \cup t_{s,i_s^{(m)}(k)}^{(m)} \setminus t_{s,i}^{(m)} \cap t_{s,i_s^{(m)}(k)}^{(m)}} \sum_{e \in \hat{e}} \frac{q(q-1)}{c_e^2} (\frac{x_e(k)}{c_e} + \kappa)^{q-2}. \tag{3–47}$$

The resulting algorithm is still fully distributed.

3.3.3 Convergence Results

We will show the convergence results of the synchronous gradient projection algorithm under constant step size (i.e., $\delta(k) = \delta$ for all k). We will adapt the results from [53] to find an upper bound on the step size δ that guarantees the global convergence of the synchronous gradient projection algorithm to an optimal solution. Furthermore, with the strictly positive

regularization vector $\vec{\kappa}$ (i.e., $\kappa > 0$), the convergence speed is linear (i.e., geometric). The same convergence results can be said for the case of optimally allocated bandwidth.

In the optimization problem (3–21), we assume $q \geq 2$, so that $\hat{f}_e(x_e)$ is continuous on the interval $[0, \infty)$, tends to ∞ as x_e approaches ∞, and its derivative and second derivative are continuous and positive on $(0, \infty)$. Assuming the links are indexed from 1 to $|\hat{E}|$, the Hessian $\nabla^2 \hat{f} = \text{diag}[\frac{\partial^2 \hat{f}}{\partial x_1^2}, \cdots, \frac{\partial^2 \hat{f}}{\partial x_{|\hat{E}|}^2}]$ is an $|\hat{E}| \times |\hat{E}|$ diagonal matrix with nonnegative diagonal entries. Furthermore, if $\kappa > 0$, the diagonal entries of $\nabla^2 \hat{f}$ are positive and bounded below by $\min_{e \in \hat{E}} \{\frac{q(q-1)}{c_e^2} \kappa^{q-2}\}$.

We assume there is at least one feasible solution (i.e., $z(0) \in \mathcal{Z}$ satisfying $Hz(0) \in \prod_{e \in \hat{E}} [0, \infty)$), and define a compact set $\mathcal{Z}_0 = \{z \in \mathcal{Z} | f(z) \leq f(z(0))\}$. Since this set is compact, f must attain a minimum on this set. Hence, there is a $z^* \in \mathcal{Z}_0$ satisfying $f(z^*) = f^*$, where $f^* = \min_{z \in \mathcal{Z}_0} f(z)$. We call any such z^* an optimal solution, and we denote by $\mathcal{Z}^*$ the set of optimal solutions. (There may be more than one optimal solutions since, although $\hat{f}$ is strictly convex, f is not.) That is

$$\mathcal{Z}^* = \{z \in \mathcal{Z}_0 | f(z) = \min_{z \in \mathcal{Z}_0} f(z)\}.$$

For simplicity, we assume the scaling factor $d_{s,i}(k) = 1.0$. The convergence results still hold for other scaling factors as long as $\{d_{s,i}(k)\}$ are bounded between two fixed positive scalars [53] [47].

Let $\delta_1 = \underline{a}/(L \max_{s \in S} |T_s|)$, where $L > 0$ is an upper bound of the norm of $\nabla^2 f$ over $\mathcal{Z}_0$.

Lemma 5. *For $0 < \delta \leq \delta_1$, we have for all k that $z(k) \in \mathcal{Z}_0$*

$$f(z(k+1)) - f(z(k)) \leq -(\frac{\underline{a}}{\delta \max_{s \in S} |T_s|} - \frac{L}{2}) ||z(k) - \bar{z}(k)||^2. \tag{3–48}$$

The proof Lemma 5 follows the proof for a similar lemma in [53]. The only change involves substitution of appropriate constants.

Theorem 6. *(Globally Convergence) Suppose $\kappa \geq 0$. For any δ, $0 < \delta < \delta_1$, every limit point of $\{z(k)\}$ generated by the synchronous gradient projection algorithm (3–35)-(3–37) with $z(0) \in \mathcal{Z}_0$ is optimal.*

Proof. With the constant step size $\delta < \frac{\underline{a}}{L \max_{s\in S} |T_s|}$, the right-hand side of the inequality (3–48) is non-positive. Hence, if $\{z(k)\}$ has a limit point, the left-hand side tends to 0. The algorithm (3–35)-(3–37) can be denoted as a function $A(z)$ (i.e., $z(k+1) = A(z(k))$). Therefore, $||z(k) - \bar{z}(k)|| \to 0$, which implies that for every limit point $\tilde{z}$ of $\{z(k)\}$ we have $\tilde{z} = A(\tilde{z})$. It is easy to show if $\tilde{z} = A(\tilde{z})$, for any $s \in S$, we have $\tilde{z}_i^s > 0$ only if $\frac{\partial f(\tilde{z})}{\partial \tilde{z}_{s,i}} \leq \frac{\partial f(\tilde{z})}{\partial \tilde{z}_{s,j}}, \forall t_{s,j} \in T_s$, which is exactly the optimality condition in (3–25). So $\tilde{z}$ is stationary (Proposition 2.3.2 and Example 2.1.2 in [47]). □

When the regularization vector $\vec{\kappa}$ is strictly positive, the diagonal entries of $\nabla^2 \hat{f}$ are positive and bounded below by $\min_{e\in\hat{E}}\{\frac{q(q-1)}{c_e^2}\kappa^{q-2}\} > 0$. When $q = 2$, the diagonal entries of $\nabla^2 \hat{f}$ are positive and bounded below by $\min_{e\in\hat{E}}\{\frac{q(q-1)}{c_e^2}\} > 0$ for all $\kappa \geq 0$. In these two cases, all conditions for global geometric convergence required by [53] are satisfied. We can state the global geometric convergence rate for algorithm (3–35)-(3–37).

Theorem 7. *(Globally Geometric Convergence Rate)*

Suppose $\kappa > 0$. Let δ satisfy $0 < \delta \leq \delta_1$. The sequence $\{z(k)\}$ generated by the synchronous gradient projection algorithm (3–35)-(3–37) converges to an element of $\mathcal{Z}^*$ with an initial feasible $z(0)$ and the convergence rate is linear (i.e., geometric) in the sense that for all k,

$$f(z(k+1)) - f^* \leq (1 - D_5\delta)(f(z(k)) - f^*). \tag{3–49}$$

Furthermore, when $q = 2$, the above conclusion holds for all $\kappa \geq 0$.

The constants and parameters are as follows. $D_5 = \underline{a}/(D_4 + \delta_1)$, $D_4 = ((5L + 1)(D_3)^2 + 1 + 2\delta_1 + 6L(\delta_1)^2/\underline{a}) \max_{s\in S} |T_s|$ and $D_3 = D \max\{1, \delta_1\}$ for some $D > 0$. Moreover, D is bounded above by $D_1(D_1 + (\sqrt{\max_{s\in S} |T_s|} + 1)\hat{L}||H^T||)/\hat{\sigma}$, where $D_1 = \max\{||Q^{-1}|| \mid Q$ an invertible submatrix of $H\}$. $\hat{\sigma} \leq \hat{L}$ are any two positive scalars such that the diagonal entries of $\nabla^2 \hat{f}(Hz)$ lie inside $[\hat{\sigma}, \hat{L}]$ for all $z \in \mathcal{Z}_0$, where $\nabla^2 \hat{f}(Hz)$ is a positive diagonal matrix.

3.4 Column Generation Method

Though the algorithms presented in Section 3.3 avoid the enumeration of all possible spanning trees, they compute a minimum-cost spanning tree (cf. (3–24) and (3–44)) at each iteration, which is expensive. In this section we will introduce the column generation method to reduce the number of times when the computation of minimum-cost trees is invoked. We will describe the column generation method on the problem with fixed overlay link bandwidth; the column generation method works on the problem with optimally allocated overlay bandwidth in a similar way, and for simplicity, we do not detail it in this chapter.

Let us call the problem (3–17) the master problem (MP). We call the sub-problem of finding a new minimum-cost spanning tree in (3–24), *a global tree searching problem*, since it involves finding a tree from all possible ones, the solution to this sub-problem *a global minimum-cost tree*, and the achieved minimum cost *the global minimum tree cost*. We denote this global minimum cost under a fixed z for any source $s \in S$ by

$$\gamma_s(z) = \min_{i=1}^{|T_s|} \{\frac{\partial f(z)}{\partial z_{s,i}}\}. \tag{3–50}$$

3.4.1 Introduction of Column Generation Method

The main idea of column generation is to start with a subset of trees and bring in new trees only when needed. Consider a subset of $\mathcal{Z}$, $\mathcal{Z}^{(\vec{w})} = \{z : \sum_{i=1}^{w_s} z_{s,i} = r_s, \forall s \in S, z_{s,i} \geq 0, \forall i = 1, \cdots, w_s, \forall s \in S\}$, where $1 \leq w_s \leq |T_s|$ for any source $s \in S$. We can formulate the following restricted master problem (RMP) for $z \in \mathcal{Z}^{(\vec{w})}$.

$$\vec{w}\text{-RMP: } \min \sum_{e \in \hat{E}} (\frac{x_e}{c_e} + \kappa)^q \tag{3–51}$$

$$\text{s.t.} \sum_{i=1}^{w_s} z_{s,i} = r_s, \quad \forall s \in S \tag{3–52}$$

$$z_{s,i} \geq 0, \quad \forall i = 1, \cdots, w_s, \quad \forall s \in S. \tag{3–53}$$

The value of each component of $\vec{w}$ is usually small and the trees of $\mathcal{Z}^{(\vec{w})}$ in the $\vec{w}$-RMP are enumerable.

The $\vec{w}$-RMP is more restricted than the MP. Thus, any optimal solution to the the $\vec{w}$-RMP is feasible to the MP and serves as an upper bound of the optimal value of the MP. By gradually introducing more trees (columns) into $\mathcal{Z}^{(\vec{w})}$ and expanding the subset $\mathcal{Z}^{(\vec{w})}$, we will improve the upper bound of the MP.

3.4.2 Apply the Gradient Projection Algorithm to the Restricted Problem

The distributed diagonally scaled gradient projection algorithm can be used to solve the $\vec{w}$-RMP. Here, we define the following problem under the sending rate vector z.

$$i_s^{(\vec{w})} = \text{argmin}_{i=1}^{w_s}\{\frac{\partial f(z)}{\partial z_{s,i}}\}, \forall s \in S. \tag{3–54}$$

The optimization is taken over the w_s currently known trees for each source $s \in S$. The problem in (3–54) is called a *local tree searching problem*, the solution to this sub-problem is called *a local minimum-cost tree*, and the achieved minimum cost is called the *local minimum tree cost*. We denote this local minimum cost under z by

$$\gamma_s^{(\vec{w})}(z) = \min_{i=1}^{w_s}\{\frac{\partial f(z)}{\partial z_{s,i}}\}. \tag{3–55}$$

If there are more than one tree achieving the local minimum cost, the tie is broken arbitrarily.

3.4.3 Gap between the Master Problem and the Restricted Problem

Now the question is how to check whether the optimum of the $\vec{w}$-RMP is optimal to the MP, and if not, how to introduce a new column (tree). It turns out there is an easy way to do both.

Let z^* denote one of the optimal solutions of the MP, and $\bar{z}^{(\vec{w})}$ denote one of the optimal solutions of the $\vec{w}$-RMP. Since any optimal solution to the $\vec{w}$-RMP is feasible to the MP and the $\vec{w}$-RMP is more restricted than the MP, we get the following upper bound for the optimal objective value of the MP.

$$f(z^*) \leq f(\bar{z}^{(\vec{w})}). \tag{3–56}$$

To derive a lower bound of the optimal objective value, consider the optimality condition of the $\vec{w}$-RMP, which is for any source $s \in S$,

$$\bar{z}_{s,i}^{(\vec{w})} > 0 \text{ only if } [\frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,j}} \geq \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}}, \forall j = 1, \cdots, w_s]. \tag{3–57}$$

For each source $s \in S$, let I_s denote the set of indices of the minimum-cost trees at the optimum $\bar{z}^{(\vec{w})}$ of the $\vec{w}$-RMP, that is, $I_s = \{i : \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} \leq \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,j}}, j = 1, \cdots, w_s, i = 1, \cdots, w_s\}$. Obviously, for any i^{th} tree $t_{s,i} \in T_s$ and $i \notin I_s$, $\bar{z}_{s,i}^{(\vec{w})} = 0$. Since $f(z)$ is convex and $\bar{z}^{(\vec{w})}$ is feasible to the MP, we have

$$\begin{aligned}
f(z^*) &\geq f(\bar{z}^{(\vec{w})}) + \nabla f(\bar{z}^{(\vec{w})})^T (z^* - \bar{z}^{(\vec{w})}) \\
&= f(\bar{z}^{(\vec{w})}) + \sum_{s \in S} (\sum_{i \in I_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} (z_{s,i}^* - \bar{z}_{s,i}^{(\vec{w})}) + \sum_{i \notin I_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} (z_{s,i}^* - \bar{z}_{s,i}^{(\vec{w})})) \\
&= f(\bar{z}^{(\vec{w})}) + \sum_{s \in S} (-\sum_{i \in I_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} \bar{z}_{s,i}^{(\vec{w})} + \sum_{i \in I_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} z_{s,i}^* + \sum_{i \notin I_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} z_{s,i}^*) \\
&= f(\bar{z}^{(\vec{w})}) + \sum_{s \in S} (-\min_{i=1}^{w_s} \{\frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}}\} \sum_{i \in I_s} \bar{z}_{s,i}^{(\vec{w})} + \sum_{i=1}^{T_s} \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}} z_{s,i}^*) \\
&\geq f(\bar{z}^{(\vec{w})}) + \sum_{s \in S} (-r_s \min_{i=1}^{w_s} \{\frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}}\} + \min_{i=1}^{|T_s|} \{\frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}}\} \sum_{i=1}^{T_s} z_{s,i}^*) \\
&= f(\bar{z}^{(\vec{w})}) - \sum_{s \in S} r_s (\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) - \gamma_s(\bar{z}^{(\vec{w})})).
\end{aligned} \tag{3–58}$$

The first inequality is due to the property of convex functions. The second equality is expanding the inner product. The third equality holds because $\bar{z}_{s,i}^{(\vec{w})} = 0$ for any i^{th} tree $t_{s,i} \in T_s$ and $i \notin I_s$. The fourth equality holds by the definition of I_s and re-arranging the terms. The fifth inequality holds because $\sum_{i \in I_s} \bar{z}_{s,i}^{(\vec{w})} = r_s$ by the optimality condition (3–57), the definition of I_s and the feasibility of $\bar{z}^{(\vec{w})}$; and $z^* \geq 0$. The last equality is due to $\sum_{i=1}^{T_s} z_{s,i}^* = r_s$.

By (3–56) and (3–58), the gap between the upper and lower bounds for the optimal objective value of the MP is $\sum_{s \in S} r_s (\gamma_s^{(w)}(\bar{z}^{(\vec{w})}) - \gamma_s(\bar{z}^{(\vec{w})}))$. Since $r_s (\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) - \gamma_s(\bar{z}^{(\vec{w})})) \geq 0$ and the equality holds only if $\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) = \gamma_s(\bar{z}^{(\vec{w})})$ for any $s \in S$.

Lemma 6. *Let $\bar{z}^{(\vec{w})}$ denote one of the optimal solutions of the w^{th}-RMP. $\bar{z}^{(\vec{w})}$ is optimal to the MP if and only if $\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) = \gamma_s(\bar{z}^{(\vec{w})})$ for any source $s \in S$.*

Proof. By (3–56) and (3–58), if $\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) = \gamma_s(\bar{z}^{(\vec{w})})$ for any source $s \in S$, $f(\bar{z}^{(\vec{w})}) = f(z^*)$ and $\bar{z}^{(\vec{w})}$ is optimal to the MP.

If $\bar{z}^{(\vec{w})}$ is optimal to the MP, according to the optimality condition of the MP (3–26), for any source $s \in S$, $\bar{z}_{s,i}^{(\vec{w})} > 0$ only if $[\frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,j}} \geq \frac{\partial f(\bar{z}^{(\vec{w})})}{\partial z_{s,i}}, \forall j = 1, \cdots, |T_s|]$. Furthermore, $\bar{z}_{s,i}^{(\vec{w})} = 0$ for all $i = w_s + 1, \cdots, |T_s|$, for any source $s \in S$, which implies $\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) = \gamma_s(\bar{z}^{(\vec{w})})$. □

3.4.4 Introduce One More Column (Tree)

If the gap between the upper and lower bound, $\sum_{s \in S} r_s(\gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})}) - \gamma_s(\bar{z}^{(\vec{w})}))$, is not narrowed enough, then $\mathcal{Z}$ is not sufficiently well characterized by $\mathcal{Z}^{(\vec{w})}$ and a new tree should be added to the RMP. We state the rule of introducing a new tree in the following.

