Optimizing Network Objectives in Collaborative Content Distribution

Xiaoying Zheng and Ye Xia
Computer and Information Science and Engineering Department
University of Florida
Gainesville, FL 32611-6120
Email: {xiazheng, yx1}@cise.ufl.edu

***Abstract*—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. In this paper, 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.**

***Index Terms*—Content Distribution, Peer-to-Peer Network, Bandwidth Allocation, Congestion Control, Server Selection, Optimization**

I. 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 [1].

An important networking problem addressed by this paper 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 [2]. 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 paper 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 paper, 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 paper 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. 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. [4] and Low et al. [5]. 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 paper 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 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 [6]. 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 paper is organized as follows. In the remaining part of the introduction, we summarize the common notation used throughout the paper. In Section II, 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 III, 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 V, we present experimental results to compare the performance of the subgradient algorithm and the gradient projection algorithm without and with scaling. In Section VI, additional related work is reviewed. The conclusion is in Section VII.

[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 paper 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 [3]. 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.

A. Notation

Let the network be represented by a directed graph $G = (N, E)$, where N 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 $i, j \in N$ are distinct nodes. An arc represents a directed communication link. The link capacity is denoted by u_{ij} for $(i, j) \in E$, or u_e for $e \in E$, interchangeably.

Let $S \subseteq N$ be the set of servers and $C \subseteq N$ 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 $\mu_e = y_e/u_e$. 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.

II. 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 paper 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.

A. 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 paper,

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

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.

$$\textbf{Min-Congestion}: \min \quad \mu \tag{2}$$

$$\text{s.t.} \quad \sum_{P_{ij}\ni e} x_{ij} \le u_e\mu, \quad \forall e \in E \tag{3}$$

$$\sum_{j\in C} x_{ij} \le K_i, \quad \forall i \in S \tag{4}$$

$$\sum_{i\in S} x_{ij} = Q_j, \quad \forall j \in C \tag{5}$$

$$x_{ij} \ge 0, \quad \forall i \in S,\ \forall j \in C. \tag{6}$$

Condition (3) says that the link utilization (i.e., congestion level) at each link e, $(\sum_{P_{ij}\ni e} x_{ij})/u_e$, 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 (4) 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 (5) 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 $\sum_{i\in S} x_{ij} \ge Q_j$ 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 (5) 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 paper 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. 1 as an example.) The formulation can be modified accordingly and the structure of the solution algorithms remains the same.

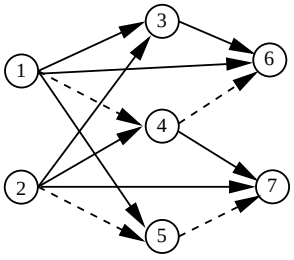


Fig. 1. A server-client dependency graph. The solid and dashed lines indicate server-to-client relationship. The solid lines indicate the final server selection results. For instance, node 6 has the server set $S_6 = \{1, 3, 4\}$. It ends up selecting servers 1 and 3. Node 1 has the client set $C_1 =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

Remark 3 Fig. 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 [7]. Similar to the MCF problem, the Min-Congestion problem has an equivalent throughput-maximization formulation as follows. Let $\gamma = 1/\mu$, and $\tilde{x}_{ij} = x_{ij}/\mu$.

$$\textbf{Max-Throughput}: \max \quad \gamma$$

$$\text{s.t.} \quad \sum_{P_{ij}\ni e} \tilde{x}_{ij} \le u_e, \quad \forall e \in E$$

$$\sum_{j\in C} \tilde{x}_{ij} \le K_i\gamma, \quad \forall i \in S$$

$$\sum_{i\in S} \tilde{x}_{ij} = Q_j\gamma, \quad \forall j \in C$$

$$\tilde{x}_{ij} \ge 0, \quad \forall i \in S,\ \forall j \in C.$$

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.

B. 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 VI.). In this scheme, each client selects a random number of servers. To have fair comparison, we also require that conditions (4) and (5) are satisfied. This complicates the actual sever-selection procedure a little. But the details are not essential.