Fact 8. *Any tree achieving a cost less than the local minimum tree cost could enter the subset $\mathcal{Z}^{(\vec{w})}$ in the RMP. The tree achieving the global minimum tree cost is one possible candidate and is often preferred.*

Lemma 6 says, at the current solution $\bar{z}^{(w)}$, if none of the trees that achieve the global minimum tree cost are in the subset $\mathcal{Z}^{(\vec{w})}$, then the current optimal solution of the $\vec{w}$-RMP is not optimal to the MP. In this case, there are reasons to prefer the introduction of the globally optimal tree specified by (3–24) as the new tree to the restricted master problem. This strategy is a local greedy approach to improve the upper bound of the optimal value of the MP. In fact, it can be viewed as a conditional gradient method for optimizing the upper bound, when the upper bound is viewed as a function of z [7].

3.4.5 Summary of the Algorithm

We summarize the column generation method in Algorithm 3 and make several comments regarding this algorithm.

- In the worst case, the column generation method may bring in all the trees. However, it often happens that, within a relatively small number of column-generation steps, the optimal solution to the MP is already in $\mathcal{Z}^{(\vec{w})}$. Thus, the original problem may be solved without generating all the trees [7].

Algorithm 3 Column Generation Method

- Initialize: Start with a collection of $\vec{w}$ trees
- Step 1: Run the diagonally scaled gradient projection update (3–35)-(3–37) for several (a finite number) times on the RMP.
- Step 2: Solve the global tree searching problem (3–24) under the current first derivative cost $\partial f(z)$ for each source $s \in S$.
  0. If for each source $s \in S$, the tree corresponding to the solution of (3–24) is already in the current collection of trees, go to Step 1;
  0. otherwise, introduce this tree into the current collection of trees, increase $\vec{w}$, and go to Step 1.

- Algorithm 3 in fact describes a whole class of algorithms. In one end of the spectrum, if the diagonally scaled gradient projection algorithm in step 1 runs only once on the RMP, the algorithm becomes a pure diagonally scaled gradient projection algorithm as in Section 3.3. In the other end of the spectrum, if the diagonally scaled gradient projection algorithm runs on the RMP until convergence, the algorithm becomes a pure column generation method with the diagonally scaled gradient projection algorithm as a building block for solving the restricted problems between consecutive column generation steps. By choosing different numbers of times to run the diagonally scaled gradient projection algorithm in step 1, we have many algorithms, representing different performance, convergence speed and complex tradeoffs.

3.4.6 Convergence Result

Theorem 9. *There exists a $\vec{w}$, $1 \leq w_s \leq |T_s|$ for any source $s \in S$, such that Algorithm 3 converges to one optimal solution of this particular $\vec{w}$-RMP (i.e., $\bar{z}^{(\vec{w})}$). Furthermore, after Algorithm 3 converges to $\bar{z}^{(\vec{w})}$, $\gamma_s(\bar{z}^{(\vec{w})}) = \gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})})$ for any $s \in S$.*

Proof. Since the number of trees of $\mathcal{Z}$ is finite, eventually Algorithm 3 will stop introducing new trees. Hence, there exists a $\vec{w}$, $1 \leq w_s \leq |T_s|$ for any source $s \in S$, such that, after Algorithm 3 stops introducing new trees, the number of trees that have been introduced is w_s for each source $s \in S$. Let the convex hull formed by these $\vec{w}$ trees be denoted by $\mathcal{Z}^{(\vec{w})}$. After Algorithm 3 no longer introduces new trees, it behaves just like the diagonally scaled gradient projection algorithm but on the restricted set $\mathcal{Z}^{(\vec{w})}$. According to the Theorem 7, the diagonally scaled gradient projection algorithm converges. Thus, Algorithm 3 converges to $\bar{z}^{(\vec{w})}$ on this particular $\vec{w}$-RMP.

We next show that, after Algorithm 3 converges to $\bar{z}^{(\vec{w})}$, we have $\gamma_s(\bar{z}^{(\vec{w})}) = \gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})})$ for any $s \in S$. First, note that $\gamma_s(z) \leq \gamma_s^{(\vec{w})}(z)$ by definition. Next, it must be true that $\gamma_s(\bar{z}^{(\vec{w})}) \geq \gamma_s^{(\vec{w})}(\bar{z}^{(\vec{w})})$ for any $s \in S$. Otherwise, the tree whose cost is $\gamma_s(\bar{z}^{(\vec{w})})$ must not have already been in $\mathcal{Z}^{(\vec{w})}$ and will be selected to enter. This violates the assumption that the algorithm never selects more than $\vec{w}$ trees. □

By Theorem 9 and Lemma 6, finally we have,

Corollary 3 (Convergence of Column Generation Method)**.** *Algorithm 3 converges to an optimum of the MP.*

3.5 Practical Considerations

The problem formulations in the previous sections omit some details that are practically required. The purpose of this omission is for ease of presentation. These simplified formulations contain the technical core, or the most difficult aspect, of the problem. For the most part, the formulations are without loss of generality. Practical details can be easily incorporated into the formulations. We now address several of these.

3.5.1 Overlapping Content

When some sources share overlapping chunks, we can create a virtual source node that connects all those sources and move all the overlapping chunks to the virtual source. The virtual source has one outgoing virtual link with infinite capacity connecting to each original source. We then arrive at an expanded network. Of course, in actual operation, one of the original physical sources will "act" as the virtual source and run the algorithms assigned to the virtual source.

3.5.2 Mixed Architecture of Fixed and Allocated Bandwidth

In Section 3.2.2 and 3.2.3, we see two content distribution scenarios with either fixed or optimally-allocated overlay bandwidth. The latter should achieve better downloading time than the former. However, the latter requires the deployment of our algorithms to all network element (i.e. routers), which is almost certainly impossible. There is an alternative framework in which some routers or devices attached to the routers are deployed with our algorithm, while others not. For instance, our algorithm can be deployed at the cross-ISP links and access links, where the

bandwidth is more likely to be small. In this framework, for those physically directly connected router pairs deployed with our algorithms, we model the physical links between them exactly; while for those not directly connected devices, routers and end-systems, we let TCP allocate the end-to-end bandwidth between them and model these end-to-end paths as overlay links. Thus, in our graph of the network, some links are real physical links and others are overlay links with TCP-allocated bandwidth. The algorithm applies as usual.

3.5.3 Network Dynamics and Churn

Thus far, we assume that the network is stable and no members depart or join until all existing members finish downloading. Since we are considering the distribution of massive files or large collections of files, the assumption is reasonable for the most part. However, we do need to deal with low-degree member churn and network dynamics such as link capacity variations and failures.

If any source owning unique chunks leaves before it finishes disseminating them, those chunks are no longer available in the network. To minimize such risk, in the optimization formulation, we can adjust the requested sending rates r_s of the sources. If any source is expected to leave the network soon, it may request (or be assigned) a higher sending rate so that it can spread its chunks to network more quickly.

Other types of network and member dynamics include link failures, the change of link capacities, the arrival of new sources, and the departure and arrival of receivers. As argued in [22], a distributed algorithm has built-in ability to adapt to variations. A distributed algorithm can react rapidly to a local disturbance at the point of disturbance with slower fine tuning in the rest of the network. Such adaptive ability is intimately connected with the algorithm's speed of convergence in the static case. Since our algorithm is the result of conscious effort to improve the convergence speed (by diagonal scaling of the gradient algorithm), we believe it is superior in coping with network and member dynamics compared to other similar distributed algorithms. In addition, our distributed algorithm is naturally robust because of the lack of reliance on a

central node that might fail. In Section 3.6, we will show a small example of how our algorithm successfully adapts to the departure and arrival of receivers.

3.5.4 Scalability and Hierarchical Partition of Sessions

In each overlay network, the sources know the complete information of all overlay links and need to run the expensive centralized MDSP algorithm (There is a distributed MDSP algorithm [85]. But the price to pay is the potentially slower speed due to the coordination overhead of distributed operation.). Thus, our gradient algorithm can only deal with distribution sessions with limited size, say several thousands of members in each session. In order to improve the scalability of our algorithm, we shall partition each session and run the algorithm hierarchically, as most scalable network algorithms would do. Though currently we do not have a well-defined way to partition the session, we will show one naive approach to partition a large session in Section 3.6.

3.5.5 Asynchronous Algorithm

Time synchrony is usually difficult to maintain in a large network. An asynchronous version of the scaled gradient algorithm (3–35)-(3–37) (or (3–45)-(3–47), respectively) could be developed and the corresponding convergence result could be stated following the approach in [54] [53].

3.6 Performance Evaluation

In this section we will compare the performance of our gradient algorithm (GP) with known theoretical bounds, BitTorrent (BT) and Adaptive FastReplica (AFR) [74]. We select BitTorrent because its techniques are interesting and it is the most popular P2P application. We select AFR because it can be thought as using multiple multicast trees for distribution. But only a subset of the trees are allowed, which we call two-level two-phase trees. In each of these trees, the source is connected to one receiver at level 1, and then the level 1 receiver is connected to all other receivers at level 2. It might appear that such a collection of trees is quite enough for achieving near optimal performance. In an access-constrained network, this is indeed true. However, we

will show this is not the case for networks with interior bottlenecks. In that case, a different, maybe larger, collection of trees is needed.

Although we are more interested in network interior bottlenecks, our algorithm can equally deal with bottlenecks at the access links, at the ISP backbone or at the cross-ISP links. Hence, we will consider all these cases. The commercial ISP backbone and cross-ISP topologies are obtained from the Rocketfuel project [86]. In the terminology of BitTorrent, a seed is a source, and a leecher is a receiver. In the previous sections, our objective function is the worst network utilization $||\vec{\mu}||_\infty$. In the evaluation part, we will focus on the source throughput $R_s = r_s/||\vec{\mu}||_\infty$, where r_s is the scaled sending rate, and the downloading time $t = L_s/R_s$, since these are what BitTorrent experiments yield directly. However, recall that the two measures are the two sides of the same coin.

3.6.1 Performance Evaluation Metrics

BitTorrent simulation. We use the Bittorrent simulator developed by Bharambe et. al. [87]. Since the original simulator only supports access link constraint, we modified the simulator so that it supports general physical network topologies. The overlay link bandwidth, which is the per-connection bandwidth at the underlay, is determined by the max-min bandwidth allocation [82]. In the BitTorrent simulation, we use the following simulation environment.

- Each peer opens 5 uploading connections and one of them is selected by optimistic unchoking, which means that it connects to a random neighbor.
- The seed uses the smart seed policy, which is introduced in [87]. Seeds are with distinct files.
- All the peers join the network at time 0 and continue to be in the network until all of the leechers complete the download.
- All the other parameters follow the regular BitTorrent environment.

Table 3-2 summarizes the simulation environment for our test cases.

Adaptive FastReplica. AFR supports single source. To compare with AFR, we partition the physical network into several overlay networks, each for one source according to max-min fairness allocation. AFR constructs two-phase trees as described earlier. From [74], the

theoretical throughput of AFR in its best behavior can be computed as

$$\sum_{i=1,\cdots,m} \min\{c_{n_0 n_i}, \min_{j=1,\cdots,m, j\neq i}\{c_{n_i n_j}\}\}, \tag{3–59}$$

where n_0 is the source, n_i, $i = 1, \cdots, m$ are the receivers, and $c_{n_i n_j}$ is the end-to-end path (overlay link) capacity between n_i and n_j.

Theoretical bounds. Some theoretical results are used as performance benchmark in our performance comparison. In several studies [88–90], researchers have analyzed a model of P2P file sharing among residential users in low access-speed environment. Each participating end-system has an uplink (to the network) and a downlink with limited capacity. The capacity of the network is considered unlimited. The source is to distribute a file to L receivers. Let the uplink (downlink) bandwidth of receiver i be u_i (d_i, respectively), for $i = 1, \cdots, L$. Let the uplink capacity of the source be u_s. Then, the maximum distribution speed is shown to be

$$\min(u_s, \min_{1\leq i\leq L} d_i, \frac{u_s + \sum_{1\leq i\leq L} u_i}{L}). \tag{3–60}$$

In (3–60), the three terms are the optimal speeds when the bottleneck is at the source upload link, at a download link, or due to the aggregate upload bandwidth, respectively.

By two-phase distribution, we mean each distribution tree has a depth at most 2. The following fact is known to be true.

Fact 10. *In a network with only access-speed constraint, the two-phase distribution [88] achieves the (overlay-network) routing capacity [88], which is (3–60).*

In the single source case, there is a maximum flow from the source to each receiver. The minimum of all these maximum flows will be called the *max-flow limit* (MFL). The max-flow limit is a throughput (total distribution rate) upper bound. If all nodes but the source are receivers, the max-flow limit is achievable. This result is known as Edmond's Theorem [91] [92].

3.6.2 Bottleneck at the Access Links (Profiles 1 to 4)

In this case, we assume the network has infinite capacity but the access links have finite capacities. We also assume that all access links are deployed with our gradient algorithm. In the

four test cases (profile 1-4), we have a single source with uploading bandwidth u_s. Let u_i and d_i be leecher (receiver) i's upload and download bandwidth, respectively.

- Profile 1: $u_i = d_i = 360$ Kbps for all 299 receivers, $u_s = 640$ Kbps. The download link is the bottleneck.
- Profile 2: $u_i = d_i = 360$ Kbps for all 299 receivers, $u_s = 280$ Kbps. The source upload link is the bottleneck.
- Profile 3: $d_i = 360$ Kbps, $u_i = 200$ Kbps for all 299 homogeneous receivers, $u_s = 640$ Kbps. The aggregate upload bandwidth is the bottleneck.
- Profile 4: $d_i = 360$ Kbps for all 100 receivers, $u_i = 100$ Kbps for half of receivers, and $u_i = 1$ Kbps for the rest receivers. $u_s = 100$ Kbps. The aggregate upload bandwidth is the bottleneck.

In Table 3-3, the optimal downloading time is computed from by (3–60). The results indicate that the gradient algorithm obtains near the theoretically optimal solution.

Comparison with BitTorrent. Table 3-3 shows the time when 50%, 95% and 100% receivers finish downloading in BitTorrent, respectively. BitTorrent's performance is not bad compared with the optimal value. This was explained in [66], which models the downloading time of BiTtorrent. It shows that, in the case that a flash crowd arrives at the same time, the bandwidth constraint is at the access links, and the receivers stay after they finish downloading, BitTorrent achieves near optimal distribution speed. In Fig. 3-3 and 3-4, we show the performance comparison of different distribution schemes under profile 1 and 4. The results under profile 2 and 3 are omitted for brevity. The download percentage refers to the total amount of data downloaded at each time instance normalized against the total data downloaded at the end of the distribution. Since the two lines have different slopes, we can extrapolate the lines and expect the gradient algorithm to do much better if the file size becomes larger. This observation seems to contradict the conclusion in [66].

Comparison with AFR. AFR downloading time is given by (3–59). Table 3-3 shows that AFR's two-phase approach achieves the optimal downloading time when the bottleneck is either at the download link or at the source. But when the bottleneck is due to the aggregate upload bandwidth, AFR fails to achieve the optimum, although we know that some other two-phase

solution is optimal. The reason is that, in AFR, every receiver is required to relay all chunks it receives from the source to other receivers. This unnecessary constraint leads to a sub-optimal solution. AFT does not allow the breadth-first search tree, but an optimal two-phase tree does. Nevertheless, AFR achieves good performance in this kind of access-limited situation.

We also compare the number of active trees AFR and the gradient algorithm eventually use. The gradient algorithm uses fewer trees than AFR. We inspected the active trees. With the access-link constraint, the gradient algorithm uses three kinds of trees, the depth-first search tree (DFS), the breadth-first search tree (BFS), and a two-phase tree. Fig. 3-5 shows the structures of the three types of trees on the overlay network. In general, there exists an optimal solution that uses only two-phase trees for networks with such a star topology. But it seems that the gradient algorithm prefers the chain-like DFS tree and the fat BFS tree. It may appear counter-intuitive that such a chain-like distribution path is preferred because the chain seems to involve largest delay. However, this is in fact not true because of our fluid model of traffic and because we do not consider propagation delay. The bit that arrives at node 1 from the source can immediately be transmitted to node 2, and to node 3, so on. We leave it to future work on how to incorporate the propagation delay in optimal tree selection. Table 3-1 shows that, when the bottleneck is at either the download links or the source, the gradient algorithm naturally prefers the DFS tree. When the aggregate upload bandwidth is the bottleneck, we have to distribute the chunks over the two-phase trees. In addition, the more heterogenous the receivers are, the more two-phase trees we need and the more bandwidth is allocated to the two-phase trees. The key conclusion here is that the gradient algorithm is able to find the best distribution trees for the particular network environment. Without the help of the gradient algorithm, what types of trees are selected is not always obvious.

3.6.3 Bottleneck at the Internal of ISP Backbone (Profile 5)

In this case, we assume all access links have unlimited bandwidth, and congestion happens at the core network. We wish to see how our algorithm performs in infrastructure-mode content distribution. We did experiments with the ISP Sprintlink's backbone obtained from [86]. The

network has 315 backbone nodes and 1944 links. It is the largest backbone ISP with the highest node degree among the six commercial backbone networks that the RocketFuel project provides. We attach 100 peers with unlimited access bandwidth randomly to some backbone nodes, with at most one peer per backbone node. One may think a peer is a large content distribution server cluster. Among the 100 peers, we choose 2 sources with the normalized sending rates $r_s = 1.0$ (dimensionless value).

We did several experiments with the link capacities uniformly distributed in some range. The actual link capacity data is unavailable. We find the gradient algorithm often gives trivial optimal solutions. After inspecting the solutions and the network graphs, it turns out that the ISP backbone is poorly connected. There are many links that lies on all the routing paths between one peer and all other peers, which means if any one of these critical links is removed, at least one peer will be disconnected. If these critical links do not have much larger capacity than other links, they are likely to become the bottleneck. The gradient algorithm is able to locate the bottleneck immediately. Other five ISP backbones show the same property. Presumably in reality, the ISPs are aware of such links and would ensure they have very large bandwidth so that they are never the bottleneck. In order to test our algorithm in this non-trivial scenario, we assign the same bandwidth, 1000, to all backbone links. Then, we scale up the bandwidth of all critical links (those links that, when removed, will leave some peers disconnected in the overlay) to be large enough so that they are not the bottleneck.

We did three tests on Profile 5.

- (Test a) Our algorithm is deployed at all links. This is the case of optimally-allocated overlay bandwidth.
- (Test b) Our algorithm is deployed only at the peers. The overlay bandwidth between each pair of peers is fixed by the max-min allocation. The 100 peers form a single overlay network.
- (Test c) Our algorithm is deployed only at the peers. In order to compare with AFR, we partition the backbone into two overlay networks, one for each source. Note that the max-flow limit is achievable in this case.

Comparison with BitTorrent. Fig. 3-6 shows that, in Test a, the downloading time in the gradient algorithm is only 30% of that in BitTorrent. BitTorrent is unable to give good performance when the core network is congested. But in Test b, after the overlay bandwidth is fixed, the optimal downloading time is much higher than that of Test a, and almost equal to BitTorrent's time. In the figure, the download percentage refers to the total amount of data downloaded at each time instance normalized against the total data downloaded at the end of the distribution. Since the lines have different slopes, we can extrapolate the lines and expect the gradient algorithm to do much better if the file size becomes larger.

Comparison with AFR. Fig. 3-6 also shows that, in Test c, the gradient algorithm approaches the max-flow limit while AFR achieves something far from the optimum. When the congestion happens at the core network, the two-phase trees alone fail to give good solution.

Convergence speed. Fig. 3-7 shows the convergence of the algorithm in Test a, b and c respectively. The time spent on one iteration is about one round trip time plus the time to compute the MDSP. It seems that the algorithm that optimally allocates the overlay bandwidth converges much faster than the algorithm with fixed overlay bandwidth. This has to do with the fact that, in the final solution, Test a has 92 active trees, while Test b has totally 4746 active trees. It is possible Test b has another optimal or nearly optimal solution that has much fewer trees. Finding solutions with fewer trees should improve convergence speed, and is an important direction to pursue.

3.6.4 Bottleneck at the Cross-ISP Links (Profile 6-7)

In reality, the cross-ISP links are often the bandwidth bottleneck. The goal here is to evaluate the effectiveness of our algorithm in handling the bottleneck at cross-ISP links when it is deployed at ISP gateways. We did experiments both on an artificial small cross-ISP network and the cross-ISP network obtained from the RocketFuel project. We created scenarios where congestion happens at the cross-ISP links. Our algorithm is deployed at all access links and cross-ISP links. Hence, we are able to run the algorithm to optimally allocate the overlay bandwidth.

- Profile 6 (P6): 6 completely connected ISPs with 30 cross-ISP links. Each cross-ISP link has a capacity of 1000. 300 peers are attached to the ISPs, 50 per ISP, with sufficient access bandwidth. A single source is attached to one ISP.

- Profile 7 (P7): (RocketFuel topology): 69 ISPs connected with 1336 links. The cross-ISP link capacities are uniformly distributed on $(100, 1000)$. 500 peers are randomly attached to the ISPs with sufficient bandwidth. A single source is attached to one ISP.

Note that, in the case of a single source, congestion at the cross-ISP links and each ISP containing some peers, the max-flow limit is achievable.

In both profiles 6 and 7, the gradient algorithm approaches the max-flow limit and beats BitTorrent and AFR by a large amount, up to a factor of 10 (Tabel 3-4: 50%, 95% and 100% are the percentage of the peers that have finished downloading. Also see Fig. 3-8 and 3-9.). We found, with more peers per ISP, BitTorrent's performance deteriorates. FastReplica's performance is far worse than both the gradient algorithm and BitTorrent, and we did not even show it in Fig. 3-8 and 3-9.

Usually, cross-ISP traffic is more expensive. We investigated the traffic redundancy over the cross-ISP links. With neither inside ISP congestion nor access speed constraint, ideally, each destination ISP should receive only one copy of each chunk from other ISPs and the source ISP should not receive any copy from other ISPs. In Profile 6, we inspected the active (optimal) trees the algorithm constructed and found that each destination ISP indeed only received one copy from other ISPs, but the source ISP might receive some copies from other ISPs. Suppose the normalized cross-ISP traffic under the ideal distribution is 1.0. We found the cross-ISP traffic was 1.103 in the gradient algorithm and 5.982 in BitTorrent. But in Profile 7, we found the destination ISPs received multiple copies from other ISPs in the gradient algorithm. This is because, in Profile 6, each max-flow between the source ISP and the destination ISP has the same value. This is not the case in Profile 7. Thus, the optimal solution does allow multiple copies to be sent to one destination ISP. Again, if the normalized cross-ISP traffic under the ideal distribution is 1.0, then the traffic is 2.22 in the gradient algorithm and 9.72 in BitTorrent.

3.6.5 Introduce Trees at Varying Degree of Frequency (Profile 5)

As discussed in Section 3.4, we can introduce new trees at varying degree of frequency. We did experiments with the Profile 5. In the experiments, we will use three frequencies: fast, medium and slow. With the fast frequency, we try to introduce trees by solving the global tree searching problem (3–24) at each update of (3–35)-(3–37), in which case, Algorithm 3 degenerates into the pure gradient projection algorithm. With the slow frequency, we try to introduce a new tree after every 20 updates of (3–35)-(3–37). In this case, Algorithm 3 is more close to the pure column generation method. With the medium frequency, we introduce a new tree every 5 updates.

We evaluate the algorithm on the overlay network with the single source obtained from Profile 5 Test c. In Fig. 3-10, the fast scheme always improves the throughput more quickly at the beginning, while the slow scheme improves it much more slowly than the other two schemes. The reason is that, with the fast scheme, plenty of trees are introduced quickly. The slow scheme always tries to take full advantage of the current collection of trees. But later, the slow scheme catches up the fast scheme. This motivates the use of the medium scheme. In Fig. 3-10, we see that the medium scheme increases the throughput nearly as quickly as the fast scheme at the beginning and it surpasses the fast scheme soon after.

In Table 3-5, we compare the three schemes for their computation costs. Since the most expensive computation is for solving the global tree searching problem (3–24), the total computation time is mainly characterized by the number of times the global tree searching problem is solved. One expects that lowering the frequency of introducing new trees is correlated with fewer computations for the global tree searching problem. But, we know no theoretical reasons why this must be true. The result confirms the expectation: The number of such computations is 3500, 700, and 300, for the fast, medium and slow schemes. The reduction is dramatic.

We also find, with a lower frequency, the algorithm usually produces a solution with fewer active trees. Fewer active trees may be desirable since it is easier to manage and control them, which may reduce the system complexity and control overhead. The slow scheme only uses

244 active trees in the end, which are all those that ever computed and entered. In other words, there are no redundant trees; nor are there redundant computations for the trees. The fast and medium schemes use 1815 and 313 active trees respectively. In the fast scheme, 354 trees have been introduced into the collection but are not used in the final optimal solution. In the medium scheme, the number of redundant trees is 6.

Based on the study of Fig. 3-10 and Table 3-5, we conclude that the pure two-timescale (fast) or the pure column generation (slow) algorithms have both pros and cons. An intermediate algorithm (medium) may achieve a more desirable balance among factors such as optimization performance, the computational cost, and system complexity and overhead.

3.6.6 Arrival and Departure Dynamics (Profile 8)

Here, we wish to examine how well the distributed gradient algorithm copes with the peer arrival and departure dynamics. We applied the algorithm with optimally allocated overlay bandwidth on a small star network with receivers arrive and depart randomly. All peers have sufficient download capacities; the receivers each have upload bandwidth 200 Kbps; and the source has upload bandwidth 640 Kbps. In Phase-I, we have one source and 10 receivers; in Phase-II, 2 receivers leave; in Phase-III, 5 new receivers arrive. Assume at the beginning of Phase-III the source has distributed half of the chunks to the 8 existing receivers. We assume at the end of Phase-III, all 13 receivers finish at the same time (though it is unfair). Thus, we set up two more sessions at Phase-III: one consists of the 5 new receivers and the original source for downloading the second half of the chunks; and the other consists of the 5 new receivers, the source and the 8 existing receivers (now also serving as sources) for downloading the first half of the chunks. Thus, in Phase-III, we have 3 sessions, and in the 3rd session, we have multiple sources with overlapping chunks. Fig. 3-11 shows the algorithm adapts to the dynamics quickly.

3.7 Additional Related Work

In our optimization-based approach, we follow the tradition of Kelly et. al. [31] and Low et. al. [32] on optimal flow control/bandwidth allocation. Many recent papers extended this approach and solved networking problems by collective actions taken across networking layers,

especially in wireless networks (e.g., [8, 9, 35, 93]). Several other related studies, either in topics or methods, are [94–97].

3.8 Conclusion

This chapter represents a systematic study on how best to conduct content distribution using advanced P2P techniques. In response to growing massive content that threatens to congest the core network, our objective is to manage the network congestion not only at the access links but throughout the network, especially at cross-ISP links. We showed that this objective is "equivalent" to speeding up content distribution. The main contribution of this chapter is that we envision optimal content distribution as a multicast tree packing problem, and we derive a distributed algorithm for solving the problem. Furthermore, to reduce the number of the computation of minimum-cost trees, we adopt a column generation approach. The combination of the tree packing algorithm and column generation leads to a family of algorithms with interesting tradeoffs. The tree-packing framework is also useful for contemplating existing P2P swarming/collaborative downloading techniques, by asking the questions: What kind of trees do existing algorithms use? How are the trees selected? And how is bandwidth assigned to the trees? Hence, our framework has the potential to provide a unified understanding of P2P distribution techniques. Finally, our distributed algorithm is based on a specialized gradient projection algorithm for optimization under simplex constraints and we develop a scaled version of it. Our computation experiences show that it has much faster convergence than the more frequently used subgradient algorithm.

Table 3-1. Distribution of bandwidth allocated for different trees.

	Profile 1	Profile 2	Profile 3	Profile 4
DFS	99.4%	99.999%	98.91%	1.93%
BFS	0.26%	0	0.754%	0.95%
Two-phase	0.33%	0.001%	0.333%	97.1%

Table 3-2. BitTorrent simulation parameters.

	#Seeds	File Size per Seed	Neighborhood Size
Profiles 1-3	1	62.765 MB	38 - 80
Profile 4	1	128 MB	18 - 40
Profile 5	2	64 MB	18 - 40
Profile 6	1	128 MB	38 - 80
Profile 7	1	32 MB	38 - 80

Table 3-3. Comparison of downloading time (minutes) and number of active trees.

	Profile 1	Profile 2	Profile 3	Profile 4
Optimum	23.8	30.6	42.4	331.4
BT(50%)	27.6	41.0	44.9	264.8
BT(95%)	28.5	41.3	49.7	428.6
BT(100%)	30.4	41.5	51.0	441.8
AFR	23.8	30.6	42.7	337.9
GP	23.9	30.6	43.5	333.1
GP #trees	3	2	3	53
AFR #trees	299	299	299	100

Table 3-4. Downloading time (minutes) comparison of Profile 6 and 7.

	BT(50%)	BT(95%)	BT(100%)	GP	AFR	MFL
P6	16	19.5	19.6	3.8	142.7	3.4
P7	5.83	26.5	90	8.74	131.2	8.72

Table 3-5. Performance comparison of the family of algorithms (Profile 5).

	Fast	Medium	Slow
#Iterations	3500	3500	6000
#Trees Computed	3500	700	300
#Active Trees	1815	313	244
#Trees Introduced	2169	319	244

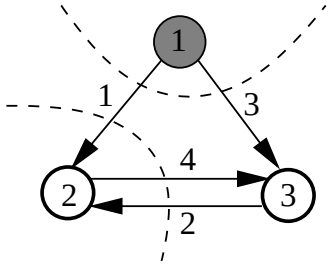


Figure 3-1. Node 1 sends the file to node 2 and 3.

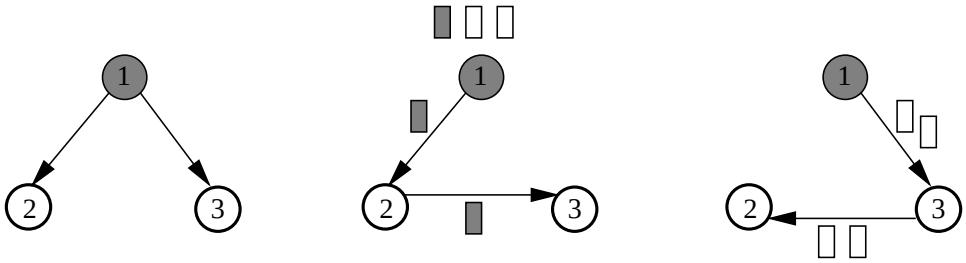


Figure 3-2. All possible distribution trees for the example in Fig. 3-1.

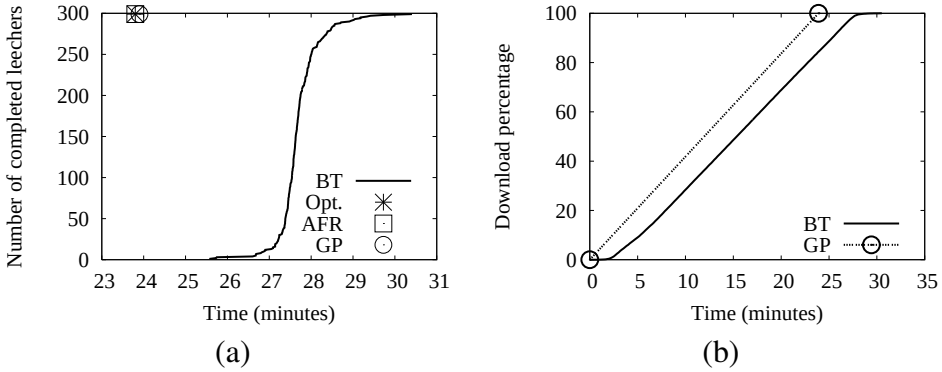


Figure 3-3. Performance comparison (Profile 1): (a) number of leechers that have completed download over time; (b) download percentage over time.

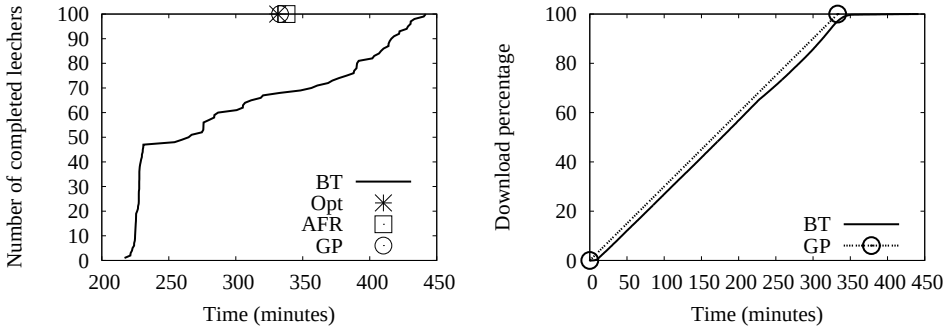


Figure 3-4. Performance comparison (Profile 4): (a) number of leechers that have completed download over time; (b) download percentage over time.