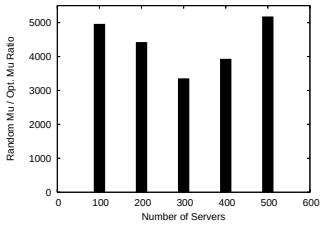
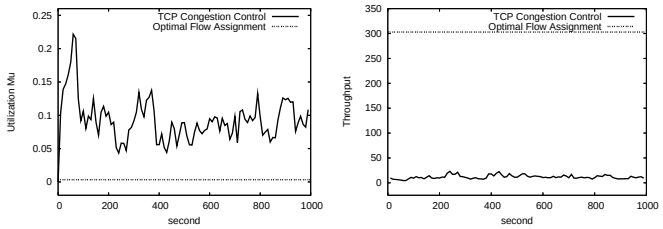


Fig. 2. Ratio of worst-case link utilization: Random vs. optimal flow assignment. The networks have 1000 nodes, 16144 links and 500 clients.

1) 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 [8] 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.



(a) (b)

Fig. 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.

The precise model for the random networks is described in Section V. 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 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.

2) 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. 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.

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

III. 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.

A. Nonlinear Approximation of the Min-Congestion Problem with Barrier

One of the theoretical tools for finding distributed optimization algorithms is convex separable optimization, as has been amply demonstrated by [4], [5], [9]–[13] for network control and resource allocation problems. A convex separable optimization problem on ^{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 [6], which is widely used. However, we will take a different approach. After some approximation steps, we will convert our problem into a nonlinear convex problem with simplex constraints. Such a problem admits a special gradient projection method, which works much better than the subgradient algorithm. The performance comparison between the two algorithms will be given in Section V.

We will first convert the Min-Congestion problem into a nonlinear convex separable form, using a similar approximation as in [14]. For each $e \in E$, let $\mu_e = \sum_{P_{ij} \ni e} x_{ij}/u_e$, 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 (4) as a barrier function. Then, the Min-Congestion problem can be approximated by the following problem.

[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 $Q_j = 10$.

$$\min \quad ||\vec{\mu}||_q^q - \epsilon \sum_{i\in S} \ln(K_i - \sum_{j\in C} x_{ij}) \tag{7}$$

$$\text{s.t.} \quad \sum_{P_{ij}\ni e} x_{ij} = u_e \mu_e, \qquad \forall e \in E \tag{8}$$

$$\sum_{i\in S} x_{ij} = Q_j, \qquad \forall j \in C \tag{9}$$

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

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 (8) is not really constraints but definitions 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 (9) 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 $\mathcal{X} \subset^{|P|}$ the compact set of all feasible flow rates.

$$\mathcal{X} = \{x \mid x_{ij} \geq 0, \forall i \in S, \forall j \in C; \sum_{i\in S} x_{ij} = Q_j, \forall j \in C\}. \tag{10}$$

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

$$f_e^1(x) = (\frac{\sum_{P_{ij}\ni e} x_{ij}}{u_e})^q \tag{11}$$

and

$$\hat{f}_e^1(y_e) = (\frac{y_e}{u_e})^q. \tag{12}$$

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

$$f_i^2(x) = \begin{cases} -\epsilon^q \ln(K_i - \sum_{j\in C} x_{ij}) \\ \qquad \text{if } \sum_{j\in C} x_{ij} \leq K_i - \rho, \\ \epsilon^q((\sum_{j\in C} x_{ij} - (K_i - \rho) + \frac{1}{2\rho})^2 - (\frac{1}{4\rho^2} + \ln \rho)) \\ \qquad \text{if } \sum_{j\in C} x_{ij} > K_i - \rho, \end{cases} \tag{13}$$

and

$$\hat{f}_i^2(z_i) = \begin{cases} -\epsilon^q \ln(K_i - z_i) \\ \qquad \text{if } z_i \leq K_i - \rho, \\ \epsilon^q((z_i - (K_i - \rho) + \frac{1}{2\rho})^2 - (\frac{1}{4\rho^2} + \ln \rho)) \\ \qquad \text{if } z_i > K_i - \rho, \end{cases} \tag{14}$$

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.

$$\min \quad f(x) = f^1(x) + f^2(x) = \hat{f}^1(G_1 x) + \hat{f}^2(G_2 x) \tag{15}$$

$$\text{s.t.} \quad x \in \mathcal{X}. \tag{16}$$

B. The Gradient Projection Algorithm

For optimization problems with the simplex constraint of the form in (16), the optimality condition is especially simple [6]. Furthermore, there exists a special gradient projection algorithm [15] [16]. In this subsection, we describe the optimality condition and the algorithm. In the next subsection, we give the convergence results.