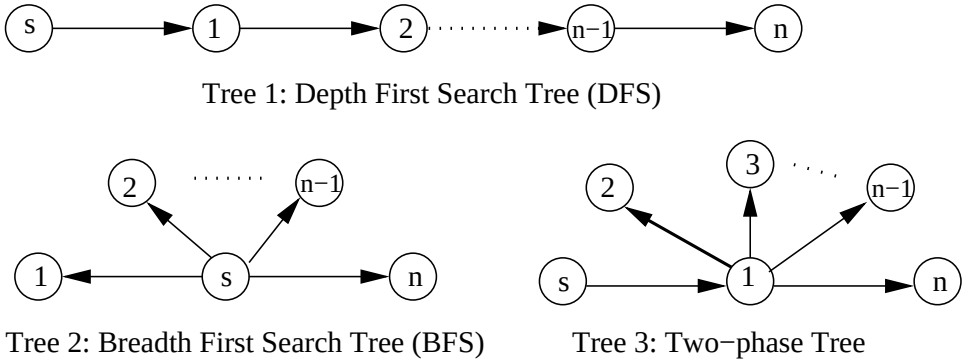


Figure 3-5. Structures of trees on the overlay network.

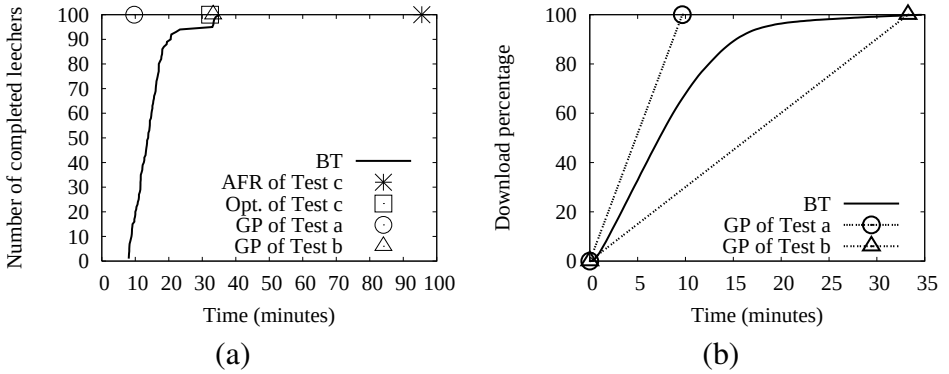


Figure 3-6. Performance comparison (Profile 5): (a) number of receivers that have completed download over time; (b) download percentage over time.

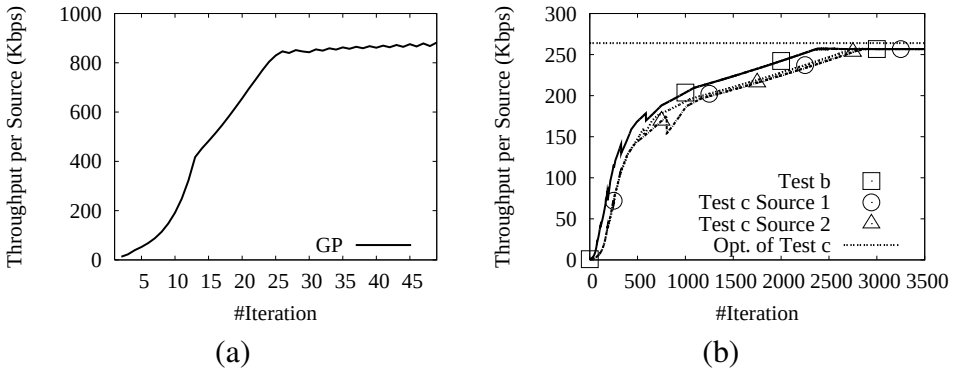


Figure 3-7. Convergence of throughput (Profile 5). Two sources with $r_s = 1.0$: (a) Test a; (b) Test b and c.

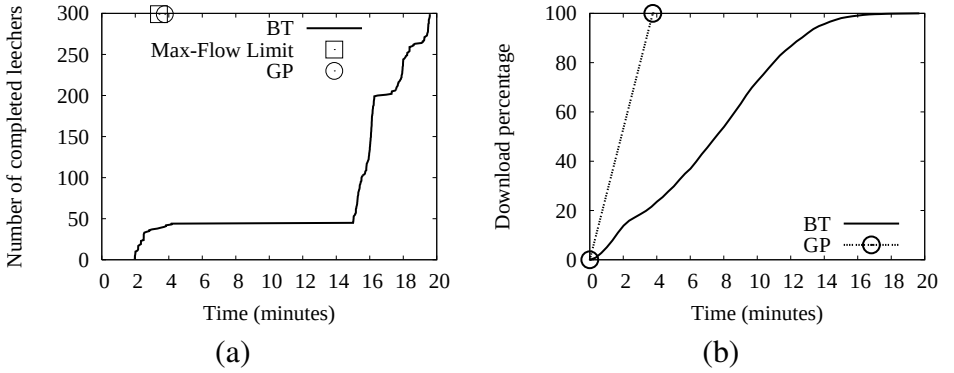


Figure 3-8. Performance comparison (Profile 6): (a) number of receivers that have completed download over time; (b) download percentage over time.

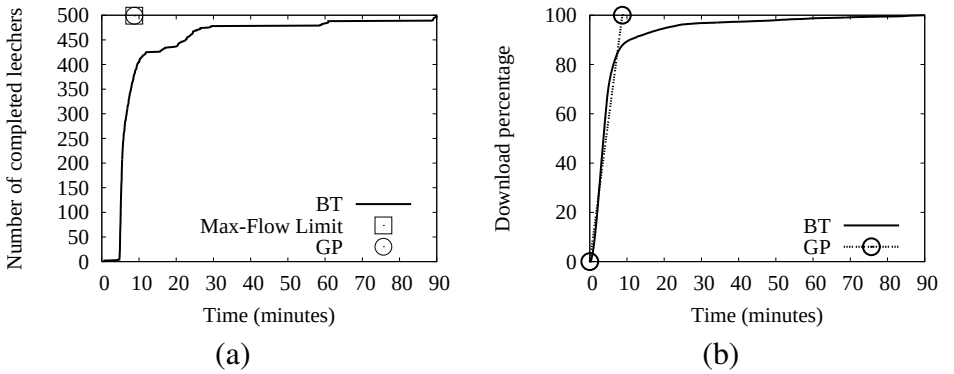


Figure 3-9. Performance comparison (Profile 7): (a) number of receivers that have completed download over time; (b) download percentage over time.

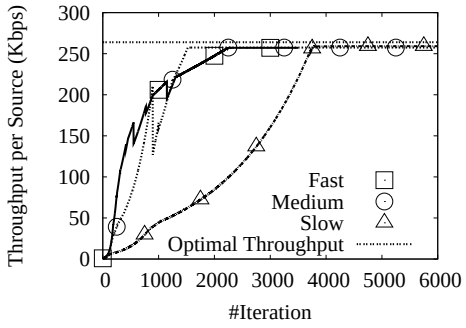


Figure 3-10. Convergence of the family of algorithms (Profile 5).

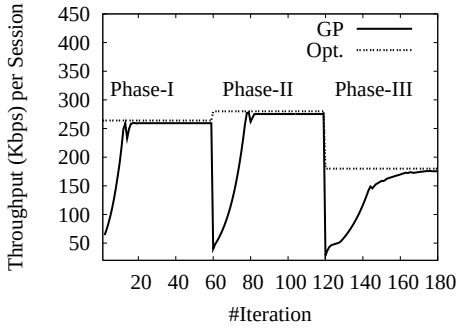


Figure 3-11. Dynamic departure and arrival of receivers.

CHAPTER 4
A CLASS OF CROSS-LAYER OPTIMIZATION ALGORITHMS FOR PERFORMANCE AND COMPLEXITY TRADE-OFFS IN WIRELESS NETWORKS

4.1 Introduction

The joint congestion-control and scheduling problem in multi-hop wireless networks has become a very active research area in the last few years [7–19]. The problem can be formulated as the maximization of the aggregate source utility over the network capacity constraints. Unlike the similar problem in the wired network, the essential nature of the problem in the wireless setting is that the network capacity itself is a decision variable. Due to wireless interference, not all transmission configurations are allowed at each time instance. For instance, in the well-known model of the multiple access scheme for the 802.11 network, an allowed configuration is a subset of the links whose transmissions do not interfere with each other. Scheduling at the MAC layer is to decide which of the allowed configurations should be used and how they should be used (e.g., time shared). The result of scheduling implicitly determines the network capacity.

The standard subgradient algorithm is a good candidate in solving such a problem. By the subgradient technique, the rate control and the wireless resource allocation are decoupled: The sources adapt their source rates according to the path congestion costs, whereas the MAC-layer scheduling adjusts the time share of different allowed transmission configurations, thus varying the link capacities according to the link costs so as to support the flow rates. However, the standard subgradient technique has its own limitation, which will be discussed.

We propose a *two-timescale*, *column-generation* approach with *imperfect* global scheduling to solve the above problem. Compared with the subgradient technique and others, our approach offers the following features.

- Our approach solves the difficult issue of recovering the time share using a two-timescale method. The issue arises when the Lagrangian function of the maximization problem is not strictly concave in all its primal variables. Specifically, in the subgradient algorithm, the dual problem converges to an optimal solution. However, the primal variables corresponding to the time share proportions oscillate. Our two-timescale algorithm ensures that both the primal and dual solutions converge to the optimum.

- The column generation method introduces one extreme point at a time and and gradually expands the feasible set, where an extreme point corresponds to one allowed transmission configuration (also known as a *schedule*). Typically, introducing such an extreme point involves solving an NP-hard combinatorial optimization problem [10, 11]. In our approach, we allow the introduction of a sub-optimal extreme point, which is often far easier to obtain. This opens the door for the application of many heuristic algorithms in solving the hard combinatorial problem. Importantly, we show that, if the sub-optimal extreme point is a ρ-approximation solution to the combinatorial optimization problem, then the overall utility-maximization problem also achieves ρ approximation.

- By combining column generation and the aforementioned two-timescale algorithm, we in fact have a whole family of algorithms. On one side of the spectrum, we have a pure column generation algorithm; on the other side, we have a pure two-timescale algorithm. In between, we have a mixed algorithm that introduces new extreme points at varying degree of frequency, thus balancing various aspects of the algorithm (e.g., performance and complexity).

We subsequently call the sub-problem of finding a new extreme point in the column generation algorithm, *global scheduling*, since it involves finding an allowed transmission schedule from all possible ones. This sub-problem is a combinatorial optimization problem on an exponential number of possibilities. A *perfect schedule* refers to an optimal solution to the sub-problem; an *imperfect schedule* refers to a sub-optimal solution to the sub-problem. Other algorithms usually also contain this sub-problem. How to avoid global scheduling as much as possible and how to solve it fast when needed are two key issues. This chapter makes contributions in both.

We now give a brief summary of prior work on the joint design of congestion control, routing and scheduling in wireless networks. A survey of resource allocation and cross-layer control in wireless networks can be found in [19]. Varbrand et al. [14] propose the column generation method to solve the resource allocation problem in wireless ad hoc networks. Johansson and Xiao [7] extend the use of the column generation method to solve the same problem under more comprehensive wireless interference models. But both [14] and [7] give centralized solutions, where the restricted master problems are solved by some linear/nonlinear solvers (We are interested in distributed algorithms.); and they only consider the case where perfect scheduling is used. [17] also gives a centralized column generation solution. Bohacek and

Wang [12] implicitly apply the column generation method and their approach is centralized. In [10, 11], the authors propose a way to solve this problem by a distributed subgradient algorithm with imperfect scheduling. Their approach and conclusion are different from ours and we will detail the differences in Section 4.4. In [9, 15], the authors formulate similar problems as ours and develop subgradient algorithms. [18] discusses the framework of cross-layer optimization in wireless networks. Another related chapter is [13]. The two-timescale adaptive method is proposed in [22], and used in [98, 99] for the problem of multi-path routing. To our best knowledge, no prior work has combined the three elements together, two timescales, column generation and imperfect scheduling.

The chapter is organized as follows. The network model and problem formulation are given in Section 4.2. The two-timescale algorithm and its convergence proof are given in Section 4.3. In Section 4.4, we present the column generation approach, combine it with the two-timescale method, and study the impact of imperfect scheduling. We show the performance with imperfect scheduling is bounded. In Section 4.5, we give the experimental examples. The conclusion is drawn in Section 4.6.

4.2 Problem Description

Let the network be represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. The presence of link $e \in E$ means that the network is able to send data from the start node of e to the end node of e. Unlike in a wired system where the capacity of a link is a fixed constant, in a wireless system, due to the shared nature of the wireless medium, the rate c_e of a link e depends not only on its own modulation/coding scheme, power assignment P_e and the ambient noise, but also on the interference from other transmitting links, which in turn depends on their power assignments. Let $P = (P_e)$ denote a vector of a global power assignment and let $c = (c_e)$ denote the vector of the corresponding link rates, where $0 \leq P_e \leq P_{e,\max}$ for all $e \in E$. We assume the data rates c are completely determined by the global power assignment P, which means there exists a rate-power function u such that $c = u(P)$ [11]. The rate-power function is determined by the interference model.

We describe the following model as an example. Let $G_{ee'}$ denote the effective power gain between the sender of link e and the receiver of link e'. Let σ_e denote the thermal noise power at e's receiver. The signal to interference and noise ratio (SINR) of link e is

$$\omega_e(P) = \frac{G_{ee}P_e}{\sigma_e + \sum_{e' \in E, e' \neq e} G_{e'e}P_{e'}}. \tag{4–1}$$

According to Shannon's capacity theorem, the maximum data rate of link e is $c_e = W \log(1 + \omega_e(P))$, where W is the system bandwidth. In practice, the link rate is usually lower than the Shannon capacity. Typical wireless systems allow a finite set of link rates, say $c_e^1, \cdots, c_e^k$, which are associated with a set of thresholds for the SINR, $\beta_e^{(c_e^1)}, \cdots, \beta_e^{(c_e^k)}$. This is usually due to the finite number of modulation/coding schemes built into the wireless transceiver. A link e can use the transmission rate c_e^j, if $\omega_e \geq \beta_e^{(c_e^j)}$.

To summarize, at any time instance, the number of possible rate vectors is finite. Each of these allowed rate vectors will be called a *schedule*. Let Q denote the total number of schedules. Let $c^{(i)} = (c_e^{(i)})$ denote the i^{th} schedule (rate vector) in the set of feasible schedules, for $i = 1, \cdots, Q$, where the order is arbitrary. Though Q is finite, it might be exponential in the number of links. By time sharing of these feasible schedules, the achievable time-average link-rate region is the convex hull of $c^{(i)}$, $i = 1, \cdots, Q$. Denote this convex hull by $\mathcal{C}$. Thus, $\mathcal{C}$ is a convex polytope. With slight abuse of terminology, we call $c^{(i)}$, $i = 1, \cdots, Q$, the extreme points of $\mathcal{C}$. In fact, some of them may not be extreme points of the polytope. For any $c \in \mathcal{C}$, it could be represented by the following convex combination of the extreme points of $\mathcal{C}$,

$$c = \sum_{i=1}^{Q} \alpha_i c^{(i)}, \sum_{i=1}^{Q} \alpha_i = 1, \alpha_i \geq 0, i = 1, \cdots, Q, \tag{4–2}$$

where α_i denotes the time-share fraction of the schedule that uses the schedule $c^{(i)}$. One can find more discussion on wireless interference models in [7].

4.2.1 Network Model

Suppose there is a set of source-destination pairs. Let S be the set of sources and x_s be the source rate of source $s \in S$. Assume the flow between each source-destination pair is routed

along the fixed single path, and denote this path by p_s for each source s. Define $U_s(x_s), x_s \geq 0$, the utility function for each source $s \in S$. Assumptions on the utility functions are, for every $s \in S$,

- $A1$: U_s is increasing, strictly concave and twice continuously differentiable for all $x_s \geq 0$.
- $A2$: $U_s(x_s) \geq 0$ for all $x_s \geq 0$.
- $A3$: $U_s'(x_s)$ is well-defined and bounded at $x_s = 0$.

The optimal resource allocation and scheduling problem is formulated as

$$\max \sum_{s\in S} U_s(x_s) \tag{4–3}$$

$$\text{s.t.} \quad \sum_{s:e\in p_s} x_s \leq c_e, \quad \forall e \in E$$

$$c \in \mathcal{C} \tag{4–4}$$

$$x_s \geq 0, \qquad \forall s \in S.$$

By replacing (4–4) with the equivalent expression in (4–2), we re-write the above problem as follows.

$$\text{MP:} \qquad \max \sum_{s\in S} U_s(x_s) \tag{4–5}$$

$$\text{s.t.} \quad \sum_{s:e\in p_s} x_s \leq \sum_{i=1}^{Q} \alpha_i c_e^{(i)}, \quad \forall e \in E \tag{4–6}$$

$$\sum_{i=1}^{Q} \alpha_i = 1$$

$$x_s \geq 0, \quad \forall s \in S$$

$$\alpha_i \geq 0, \quad \forall i = 1, \cdots, Q.$$

Note that $c_e^{(i)}$ is a constant instead of a decision variable, and the only decision variables are x and α. We call the above problem the master problem (MP).

4.2.2 Dual of Master Problem

Let λ_e be the Lagrangian multiplier associated with the constraint (4–6). The Lagrangian function of MP is

$$\begin{aligned} & L(x, \alpha, \lambda) \\ = & \sum_{s \in S} U_s(x_s) + \sum_{e \in E} \lambda_e (\sum_{i=1}^{Q} \alpha_i c_e^{(i)} - \sum_{s: e \in p_s} x_s). \\ = & \sum_{s \in S} (U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e) + \sum_{i=1}^{Q} \alpha_i (\sum_{e \in E} \lambda_e c_e^{(i)}). \end{aligned}$$

The dual function is

$$\begin{aligned} \theta(\lambda) = \quad & \max \quad L(x, \alpha, \lambda) \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{Q} \alpha_i = 1 \\ & x_s \geq 0, \qquad \forall s \in S \\ & \alpha_i \geq 0, \qquad \forall i = 1, \cdots, Q. \end{aligned} \tag{4–7}$$

Now the dual problem of MP is

$$\begin{aligned} \text{Dual-MP:} \quad & \min \quad \theta(\lambda) \\ \text{s.t.} \quad & \lambda \geq 0. \end{aligned} \tag{4–8}$$

4.3 Two-Timescale Algorithm

In this section, we will illustrate how the MP can be solved by a two-timescale algorithm. In Section 4.4, we will combine this two-timescale algorithm with a column generation algorithm and derive a family of algorithms.

We first consider the rate control problem with *fixed* time fraction vector α.

$$\text{MP-A:} \quad \Phi(\alpha) := \max_x \textstyle\sum_{s \in S} U_s(x_s) \tag{4–9}$$

$$\text{s.t.} \quad \textstyle\sum_{s: e \in p_s} x_s \leq \sum_{i=1}^{Q} \alpha_i c_e^{(i)}, \quad \forall e \in E \tag{4–10}$$

$$x_s \geq 0, \qquad \forall s \in S.$$

The above problem MP-A has a strictly concave objective function and has a unique solution with respect to the only variable, vector x. $\Phi(\alpha)$ denotes the optimal objective function value of MP-A under each α. The original problem MP can be re-written as

$$\begin{aligned} \text{MP-B:} \quad & \max_{\alpha} \ \Phi(\alpha) \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{Q} \alpha_i = 1 \\ & \alpha_i \geq 0, \qquad \forall i = 1, \cdots, Q. \end{aligned} \tag{4–11}$$

4.3.1 Solve Problem MP-A with the Subgradient Method

The problem MP-A could be solved by the subgradient algorithm[1] . Let λ_e be the Lagrange multiplier associated with the constraint (4–10). The Lagrangian function of MP-A is

$$\begin{aligned} & L_A(\alpha, x, \lambda) \\ = & \sum_{s \in S} U_s(x_s) + \sum_{e \in E} \lambda_e \Big(\sum_{i=1}^{Q} \alpha_i c_e^{(i)} - \sum_{s: e \in p_s} x_s \Big) \\ = & \sum_{s \in S} \Big(U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e \Big) + \sum_{i=1}^{Q} \alpha_i \Big(\sum_{e \in E} \lambda_e c_e^{(i)} \Big). \end{aligned}$$

The dual function is

$$\theta_A(\alpha, \lambda) = \sum_{s \in S} \max_{x_s \geq 0} \{ U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e \} + \sum_{i=1}^{Q} \alpha_i \Big(\sum_{e \in E} \lambda_e c_e^{(i)} \Big)$$

Since MP-A is maximizing a strictly concave function with linear constraints, the strong duality holds for MP-A [47]. Since there is no duality gap at the optimum of MP-A under a fixed α, we can re-write $\Phi(\alpha)$ as the optimal objective function value of the dual problem of MP-A,

$$\text{Dual-MP-A:} \quad \Phi(\alpha) = \min_{\lambda \geq 0} \ \theta_A(\alpha, \lambda). \tag{4–12}$$

[1] Since the time-share variable α is a constant vector, there is no difficulty with the subgradient algorithm here.

The dual problem (4–12) can be solved by the subgradient method as in Algorithm 4 ([47, 100]), where $\delta(t)$ is a positive scalar stepsize, and $[\cdot]_+$ denote the projection onto the non-negative domain.

Algorithm 4 Fast Timescale: Subgradient Algorithm for Solving MP-A

$$\lambda_e(t+1) = [\lambda_e(t) - \delta(t)(\sum_{i=1}^{Q} \alpha_i c_e^{(i)} - \sum_{s:e\in p_s} x_s(t))]_+, \quad \forall e \in E \tag{4–13}$$

$$x_s(t+1) = [(U_s')^{-1}(\sum_{e\in p_s} \lambda_e(t+1))]_+, \quad \forall s \in S \tag{4–14}$$

Let $(x^*(\alpha), \lambda^*(\alpha))$ denote the optimal primal-dual solutions of MP-A under a fixed α.

Theorem 11. *(Convergence of the Subgradient Algorithm for MP-A): With the diminishing step size rule (i.e.,* $\lim_{t\to\infty} \delta(t) = 0$ *and* $\sum_{t=1}^{\infty} \delta(t) = \infty$*), let* $\{\lambda(t)\}$ *be a sequence generated by Algorithm (4–13)-(4–14). Then* $\theta_A(\alpha, \lambda(t)) \to \theta_A(\alpha, \lambda^*(\alpha))$ *and there exists a subsequence* T *such that* $\{\lambda(t)\}_T \to \lambda^*(\alpha)$ *[100].*

Theorem 12. *(Convergence of the Primal Sequence for MP-A): Assume* $A1$*. In Algorithm (4–13)-(4–14), if* $\lambda(t) \to \lambda^*(\alpha)$*, then* $x(t) \to x^*(\alpha)$*.*

Proof. The proof is standard and is omitted. □

4.3.2 Update Time Fraction on a Slower Timescale

The above rate control algorithm (4–13)-(4–14) works under the assumption that the time fraction vector α remains constant. Now we discuss how to adjust $\alpha_i, i = 1, \cdots, Q$, to solve the problem MP-B. We assume the update of α is much slower so that the minimization of $\theta_A(\alpha, \lambda)$ over λ can be regarded as being instantaneous. Here, we follow the approaches in [22, 98, 99].

Let k index the time slots (called stages) of the slow timescale. At stage k, given the time fraction vector $\alpha(k)$, suppose $\lambda(k) \in \text{argmin}_{\lambda\geq 0}\theta_A(\alpha(k), \lambda)$ is an optimal dual solution to MP-A. Let's call $\lambda_e(k)$ the price or cost of link e. Therefore, $\lambda_e(k)c_e^{(i)}$ is the cost of link e under the i^{th} schedule (i.e., the i^{th} extreme point of $\mathcal{C}$); and $\sum_{e\in E} \lambda_e(k)c_e^{(i)}$ is the cost of the network under

the i^{th} schedule, which will be called the *cost of the schedule*. Let $i(k)$ be the index of a schedule achieving the maximum schedule cost under the link costs $\lambda(k)$, that is,

$$i(k) = \text{argmax}_{i=1}^{Q}\{\sum_{e\in E}\lambda_e(k)c_e^{(i)}\}. \tag{4–15}$$

If there is a tie, an arbitrary maximizing index is chosen. (4–15) may be called a scheduling problem [11], since it aims at finding a schedule. Because (4–15) is an optimization problem over all allowed schedules, $1, \cdots, Q$, we call (4–15) *a global scheduling problem*, and the achieved maximum cost *the global maximum cost of the schedule*. We denote this global maximum cost under a fixed λ by

$$\gamma(\lambda) = \max_{1\le i\le Q}\{\sum_{e\in E}\lambda_e c_e^{(i)}\}. \tag{4–16}$$

The time fraction update is shown in Algorithm 5, which is similar to the one in [22, 98, 99].

Algorithm 5 Slow Timescale: Time Fraction Update for Solving MP-B

$$\alpha_i(k+1) = \alpha_i(k) + \Delta_i(k) \tag{4–17}$$

$$\Delta_i(k) = \begin{cases} -\min\{\xi(k)(\sum_{e\in E}\lambda_e(k)c_e^{(i(k))} - \sum_{e\in E}\lambda_e(k)c_e^{(i)}), \alpha_i(k)\}, & \text{if } i \neq i(k) \\ -\sum_{i\neq i(k)}\Delta_i(k), & \text{if } i = i(k). \end{cases} \tag{4–18}$$

Here, $\xi(k)$ is a positive step size. Note that $\Delta_i(k) \le 0$ for $i \neq i(k)$ and $\Delta_i(k) \ge 0$ for $i = i(k)$. Hence, the algorithm increases the time fraction of the most costly schedule while decreases the time fractions of other active schedules (i.e., those schedules with positive time fractions $\alpha_i(k)$). Furthermore, if $\sum_{i=1}^{Q}\alpha_i(k) = 1$, then $\sum_{i=1}^{Q}\alpha_i(k+1) = 1$. Hence, $\alpha(k)$ will always be valid time fraction vectors for all k if $\sum_{i=1}^{Q}\alpha_i(0) = 1$.

It can be verified that

$$\sum_{i=1}^{Q} \Delta_i(k) = 0 \tag{4–19}$$

$$\sum_{i=1}^{Q} \Delta_i(k) \sum_{e \in E} \lambda_e(k) c_e^{(i)} \geq 0. \tag{4–20}$$

Equality in (4–20) occurs if and only if $\Delta_i(k) = 0$ for all i, which is equivalent to

$$\alpha_i(k)(\sum_{e \in E} \lambda_e(k) c_e^{(i(k))} - \sum_{e \in E} \lambda_e(k) c_e^{(i)}) = 0, \forall i. \tag{4–21}$$

Conditions in (4–19) - (4–21), and those described in the proceeding paragraph guarantee that the time fraction update algorithm converges to the correct optimal value. As in [99], we consider a continuous-time, differentiable version of the algorithm (4–17)-(4–18). First, define the set

$$\Lambda(\alpha) = \{\lambda \geq 0 : \theta_A(\alpha, \lambda) = \min_{\lambda' \geq 0} \theta_A(\alpha, \lambda')\}. \tag{4–22}$$

The differentiable version of the algorithm (4–17)-(4–18) satisfies the following conditions, for any $\lambda(\alpha) \in \Lambda(\alpha)$.

$$\sum_{i=1}^{Q} \dot{\alpha}_i = 0, \tag{4–23}$$

$$\sum_{i=1}^{Q} \dot{\alpha}_i \sum_{e \in E} \lambda_e(\alpha) c_e^{(i)} \geq 0, \tag{4–24}$$

$$\sum_{i=1}^{Q} \dot{\alpha}_i \sum_{e \in E} \lambda_e(\alpha) c_e^{(i)} = 0 \text{ if and only if } \dot{\alpha}_i = 0, \forall i. \tag{4–25}$$

The condition in (4–25) is equivalent to

$$\alpha_i(\sum_{e \in E} \lambda_e(\alpha) c_e^{(i(k))} - \sum_{e \in E} \lambda_e(\alpha) c_e^{(i)}) = 0, \forall i. \tag{4–26}$$

Theorem 13. *The time fraction update algorithm (4–17)-(4–18) converges to an optimal solution of the problem MP-B.*

Proof.

$$\Phi(\alpha) = \min_{\lambda \geq 0} \theta_A(\alpha, \lambda) = \min_{\lambda \geq 0} \{ \sum_{s \in S} (U_s(x_s(\lambda)) - x_s(\lambda) \sum_{e \in p_s} \lambda_e) + \sum_{i=1}^{Q} \alpha_i (\sum_{e \in E} \lambda_e c_e^{(i)}) \}.$$

Note that $\theta_A(\alpha, \lambda)$ is a continuous function. For each $\alpha \geq 0$, $\theta_A(\alpha, \cdot)$ is bounded from below (say, by $\sum_{s \in S} U_s(0)$). Hence, $\Phi(\alpha)$ is well defined on $\alpha \geq 0$. Furthermore, $\theta_A(\cdot, \lambda)$ is concave (actually linear), for each fixed λ. Hence, $\Phi(\alpha)$ is a concave function in α, which means it has directional derivatives. We will apply Danskin's theorem ([47], page 717). The theorem requires λ to be in a compact set. In other words, it requires that there exists a compact set Λ independent of α such that $\Phi(\alpha) = \min_{\lambda \geq 0} \theta_A(\alpha, \lambda) = \min_{\lambda \in \Lambda} \theta_A(\alpha, \lambda)$. We will next construct one such compact set. Since $U_s(\cdot)$ is concave, we have $U_s'(0) \geq U_s'(x_s)$ for all $x_s \geq 0$. Under assumption $A3$, take some $K \geq \max_{s \in S} U_s'(0) > 0$. Let $\Lambda = \{\lambda : 0 \leq \lambda_e \leq K, \forall e \in E\}$. For any $\lambda \notin \Lambda$, there exists a non-empty subset $E_1 \subseteq E$, where $\lambda_e > K$ for any $e \in E_1$ and $\lambda_e \leq K$ for any $e \notin E_1$. Let denote a subset of sources by $S_1 \subseteq S$, where for any source $s \in S_1$, its routing path p_s contains some links in the set E_1. We construct a vector $\lambda' \in \Lambda$ where $\lambda_e' = K$ for any $e \in E_1$, and $\lambda_e' = \lambda_e$ for any link $e \in E \setminus E_1$. For any $s \in S$, if its accumulated path cost is no less than K, then the maximum of $U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e$ in the definition of $\theta_A(\alpha, \lambda)$ is achieved at $x_s = 0$, which means for any $s \in S_1$,

$$U_s(0) = \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e\} = \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e'\}. \tag{4–27}$$

Then

$$\begin{aligned} &\theta_A(\alpha, \lambda) \\ &= \sum_{i=1}^{Q} \alpha_i (\sum_{e \in E} \lambda_e c_e^{(i)}) + \sum_{s \in S_1} \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e\} + \sum_{s \in S \setminus S_1} \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda_e\} \end{aligned}$$

$$\geq \sum_{i=1}^{Q} \alpha_i (\sum_{e \in E} \lambda'_e c_e^{(i)}) + \sum_{s \in S_1} \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda'_e\} + \sum_{s \in S \setminus S_1} \max_{x_s \geq 0} \{U_s(x_s) - x_s \sum_{e \in p_s} \lambda'_e\}$$

$$= \theta_A(\alpha, \lambda'). \tag{4–28}$$

Thus for any α, the minimum of $\theta_A(\alpha, \lambda)$ over $\lambda \geq 0$ occurs in Λ.

The conditions required by Danskin's theorem are met. Let $\Phi'(\alpha; \dot{\alpha})$ denote the directional derivative of $\Phi(\alpha)$ in the direction of $\dot{\alpha}$. Let $\theta'_A(\alpha, \lambda; \dot{\alpha})$ be the directional derivative of $\theta_A(\cdot, \lambda)$ at α in the direction of $\dot{\alpha}$. Then, by Danskin's theorem,

$$\begin{aligned} \Phi'(\alpha; \dot{\alpha}) &= \min_{\lambda \in \Lambda(\alpha)} \theta'_A(\alpha, \lambda; \dot{\alpha}) \\ &= \min_{\lambda \in \Lambda(\alpha)} \sum_{i=1}^{Q} (\sum_{e \in E} \lambda_e c_e^{(i)}) \dot{\alpha}_i \\ &= \sum_{i=1}^{Q} (\sum_{e \in E} \bar{\lambda}_e(\alpha) c_e^{(i)}) \dot{\alpha}_i. \end{aligned} \tag{4–29}$$

where $\bar{\lambda} \in \Lambda(\alpha)$ achieves the minimum.

Then, by (4–24),

$$\Phi'(\alpha; \dot{\alpha}) \geq 0. \tag{4–30}$$

By the Lasalle invariance principle [101], $\alpha(t)$ converges to an invariant set inside $\{\alpha : \Phi'(\alpha; \dot{\alpha}) = 0\}$. Take a trajectory in this invariant set, which satisfies $\Phi'(\alpha; \dot{\alpha}) \equiv 0$. By (4–29), $\sum_{i=1}^{Q} (\sum_{e \in E} \bar{\lambda}_e c_e^{(i)}) \dot{\alpha}_i \equiv 0$. Then, by (4–25), $\dot{\alpha}_i \equiv 0$ for all i. Hence, the invariant set has only one point, which will be denoted by α^*. Hence, $\alpha(t)$ converges to α^*.

Next, we will show that α^* solves problem MP-B. Let $x^*(\alpha^*)$ and $\lambda^*(\alpha^*)$ be the optimal solution of MP-A under α^*. MP-A maximizes a strictly concave function with linear constraints, and hence, the KKT conditions are both necessary and sufficient optimality conditions for MP-A [47]. Thus, at the optimum $(x^*(\alpha^*), \lambda^*(\alpha^*))$, we have that $x^*(\alpha^*)$ is primal feasible and $\lambda^*(\alpha^*)$ is

dual feasible for MP-A, and

$$x^*(\alpha^*) = \text{argmax}_{x\geq 0}\{U_s(x_s) - x_s \sum_{e\in p_s} \lambda_e^*(\alpha^*)\} \tag{4–31}$$

$$\lambda_e^*(\alpha^*)(\sum_{i=1}^{Q} \alpha_i^* c_e^{(i)} - \sum_{s:e\in p_s} x_s^*(\alpha^*)) = 0, \forall e \in E. \tag{4–32}$$

At α^*, we have $\dot{\alpha} = 0$. Hence, according to (4–26), we have

$$\alpha_i^* > 0 \text{ only if } \lambda_e^*(\alpha^*)c_e^{(i)} = \max_{j=1}^{Q}\{\lambda_e^*(\alpha^*)c_e^{(j)}\}. \tag{4–33}$$

Also, by (4–23), if we initialize the update of α at some $\alpha(0)$ satisfying $\sum_{i=1}^{Q} \alpha_i(0) = 1$, we will have

$$\sum_{i=1}^{Q} \alpha_i^* = 1, \tag{4–34}$$

which implies that

$$\alpha^* = \text{argmax}_{\alpha\geq 0:\sum_i \alpha_i=1}\{\sum_{i=1}^{Q} \alpha_i(\sum_{e\in E} \lambda_e^*(\alpha)c_e^{(i)})\}. \tag{4–35}$$

Obviously, $x^*(\alpha^*), \lambda^*(\alpha^*)$ and α^* are all non-negative. These non-negativity conditions, the fact that $x^*(\alpha^*)$ is primal feasible for MP-A, and the conditions in (4–31) - (4–32) and (4–34) - (4–35) are the optimality conditions of the MP. Hence, $(x^*(\alpha^*), \alpha^*, \lambda^*(\alpha^*))$ is an optimal primal-dual solution to the MP (also to MP-B). □

4.3.3 Summary of the Two-Timescale Algorithm

To summarize, the two-timescale algorithm consists of

- a fast timescale distributed algorithm for rate control, which adapts the source rates and link prices according to (4–13)-(4–14),
- a slow timescale algorithm for updating the time fraction according to (4–17)-(4–18).

However, in most wireless interference models, problem (4–15) does not even have a centralized polynomial-time solution. This has been the main obstacle in developing practical rate control/scheduling algorithms. In next section, we will try to overcome this difficulty.

4.4 Column Generation Method With Imperfect Global Scheduling

The global scheduling problem (4–15) is usually an NP-hard combinatorial problem [7, 10, 11]. One fundamental reason is that the convex polytope, $\mathcal{C}$, usually has an exponential number of extreme points in terms of the number of links. The column generation method with imperfect global scheduling can be introduced to overcome this difficulty. The column generation part reduces the number of times when the global scheduling problem is invoked. Imperfect scheduling uses fast approximation or heuristic algorithms for speedup.

4.4.1 Column Generation Method

The main idea of column generation is to start with a subset of the extreme points of $\mathcal{C}$ and bring in new extreme points only when needed. Consider a subset of $\mathcal{C}$ formed by convex combination of q extreme points, that is, $\mathcal{C}^{(q)} = \{c : c = \sum_{i=1}^{q} \alpha_i c^{(i)}, \sum_{i=1}^{q} \alpha_i = 1, \alpha_i \geq 0, \forall i = 1, \cdots, q\}$. We can formulate the following restricted master problem (RMP) for $c \in \mathcal{C}^{(q)}$.

$$
\begin{aligned}
q^{th}\text{-RMP:} \quad & \max \textstyle\sum_{s\in S} U_s(x_s) && && (4\text{–}36)\\
\text{s.t.} \quad & \textstyle\sum_{s:e\in p_s} x_s \leq \sum_{i=1}^{q} \alpha_i c_e^{(i)}, && \forall e \in E && (4\text{–}37)\\
& \textstyle\sum_{i=1}^{q} \alpha_i = 1 && && \\
& x_s \geq 0, && \forall s \in S && \\
& \alpha_i \geq 0, && \forall i = 1, \cdots, q. &&
\end{aligned}
$$

The value of q is usually small and the extreme points of $\mathcal{C}^{(q)}$ in the q^{th}-RMP are enumerable.

Let λ_e be the Lagrange multiplier associated with the constraint (4–37). The Lagrangian function of the q^{th}-RMP is

$$
\begin{aligned}
& L^{(q)}(x, \alpha, \lambda) \\
= \; & \sum_{s\in S} U_s(x_s) + \sum_{e\in E} \lambda_e \Big(\sum_{i=1}^{q} \alpha_i c_e^{(i)} - \sum_{s:e\in p_s} x_s\Big) \\
= \; & \sum_{s\in S} \Big(U_s(x_s) - x_s \sum_{e\in p_s} \lambda_e\Big) + \sum_{i=1}^{q} \alpha_i \Big(\sum_{e\in E} \lambda_e c_e^{(i)}\Big).
\end{aligned}
$$

The dual function is

$$\begin{aligned} \theta^{(q)}(\lambda) = \quad & \max \quad L^{(q)}(x, \alpha, \lambda) \\ & \text{s.t.} \quad \textstyle\sum_{i=1}^{q} \alpha_i = 1 \\ & \qquad x_s \geq 0, \qquad \forall s \in S \\ & \qquad \alpha_i \geq 0, \qquad \forall i = 1, \cdots, q. \end{aligned} \tag{4–38}$$

The dual problem of the q^{th}-RMP can be formulated similarly as in (4–8).

The q^{th}-RMP is more restricted than the MP. Thus, any optimal solution to the q^{th}-RMP is feasible to the MP and serves as a lower bound of the optimal value of the MP. By gradually introducing more extreme points (columns) into $\mathcal{C}^{(q)}$ and expanding the subset $\mathcal{C}^{(q)}$, we will improve the lower bound of the MP [7, 14, 17].

4.4.2 Apply the Two-Timescale Algorithm to the RMP

The two-timescale algorithm can be used to solve the q^{th}-RMP. Here, we define the following problem under the link cost vector $\lambda(k)$.

$$i^{(q)}(k) = \text{argmax}_{i=1}^{q} \{ \sum_{e \in E} \lambda_e(k) c_e^{(i)} \}. \tag{4–39}$$

The optimization is taken over the q currently known schedules (extreme-point link-rate vectors). The problem in (4–39) is called the *local scheduling problem*, and the achieved maximum cost is called the *local maximum cost of the schedule*. We denote this local maximum cost under $\lambda \geq 0$ by

$$\gamma^{(q)}(\lambda) = \max_{1 \leq i \leq q} \{ \sum_{e \in E} \lambda_e c_e^{(i)} \}. \tag{4–40}$$

If there is more than one link-rate vector achieving the local maximum cost, the tie is broken arbitrarily.

4.4.3 Bounding the Gap between the MP and the q^{th}-RMP

Now the question is how to check whether the optimum of the q^{th}-RMP is optimal for the MP, and if not, how to introduce a new column (schedule or extreme point). It turns out there is an easy way to do both.

Let $(x^*, \alpha^*, \lambda^*)$ denote one of the optimal primal-dual solutions of the MP, and $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$ denote one of the optimal primal-dual solutions of the q^{th}-RMP. Since the strong duality holds for both problems, we have

$$\sum_{s\in S} U_s(x_s^*) = \theta(\lambda^*), \sum_{s\in S} U_s(\bar{x}_s^{(q)}) = \theta^{(q)}(\bar{\lambda}^{(q)}). \tag{4–41}$$

Since the q^{th}-RMP is more restricted than the MP, we have

$$\sum_{s\in S} U_s(x_s^*) \geq \sum_{s\in S} U_s(\bar{x}_s^{(q)}). \tag{4–42}$$

Combining (4–41) and (4–42), we get the following lower bound for the optimal objective value of the MP.

$$\sum_{s\in S} U_s(x_s^*) \geq \sum_{s\in S} U_s(\bar{x}_s^{(q)}) = \theta^{(q)}(\bar{\lambda}^{(q)}). \tag{4–43}$$

By the weak duality [47], for any λ feasible to the dual problem of the MP, $\theta(\lambda)$ is an upper bound for the optimal objective value of the MP. In particular, consider $\bar{\lambda}^{(q)}$, which is optimal to the dual of the q^{th}-RMP and feasible to the dual of the MP. $\theta(\bar{\lambda}^{(q)})$ is an upper bound of $\sum_{s\in S} U_s(x_s^*)$, that is,

$$\theta(\bar{\lambda}^{(q)}) \geq \sum_{s\in S} U_s(x_s^*). \tag{4–44}$$

By inspecting the dual functions (4–38) and (4–7) of the q^{th}-RMP and the MP, respectively, we note that $\bar{x}^{(q)}$ is the unique Lagrangian maximizer at $\bar{\lambda}^{(q)}$ for both (4–38) and (4–7). By the

definitions of the dual functions,

$$\begin{aligned} & \theta(\bar{\lambda}^{(q)}) - \theta^{(q)}(\bar{\lambda}^{(q)}) \\ = & \max_{\alpha \geq 0,\ \sum_{i=1}^{Q} \alpha_i = 1} \{\sum_{i=1}^{Q} \alpha_i (\sum_{e \in E} \bar{\lambda}_e^{(q)} c_e^{(i)})\} - \max_{\alpha \geq 0,\ \sum_{i=1}^{q} \alpha_i = 1} \{\sum_{i=1}^{q} \alpha_i (\sum_{e \in E} \bar{\lambda}_e^{(q)} c_e^{(i)})\} \\ = & \gamma(\bar{\lambda}^{(q)}) - \gamma^{(q)}(\bar{\lambda}^{(q)}). \end{aligned}$$

In the last equality, we have used (4–15) and (4–39). Hence, the gap between the upper and lower bounds for the optimal objective value of the MP is $\gamma(\bar{\lambda}^{(q)}) - \gamma^{(q)}(\bar{\lambda}^{(q)})$, which is exactly the difference between the global maximum cost and the local maximum cost of the schedule under $\bar{\lambda}^{(q)}$. Therefore, we conclude the following fact.

Lemma 7. *Let $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$ denote one of the optimal primal-dual solutions of the q^{th}-RMP. $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$ is optimal to the MP if and only if $\gamma(\bar{\lambda}^{(q)}) = \gamma^{(q)}(\bar{\lambda}^{(q)})$.*

4.4.4 Introduce One More Extreme Point (Column or Schedule)

If the gap between the upper and lower bound, $\gamma(\bar{\lambda}^{(q)}) - \gamma^{(q)}(\bar{\lambda}^{(q)})$, is not narrow enough, then $\mathcal{C}$ is not sufficiently well characterized by $\mathcal{C}^{(q)}$ and a new extreme point should be added to the RMP. We state the rule of introducing a new column in the following.

Fact 14. *Any schedule achieving a cost greater than the local maximum cost of the schedule could enter the subset $\mathcal{C}^{(q)}$ in the RMP. The schedule achieving the global maximum cost of the schedule is one possible candidate and is often preferred.*

Lemma 7 says, at the current link cost $\bar{\lambda}^{(q)}$, if none of the schedules that achieve the global maximum cost of the schedule are in the subset $\mathcal{C}^{(q)}$, then the current optimal solution of the q^{th}-RMP is not optimal for the MP. In this case, there are reasons to prefer the introduction of the globally optimal schedule specified by (4–15) as the new extreme point to the restricted master problem. This strategy is a local greedy approach to improve the lower bound of the optimal value of the MP. In fact, it can be viewed as a conditional gradient method for optimizing the lower bound, when the lower bound is viewed as a function of c [7].

4.4.5 Column Generation by Imperfect Global Scheduling

The global scheduling problem (4–15) is usually NP-hard, which makes the step of column generation very difficult. However, according to Fact 14, we do not have to solve it precisely. Instead, we may solve it approximately, and this is referred to as *imperfect global scheduling* [11]. [2]

Suppose we are able to solve (4–15) with an approximation ratio $\rho \geq 1$, that is,

$$\gamma(\lambda) \leq \rho\gamma_\rho(\lambda), \tag{4–45}$$

where $\gamma_\rho(\lambda)$ is the cost of the schedule given by the approximate solution. Note that both $\gamma(\lambda)$ and $\gamma_\rho(\lambda)$ are non-negative for all vectors $\lambda \geq 0$.

A ρ-approximation approach.

We develop a column generation method with imperfect global scheduling as follows.

Algorithm 6 Column Generation with Imperfect Global Scheduling

- Initialize: Start with a collection of q schedules
- Step 1: Run the slow timescale update (4–17)-(4–18) (which will call the fast timescale algorithm) for several (a finite number) times on the q^{th}-RMP.
- Step 2: Solve the global scheduling problem (4–15) *with approximation ratio* ρ under the current dual cost λ.
  0. If the schedule corresponding to the *approximate solution* of (4–15) is already in the current collection of schedules, go to Step 1;
  0. otherwise, introduce this schedule into the current collection of schedules, increase q by 1, and go to Step 1.

We make several comments regarding Algorithm 6.

- If the approximate schedule derived in step 2 has a lower schedule cost than that of an existing schedule already selected, we define the existing schedule with the highest cost as the solution to the approximation algorithm. Hence, the cost of the imperfect (approximate) schedule cannot be lower than any of the existing schedules.

[2] Note that the local scheduling problem (4–39) can be easily solved precisely since the number of extreme points of $\mathcal{C}^{(q)}$ is usually small, and hence, enumerable.

- In the worst case, the column generation method may bring in all the extreme points. However, it often happens that, within a relatively small number of column-generation steps, the optimal solution to the MP is already in $\mathcal{C}^{(q)}$. Thus, the original problem may be solved without generating all the extreme points [7].

- Our focus here is on approximation algorithms because we will be able to show guaranteed performance bound on the MP problem later. Other types of imperfect scheduling can also be used, including many heuristics algorithms and random search algorithms. Examples of the latter include genetic algorithms and simulated annealing [102].

- Note that since the number of extreme points of $\mathcal{C}^{(q)}$ is usually small and enumerable, it is possible for the nodes in the network to store the current collection of schedules. In order to compute the cost of each known schedule in each slow timescale update, each link e can independently compute its corresponding term for each known schedule based on the local link dual cost. Then, those components of the schedule cost can be collected by some controller elected by the nodes in the network. The controller can compute the cost of each known schedule, the locally most costly schedule, update the time fractions by (4–18), and broadcast the results. Other than that, the two-timescale algorithm (4–13)-(4–14) and (4–17)-(4–18) on the q^{th}-RMP is completely decentralized. Furthermore, if the global scheduling problem (4–15) can be solved approximately in a decentralized fashion, then Algorithm 6 is completely decentralized except the part of the controller. In Section 4.5, we will introduce one interference model, under which (4–15) can be solved in a decentralized fashion approximately [10, 11].

- Algorithm 6 in fact describes a whole class of algorithms. To see this, consider the special case where $\rho = 1$ (i.e., the case of perfect global scheduling). In one end of the spectrum, if the slow-timescale algorithm in step 1 runs only once on the RMP, the algorithm becomes a pure two-timescale algorithm as in Section 4.3. In the other end of the spectrum, if the slow-timescale algorithm runs on the RMP until convergence, the algorithm becomes a pure column generation method with the two-timescale algorithm as a building block for solving the restricted problems between consecutive column generation steps. By choosing different numbers of times to run the slow-timescale algorithm in step 1, we have many algorithms, representing different performance, convergence speed and complex tradeoffs.

Convergence with imperfect global scheduling.

Theorem 15. *Under the condition that the fast timescale optimization in the two-timescale algorithm can be regarded as being instantaneous, there exists a* q, $1 \leq q \leq Q$, *such that Algorithm 6 converges to one optimal primal-dual solution of this particular* q^{th}*-RMP (i.e.,* $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$*). Furthermore, after Algorithm 6 converges to* $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$, $\gamma_\rho(\bar{\lambda}^{(q)}) = \gamma^{(q)}(\bar{\lambda}^{(q)})$.

Proof. Since the fast timescale algorithm is assumed to converge instantaneously, we only need to consider the slow timescale algorithm and the column generation steps. Since the number of extreme points of $\mathcal{C}$ is finite, eventually Algorithm 6 will stop introducing new extreme points. Hence, there exists a q, $1 \leq q \leq Q$, such that, after Algorithm 6 stops introducing new extreme points, the number of extreme points that have been introduced is q. Let the convex hull formed by these q points be denoted by $\mathcal{C}^{(q)}$. After Algorithm 6 no longer introduces new extreme points, it behaves just like the two-timescale algorithm but on the restricted set $\mathcal{C}^{(q)}$. According to the theorems in Section 4.3, the two-timescale algorithm converges. Thus, Algorithm 6 converges to $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$ on this particular q^{th}-RMP.

We next show that, after Algorithm 6 converges to $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$, we have $\gamma_\rho(\bar{\lambda}^{(q)}) = \gamma^{(q)}(\bar{\lambda}^{(q)})$. First, note that $\gamma_\rho(\bar{\lambda}^{(q)}) \geq \gamma^{(q)}(\bar{\lambda}^{(q)})$ by the comment after Algorithm 6. Next, it must be true that $\gamma_\rho(\bar{\lambda}^{(q)}) \leq \gamma^{(q)}(\bar{\lambda}^{(q)})$. Otherwise, the schedule whose cost is $\gamma_\rho(\bar{\lambda}^{(q)})$ must not have already been in $\mathcal{C}^{(q)}$ and will be selected to enter. This violates the assumption that the algorithm never selects more than q schedules. □

Performance bound under imperfect scheduling.

Theorem 15 says that the column generation method with imperfect global scheduling converges to a sub-optimum of the MP. Next, we will prove that the performance of this sub-optimum is bounded.

Theorem 16 (Bound of Imperfect Global Scheduling)**.** *Under assumption A2, if the column generation method with imperfect global scheduling converges to* $(\bar{x}^{(q)}, \bar{\alpha}^{(q)}, \bar{\lambda}^{(q)})$ *on the* q^{th}*-RMP, we have*

$$\theta^{(q)}(\bar{\lambda}^{(q)}) \leq \sum_{s \in S} U_s(x_s^*) \leq \theta(\bar{\lambda}^{(q)}) \leq \rho\theta^{(q)}(\bar{\lambda}^{(q)}). \tag{4–46}$$

Proof. Since the q^{th}-RMP is more restricted than the MP, we have $\theta^{(q)}(\bar{\lambda}^{(q)}) \leq \sum_{s \in S} U_s(x_s^*)$. By the weak duality, we have $\sum_{s \in S} U_s(x_s^*) \leq \theta(\bar{\lambda}^{(q)})$.

By the definition of the dual function for the MP in (4–7), we have

$$\begin{aligned}
& \theta(\bar{\lambda}^{(q)}) \\
= & \max_{x \geq 0}\{\sum_{s\in S}(U_s(x_s) - x_s \sum_{e\in p_s} \bar{\lambda}_e^{(q)})\} + \gamma(\bar{\lambda}^{(q)}) \\
\leq & \rho \max_{x \geq 0}\{\sum_{s\in S}(U_s(x_s) - x_s \sum_{e\in p_s} \bar{\lambda}_e^{(q)})\} + \rho\gamma_\rho(\bar{\lambda}^{(q)}) \\
= & \rho \max_{x \geq 0}\{\sum_{s\in S}(U_s(x_s) - x_s \sum_{e\in p_s} \bar{\lambda}_e^{(q)})\} + \rho\gamma^{(q)}(\bar{\lambda}^{(q)}) \\
= & \rho\theta^{(q)}(\bar{\lambda}^{(q)}).
\end{aligned}$$

The first inequality holds because, under assumption $A2$, $\max_{x\geq 0}\{\sum_{s\in S}(U_s(x_s) - x_s \sum_{e\in p_s} \bar{\lambda}_e^{(q)})\} \geq 0$ for any λ (which can be checked by plugging in $x_s = 0$ for all s), $\rho \geq 1$, and (4–45) is assumed. The second equality holds because $\gamma_\rho(\bar{\lambda}^{(q)}) = \gamma^{(q)}(\bar{\lambda}^{(q)})$ by Theorem 15. □

Since the strong duality holds on the q^{th}-RMP, $\sum_{s\in S} U_s(\bar{x}_s^{(q)}) = \theta^{(q)}(\bar{\lambda}^{(q)})$, we have the following.

Corollary 4 (ρ-Approximation Solution to the MP)**.** *Under the assumption $A2$, we have*

$$\sum_{s\in S} U_s(\bar{x}_s^{(q)}) \leq \sum_{s\in S} U_s(x_s^*) \leq \rho \sum_{s\in S} U_s(\bar{x}_s^{(q)}). \tag{4–47}$$

If $\rho = 1.0$, (4–47) holds with equality, then Algorithm 6 is the column generation method with perfect global scheduling, and this algorithm converges to one optimum of MP.

Corollary 4 says that the column generation method with imperfect global scheduling converges to a sub-optimum of the MP and achieves the same approximation ratio as the approximate solution to the global scheduling problem. Finally,

Corollary 5 (Convergence under Perfect Scheduling)**.** *Assume $A2$. Let $\rho = 1$ in Algorithm 6, which corresponds to perfect global scheduling. Then, Algorithm 6 converges to an optimum of the MP.*

In [10, 11], the authors propose a way to solve this problem by a distributed subgradient algorithm with imperfect scheduling. With perfect scheduling, their approach guarantees the convergence of the link dual costs and the primal source rates; but it does not recover the time

share fraction of the schedules, which oscillates due to the limitation of subgradient algorithm. However, with imperfect scheduling, their approach does not guarantee the convergence. Their performance bounds are not of the constant approximation ratio type, and they are dependent of the utility function. In contrast, our Algorithm 3 guarantees the convergence of the link dual costs, the source rates and the time share proportions; and it converges to a sub-optimal solution whose function value is no less than a constant fraction of the true optimum value. The constant is independent of the utility function.

4.5 Performance Evaluation

In this section we will show the performance of our algorithm by simulation. We will use the following node exclusive interference model. The model requires that, first, the data rate of each link is fixed at c_e; and second, at any time instance, each node can only send to or receive from one other node. Under this model, the scheduling problem (4–15) becomes the *maximum weighted matching (MWM)* problem [10, 11, 103]. There is a polynomial-time algorithm to solve MWM precisely [104] and a greedy algorithm to solve it approximately with an approximation ratio $\rho = 2$. The greedy algorithm is more useful to our problem because it is distributed [11]. Under this model, our column generation algorithm with imperfect scheduling will converge to an approximate solution for the MP with an approximation ratio $\rho = 2$, and it is completely decentralized.

We remark that the node exclusive interference model is a simple instance of the conflict-graph-based models that capture the contention relations among the links [9, 12]. In a conflict graph, each vertex represents one wireless link in the network, and an edge represents contention between the two corresponding links, which are not allowed to transmit at the same time. A set of links in the wireless network that can transmit data simultaneously (i.e., a schedule), is an independent set in the corresponding conflict graph. The scheduling problem (4–15) becomes the *maximum weighted independent set (MWIS)* problem, where the edge weight is $\lambda_e c_e$. Generally, MWIS has no approximate solution. Some heuristics or random search algorithms seem necessary to carry out the imperfect scheduling.

The possible choices of utility function $U_s(x_s)$ could be

$$U_s(x_s) = w_s \ln(x_s + e) \tag{4–48}$$

or

$$U_s(x_s) = w_s \frac{(x_s + a_s)^{1-\beta}}{1-\beta}, \quad 0 < \beta < 1, \tag{4–49}$$

where w_s are the weights for $s \in S$, e is the base of the natural logarithm and $a_s > 0$ is a small constant, which make the utility functions (4–48) and (4–49) satisfy the assumptions $A2$ and $A3$. These utility functions have been discussed in [42]. In this chapter, we will use the utility function in (4–48) with $w_s = 1.0$ for all $s \in S$.

As discussed in Section 4.4, we can introduce new extreme points at varying degree of frequency. In the experiments, we will use three frequencies: fast, medium and slow. With the fast frequency, we try to introduce extreme points by solving the global scheduling problem (4–15) at each slow-timescale update (4–17)-(4–18), in which case, Algorithm 6 degenerates into the pure two-timescale algorithm. With the slow frequency, we try to introduce a new extreme point after every 20 slow-timescale updates of (4–17)-(4–18). Our experiences have shown that the restricted master problem with our experiment sizes is often optimized within 20 slow-timescale updates. If so, Algorithm 6 becomes the pure column generation method. With the medium frequency, we introduce a new extreme point every 5 slow-timescale updates.

The network in Fig. 4-1 has been studied in [10, 11]. There are 5 classes of connections as shown in Fig. 4-1. The capacity of each link is fixed at 100 units. We initialize the experiments with a set of schedules, where each contains exactly one single transmitting link. This corresponds to the traditional TDMA scheduling [17]. Fig. 4-2 shows the convergence of the connection rates with perfect scheduling and imperfect scheduling, respectively, where both are introducing new columns at the fast frequency. In Fig. 4-2 (a), we have two groups of connections. Class 4 and Class 5 achieve higher rates because they involve less wireless interference compared with others. Fig. 4-2 (b) gives the same order of the connections in terms of their rates. But, the connections are not separated into obvious rate groups. Though the two scheduling

schemes do not give the exactly same connection rates, their final objective function values are very close: 16.0989 for the imperfect scheduling and 16.1351 for the perfect one. We note that with our specific objective function in (4–49), a minor change in the connection rates will not change the objective too much. Fig. 4-3 shows the two schemes get the correct time fraction and the long time average link capacities are able to support the source flow rates. It means our two-timescale algorithm solves both the primal and dual problems at the same time.

We next experiment with a larger network with 15 nodes. The network is randomly generated and 20 end-to-end connections are placed on this network randomly. For each connection, the routing is the fixed shortest path routing. In the experiment, it turns out these 20 connections use 28 directed links. The capacity of each link is fixed at 100 units. Fig. 4-4 shows the 5 connections with the highest rates. Again, the perfect scheduling is more likely to group connections.

Next, we evaluate the algorithm with different frequencies of introducing columns on the large network. In Fig. 4-5, with both perfect and imperfect scheduling, the fast scheme always improves the objective function value more quickly at the beginning, while the slow scheme improves it much more slowly than the other two schemes. The reason is that, with the fast scheme, plenty of schedules are introduced quickly. The slow scheme always tries to take full advantage of the current collection of schedules. But later, the slow scheme catches up the fast scheme, judging from the trend of the curves. This motivates the use of the medium scheme. In Fig. 4-5, we see that the medium scheme increases the objective function value nearly as quickly as the fast scheme at the beginning and it surpasses the fast scheme soon after. The curves show some oscillations at the initial phase for the medium and slow schemes. This is because those two schemes spend more effort to obtain better performance from the current collection of schedules. At the initial phase, with fewer schedules but more optimized time sharing, introducing one more schedule abruptly will decrease the function value by a little bit.

In Table 4-1, we compare the three schemes for their computation costs. Since the most expensive computation is for solving the global scheduling problem (4–15), the total computation

time is mainly characterized by the number of times the global scheduling problem is solved. One expects that lowering the frequency of introducing new schedules is correlated with fewer computations for the global scheduling problem. But, we know no theoretical reasons why this must be true. The result confirms the expectation: The number of such computations is 300, 60, and 15, for the fast, medium and slow schemes. The reduction is dramatic.

We also find, with a lower frequency, the algorithm usually produces a solution with fewer active schedules. [3] Fewer active schedules may be desirable since it is easier to manage and control them, which may reduce the system complexity and control overhead. With the perfect scheduling, the slow scheme (i.e., the pure column generation approach) only uses (i.e., time-share) 15 active schedules in the end, which are all those that were ever computed and entered. In other words, there are no redundant schedules; nor are there redundant computations for the schedules. The fast and medium schemes use 49 active schedules. In the fast scheme, 7 schedules have been introduced into the collection but are not used in the final optimal solution. In the medium scheme, the number of redundant schedules is 3.

For the imperfect scheduling, we find that both the fast and the medium schemes generate much fewer schedules than in the perfect scheduling, although the number of computations for the schedules remain the same[4] . The fast scheme even has fewer redundant schedules than the medium scheme, which a little counter-intuitive. The reason might be that the approximation algorithm is not as sensitive to the change of link prices as the precise algorithm. Significant changes in link prices are needed to trigger the discovery of a new schedule.

Based on the study of Fig. 4-4 and Table 4-1, we conclude that the pure two-timescale (fast) or the pure column generation (slow) algorithms have both pros and cons. An intermediate

[3] In these 6 experiments, the initial TDMA-style schedules are all inactive in the optimal solutions, and we did not count them in the table.

[4] However, each computation is less expensive than in the perfect scheduling case, since it is approximate.

algorithm (medium) may achieve a more desirable balance among factors such as optimization performance, the computational cost, and system complexity and overhead.

Next, we show that, in the pure column generation method, the gap between the lower and upper bounds for the optimal object value decreases as the restricted master problem expands. With the imperfect scheduling, we can compute the upper bound by $\theta^{(q)}(\bar{\lambda}^{(q)}) - \gamma^{(q)}(\bar{\lambda}^{(q)}) + \gamma(\bar{\lambda}^{(q)}) \leq \theta^{(q)}(\bar{\lambda}^{(q)}) - \gamma^{(q)}(\bar{\lambda}^{(q)}) + \rho\gamma_{\rho}(\bar{\lambda}^{(q)})$, where $\rho = 2$ in our case. The lower bound is obtained from the current best solution. Fig. 4-6 shows that the gap is quickly narrowed after 10 columns have entered. It also shows that the objective values of both the perfect scheduling and imperfect scheduling are inside the two bounds. Also, our imperfect scheduling almost achieves the global optimum of the original problem.

Finally, we have also applied the subgradient algorithms for these experiments, and found that it is very difficult to tune the algorithm parameters to reach convergence.

4.6 Conclusions

This chapter studies the problem of how to allocate wireless resources to maximize the aggregate source utility. This optimization problem has two difficulties: First, the Lagrangian function is not strictly concave with respect to the time-share variables, which makes the subgradient algorithm unable to recover the optimal values for those variables; second, its constraint set is a convex polytope usually containing an exponential number of extreme points. In order to recover the correct time-share variables, we develop a two-timescale algorithm. To overcome the difficulty of the global scheduling problem, we adopt a column generation approach with imperfect global scheduling. If the imperfect scheduling has bounded performance, then our overall utility optimization algorithm converges to a sub-optimum with bounded performance. The combination of the two-timescale algorithm and column generation leads to a family of algorithms with interesting tradeoffs.

Table 4-1. Performance comparison of the family of algorithms (large network).

	#Schedules Computed	#Active Schedules	#Schedules Introduced
Fast Per.	300	49	56
Medium Per.	60	49	52
Slow Per.	15	15	15
Fast Imper.	300	19	22
Medium Imper.	60	19	30
Slow Imper.	15	15	15

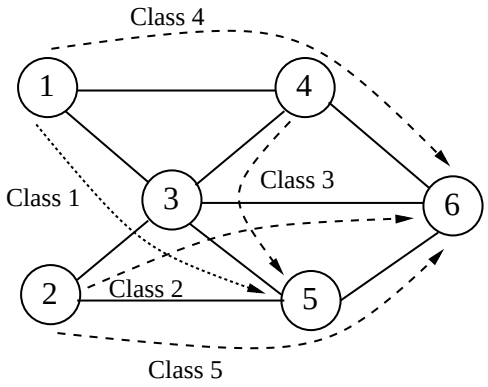


Figure 4-1. Small network topology.

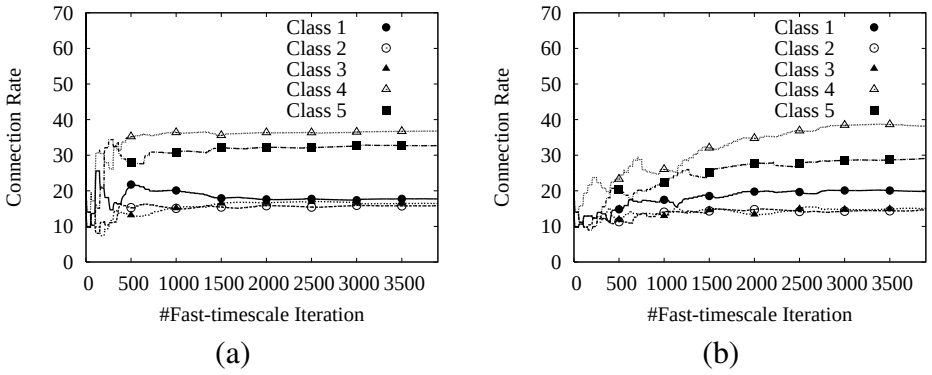


Figure 4-2. Convergence in connection rates (small network): (a) fast frequency, with perfect global scheduling; (b) fast frequency, with imperfect Global Scheduling.

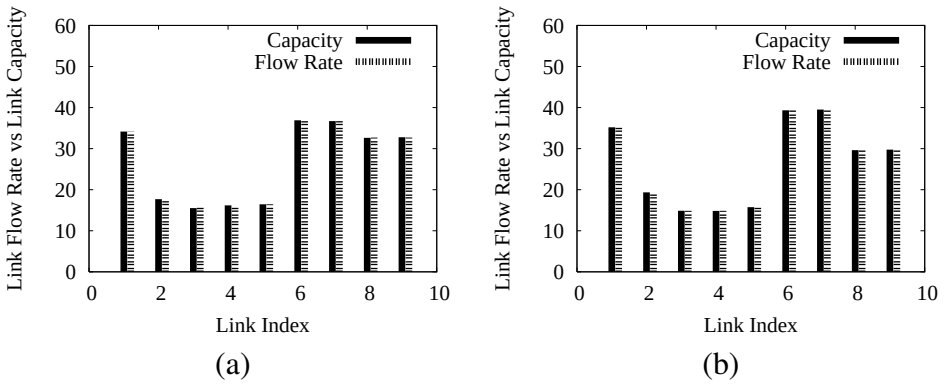


Figure 4-3. Convergence in link flow rates (small network): (a) fast frequency, with perfect global scheduling; (b) fast frequency, with imperfect global scheduling.

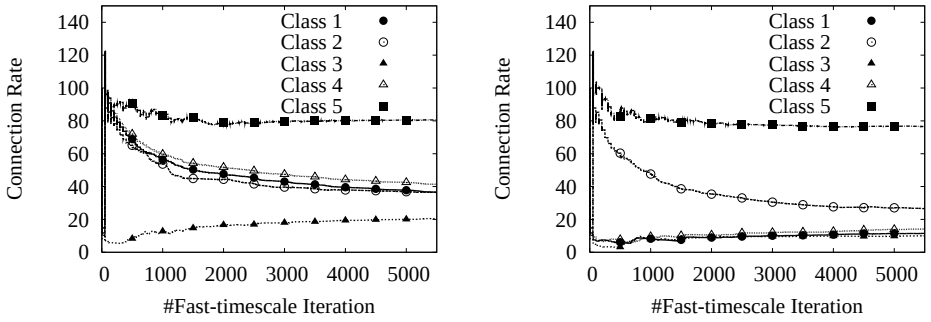


Figure 4-4. Convergence in connection rates (large network): (a) fast frequency, with perfect global scheduling; (b) fast frequency, with imperfect global scheduling.

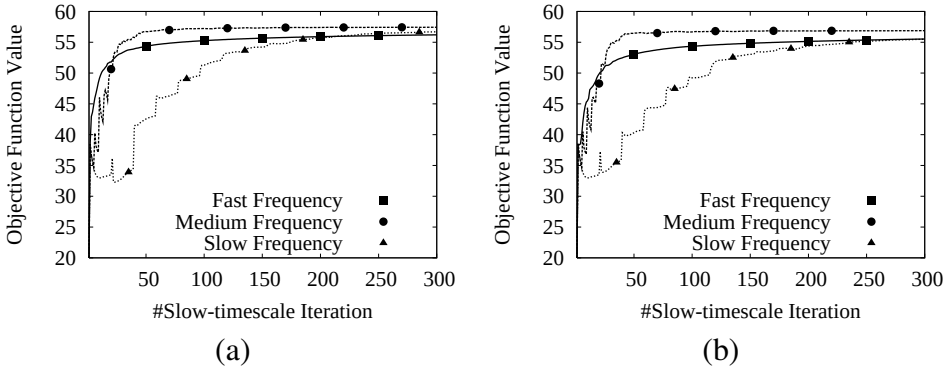


Figure 4-5. Convergence of the family of algorithms (large network): (a) perfect global scheduling; (b) imperfect global scheduling.

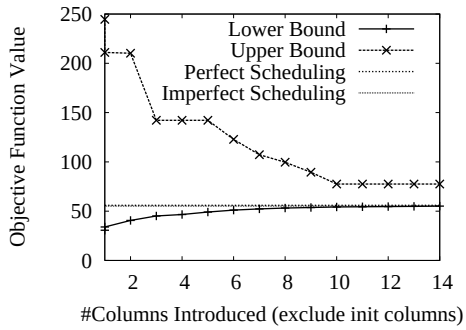


Figure 4-6. Bounds for the optimal objective value of the MP. Pure column generation method with imperfect global scheduling.

REFERENCES

[1] *BitTorrent*, BitTorrent Inc., 201 Mission Street, Suite 900, San Francisco, CA 94105, http://www.bittorrent.com/.

[2] *Kazaa*, Sharman Networks Ltd., http://www.kazaa.com.

[3] *Gnutella*, Gnutelliums LLC., http://www.gnutellaforums.com.

[4] J. W. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads," in *Proceedings of the IEEE Infocom 1999*, New York, NY, March 1999.

[5] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery Across Adaptive Overlay," in *Proceedings of ACM Sigcomm 2002*, Pittsburgh, PA, August 2002.

[6] "Multicast from wikipedia," Wikimedia Foundation, Inc., http://en.wikipedia.org/wiki/Multicast.

[7] M. Johansson and L. Xiao, "Cross-layer optimization of wireless networks using nonlinear column generation," *IEEE Transaction on Wireless Communications*, vol. 5, no. 2, pp. 435–445, Feb. 2006.

[8] J. Wang, L. Li, S. H. Low, and J. C. Doyle, "Cross-layer optimization in TCP/IP networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 582 – 595, June 2005.

[9] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proceedings of the IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[10] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[11] ——, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," *IEEE/ACM Transaction on Networking*, vol. 14, no. 2, pp. 302–315, April 2006.

[12] S. Bohacek and P. Wang, "Toward tractable computation of the capacity of multihop wireless networks," in *Proceedings of INFOCOM*, May 2007.

[13] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "Joint congestion control and distributed scheduling for throughput guarantees in wireless networks," in *Proceedings of INFOCOM*, May 2007.

[14] P. Bjorklund, P. Varbrand, and D. Yuan, "Resource optimization of spatial TDMA in ad hoc radio networks: a column generation approach," in *Proceedings of INFOCOM*, 2003.

[15] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for ad hoc wireless networks," in *Proceedings of the IEEE Infocom 2005*, Miami, USA, March 2005.

[16] H. Zhai and Y. Fang, "Impact of routing metrics on path capacity in multi-rate and multi-hop wireless ad hoc networks," in *Proceedings of The 14th IEEE International Conference on Network Protocols (ICNP'06)*, 2006.

[17] S. Kompella, J. E. Wieselthier, and A. Ephremides, "A cross-layer approach to optimal wireless link scheduling with SINR constraints," in *MilCom 2007*, 2007.

[18] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *Wireless Internet, 2005. Proceedings. First International Conference on*, July 2005, pp. 47–54.

[19] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

[20] M. Chiang and S. Low, "Optimization and control of communication networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 277–277, 2005.

[21] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1426–1438, August 2006.

[22] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, pp. 73–85, January 1977.

[23] D. Bertsekas and E. Gafni, "Projected newton methods and optimization of multicommodity flows," *Automatic Control, IEEE Transactions on*, vol. 28, no. 12, pp. 1090–1096, Dec 1983.

[24] D. Bertsekas, E. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 32, no. 8, pp. 911–919, Aug 1984.

[25] E. M. Gafni and D. P. Bertsekas, "Asymptotic optimality of shortest path routing algorithms," *IEEE Trans. Inf. Theor.*, vol. 33, no. 1, pp. 83–90, 1987.

[26] L. K. L. Fratta, M. Gerla, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, pp. 97–133, 1973.

[27] M. Florian, "An improved linear approximation algorithm for the network equilibrium (packet switching) problem," *Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications, 1977 IEEE Conference on*, vol. 16, pp. 812–818, Dec. 1977.

[28] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, pp. 149–154, 1956.

[29] D. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[30] D. P. Bertsekas, "Projected newton methods for optimization problems with simple constraints," *Decision and Control including the Symposium on Adaptive Processes, 1981 20th IEEE Conference on*, vol. 20, pp. 762–767, Dec. 1981.

[31] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow price, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[32] S. H. Low and D. E. Lapsley, "Optimization flow control - I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[33] F. Kelly, "Charging and rate control for elastic traffic," 1997. [Online]. Available: http://citeseer.ist.psu.edu/kelly97charging.html

[34] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 689–702, Octobor 2003.

[35] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proceedings of the 43rd IEEE CDC*, 2004.

[36] S. H. Low, "Optimization flow control with on-line measurement or multiple paths," in *Proceedings of the 16th International Teletraffic Congress*, Edinburgh, Scotland, June 1999.

[37] A. Sankar and Z. Liu, "Maximum lifetime routing in wireless ad-hoc networks," in *Proceedings of the IEEE Infocom 2004*, Hong Kong, March 2004.

[38] Y. Xue, Y. Cui, and K. Nahrstedt, "Maximizing lifetime for data aggregation in wireless sensor networks," *Mobile Networks and Applications*, vol. 10, no. 6, pp. 853–864, December 2005.

[39] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697 – 716, August 2003.

[40] H. Nama, M. Chiang, and N. Mandayam, "Utility lifetime tradeoff in self regulating wireless sensor networks: A cross-layer design approach," in *Proc. IEEE ICC*, June 2006.

[41] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, August 2003.

[42] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.

[43] Y. Xue, B. Li, and K. Nahrsted, "Price-based resource allocation in wireless ad hoc networks," in *Proceedings of the 11th International Workshop on Quality of Service (IWQoS), also in Lecture Notes in Computer Science, vol. 2707*, June 2003.

[44] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[45] A. Khreishah, C.-C. Wang, and N. B. Shroff, "Optimization based rate control for communication networks with inter-session network coding," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[46] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The impact and implications of the growth in residential user-to-user traffic," in *Proceedings of ACM Sigcomm*, Pisa, Italy, September 2006.

[47] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[48] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the Association for Computing Machinery*, vol. 37, no. 2, pp. 318–334, April 1990.

[49] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528– 1540, Octobor 2002.

[50] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Wireless Communications*, June 2006.

[51] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Method*. Athena Scientific, 1997.

[52] R. Madan, Z. Luo, and S. Lall, "A distributed algorithm with linear convergence for maximum lifetime routing in wireless networks," in *Proceedings of the Allerton Conference on Communication, Control, and Computing*, September 2005.

[53] Z.-Q. Luo and P. Tseng, "On the rate of convergence of a distributed asynchronous routing algorithm," *IEEE Transcations on Automatic Control*, vol. 39, pp. 1123–1129, May 1994.

[54] J. N. Tsitsiklis and D. P. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Transactions On Automatic Control*, vol. AC-31, pp. 325–332, 1986.

[55] M. Hefeeda, A. Habib, B. Boyan, D. Xu, and B. Bhargava, "Promise: peer-to-peer media streaming using collectcast," in *Proc. of ACM Multimedia*, Berkeley, CA, November 2003.

[56] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S.-Q. Yang, "Large-scale live media streaming over peer-to-peer networks through global internet," in *Proceedings of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*, 2005.

[57] R. Rejaie and S. Stafford, "A framework for architecting peer-to-peer receiver-driven overlays," in *Proceedings of NOSSDAV 2004*, June 2004.

[58] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proceedings of ACM SIGMETRICS 2004*, June 2004.

[59] L. Cherkasova and J. Lee, "Fastreplica: Efficient large file distribution within content delivery networks," in *Proceedings of the 4th USITS*, Seattle, WA, March 2003.

[60] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, October 2003.

[61] D. Kostić, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat, "Maintaining high bandwidth under dynamic network conditions," in *Proceedings of USENIX Annual Technical Conference*, 2005.

[62] B.-G. Chun, P. Wu, H. Weatherspoon, and J. Kubiatowicz, "ChunkCast: An anycast service for large content distribution," in *Proceedings of the Internaltional Workshop on Peer-to-Peer Systems (IPTPS)*, February 2006.

[63] K. Park and V. S. Pai, "Scale and performance in the CoBlitz large-file distribution service," in *Proceedings of the 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[64] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, October 2003.

[65] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," in *Proceedings of IEEE Infocom*, Hong Kong, March 2004.

[66] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Sigcomm'04*, 2004.

[67] D. Bickson, D. Malkhi, and D. Rabinowitz, "Efficient large scale content distribution," in *Proceedings of the 6th Workshop on Distributed Data and Structures (WDAS'2004)*, Lausanne, Switzerland, July 2004.

[68] P. A. Padmavathi Mundur, "Optimal server allocations for streaming multimedia applications on the Internet," *Computer Networks*, vol. 50, pp. 3608–3621, 2006.

[69] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, and D. Yao, "Optimal peer selection for P2P downloading and streaming," in *Proceedings of IEEE Infocom*, Miami, FL, March 2005.

[70] R. L. Carter and M. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of IEEE Infocom*, 1997, pp. 1014–1021.

[71] P. Rodriguez, A. Kirpal, and E. Biersack, "Parallel-access for mirror sites in the internet," in *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 2000.

[72] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of IEEE Infocom*, Anchorage, AK 2001.

[73] E. B. P. Rodriguez, "Dynamic parallel access to replicated content in the internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 455–465, August 2002.

[74] J. Lee and G. de Veciana, "On application-level load balancing in FastReplica," *Computer Communications*, vol. 30, no. 17, pp. 3218–3231, November 2007.

[75] K. Wieland, "DigiWorld 2006: Voice and low prices drive FTTH in Japan," *Telecommunications Online*, Nov. 15 2006, http://www.telecommagazine.com.

[76] J. George, "FTTH design with the future in mind," *Broadband Properties*, September 2005.

[77] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. Kubiatowicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)* , June 2001.

[78] *Akamai*, Akamai Technologies, http://www.akamai.com.

[79] J. Cannons, R. Dougherty, C. Freiling, and K. Zeger, "Network routing capacity," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 777–788, March 2006.

[80] M. Grotchel, A. Martin, and R. Weismantel, "Packing Steiner trees: a cutting plane algorithm and computation," *Mathematical Programming*, vol. 72, pp. 125–145, 1996.

[81] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing Steiner trees," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '03)*, 2003.

[82] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1991.

[83] R. E. Tarjan, "Finding optimum branchings," *Networks, Vol. 7*, pp. 25–35, 1977.

[84] L. F. P. M. Camerini and F. Maffioli, "A note on finding optimum branchings," *Networks, Vol. 9*, pp. 309–312, 1979.

[85] P. A. Humblet, "A distributed algorithm for minimum weight directed spanning trees," *IEEE Transactions on Communications*, pp. 756–762, June 1983.

[86] *Rocketfuel: An ISP Topology Mapping Engine*, University of Washington, Department of Computer Science Engineering, University of Washington, Box 352350, Seattle, WA 98195-2350, http://www.cs.washington.edu/research/networking/rocketfuel/.

[87] A. R. Bharambe and C. Herley, "Analyzing and improving BitTorrent performance," Microsoft Research, Tech. Rep. MSR-TR-2005-03, 2005.

[88] R. Kumar and K. Ross, "Peer-assisted file distribution: The minimum distribution time," in *IEEE Workshop on Hot Topics in Web Systems and Technologies (HOTWEB)*, 2006.

[89] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, "Can network coding help in P2P networks?" in *The fourth International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2006.

[90] J. Mundinger, R. R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," *arXiv e-print service*, June 2006, http://arxiv.org/abs/cs.NI/0606110.

[91] Y. Wu, P. A. Chou, and K. Jain, "A comparison of network coding and tree packing," in *The Proceedings of IEEE International Symposium on Information Theory (ISIT)*, June 2004.

[92] J. Edmonds, "Edge-disjoint branchings," in *Combinatorical Algorithms*, R. Rustin, Ed. Academic Press, 1973, pp. 91–96.

[93] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proceedings of the IEEE Infocom 2005*, Miami, USA, March 2005.

[94] Y. N. Wu and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: a shortest path approach," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1475– 1488, Aug. 2006.

[95] P. Key, L. Massouliè, and D. Towsley, "Path selection and multipath congestion control," in *Proceedings of INFOCOM 2007*, May 2007.

[96] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," in *Proceedings of the International Conference on Distributed Computing Systems (ICDCS'06)*, 2006.

[97] H. Zhang, G. Neglia, D. Towsley, and G. L. Presti, "On unstructured file sharing networks," in *Proceedings of INFOCOM*, May 2007.

[98] F. Paganini, "Congestion control with adaptive multipath routing based on optimization," in *The 40th Annual Conference on Information Sciences and Systems*, 2006.

[99] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proceedings of IEEE INFOCOM*, 2007.

[100] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Wiley-Interscience, 2006.

[101] H. K. Khalil, *Nonlinear Systems*. Prentice-Hall, 1996.

[102] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of IEEE INFOCOM*, 1998.

[103] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "Maximum weighted matching with interference constraints," in *PERCOMW '06: Proceedings of the 4th annual IEEE*

international conference on Pervasive Computing and Communications Workshops, Washington, DC, 2006.

[104] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *Symposium on Discrete Algorithms*, 1990.

BIOGRAPHICAL SKETCH

Xiaoying Zheng received her bachelor's and master's degrees in computer science and engineering from Zhejiang University, P.R. China, in 2000 and 2003, respectively. Her research interests are in the computer networking area, including applications of optimization theory in networks, peer-to-peer overlay networks, content distribution, wireless ad hoc networks, and congestion control.