1) 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

$$\frac{\partial f(x)}{\partial x_{ij}} = \sum_{e\in P_{ij}} (\hat{f}_e^1)'(\sum_{P_{kl}\ni e} x_{kl}) + (\hat{f}_i^2)'(\sum_{l\in C} x_{il}), \tag{17}$$

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

$$\frac{\partial^2 f(x)}{\partial x_{ij} \partial x_{kl}} = \sum_{e\in P_{ij}\cap P_{kl}} (\hat{f}_e^1)''(\sum_{P_{kl}\ni e} x_{kl}) + \begin{cases} (\hat{f}_i^2)''(\sum_{l\in C} x_{il}) & \text{if } i = k \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

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 (17), $\frac{\partial f(x)}{\partial x_{ij}}$ 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]

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

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 (18), similar statements as 2) – 4) can be said about the second derivative, $\frac{\partial^2 f(x)}{\partial^2 x_{ij}}$.

The computation of (17) and (18) 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

$$(\hat{f}_e^1)'(y_e) = q u_e^{-q} y_e^{q-1}, \tag{19}$$

$$(\hat{f}_i^2)'(z_i) = \begin{cases} \frac{\epsilon^q}{K_i - z_i} & \text{if } z_i \le K_i - \rho \\ 2\epsilon^q (z_i - (K_i - \rho) + \frac{1}{2\rho}) & \text{if } z_i > K_i - \rho. \end{cases} \tag{20}$$

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

$$(\hat{f}_e^1)''(y_e) = q(q-1) u_e^{-q} y_e^{q-2}, \tag{21}$$

$$(\hat{f}_i^2)''(z_i) = \begin{cases} \frac{\epsilon^q}{(K_i - z_i)^2} & \text{if } z_i \le K_i - \rho \\ 2\epsilon^q & \text{if } z_i > K_i - \rho. \end{cases} \tag{22}$$

2) 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 (15)-(16) by the following optimality condition.

$$\sum_{j \in C} \sum_{i \in S} \frac{\partial f(x^*)}{\partial x_{ij}} (x_{ij} - x_{ij}^*) \ge 0,$$
$$\forall x \ge 0 \text{ with } \sum_{i \in S} x_{ij} = Q_j, \forall j \in C. \tag{23}$$

This condition is equivalent to, for any $i \in S$ and $j \in C$ (See [6].),

$$x_{ij}^* > 0 \text{ only if } [\frac{\partial f(x^*)}{\partial x_{kj}} \ge \frac{\partial f(x^*)}{\partial x_{ij}}, \forall k \in S]. \tag{24}$$

Important insight is contained in (24). 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 (17), 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,

$$s_j(x) = \text{argmin}_{i \in S} \{\frac{\partial f(x)}{\partial x_{ij}}\}. \tag{25}$$

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

3) The Synchronous Gradient Projection Algorithm: Based on a similar algorithm in [15], we apply the synchronous iterative gradient projection algorithm listed in Algorithm 1 to solve the problem (15)-(16).

Here, $\delta > 0$ is a scalar step size, and $\alpha(t)$ in (26) is a scalar on $[\underline{a}, 1]$, for some $\underline{a}$, $0 < \underline{a} \le 1$. Given the current $x(t)$, (27) and (28) compute the end point, $\bar{x}(t)$, of a feasible direction, entry by entry. For each client j, there are two cases.

Algorithm 1 Synchronous Gradient Projection Algorithm

$$x(t+1) = \alpha(t)\bar{x}(t) + (1 - \alpha(t))x(t), \tag{26}$$

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

$$\bar{x}_{ij}(t) = [x_{ij}(t) - \delta(\frac{\partial f(x(t))}{\partial x_{ij}(t)} - \frac{\partial f(x(t))}{\partial x_{s_j(x(t)),j}(t)})]_+ \tag{27}$$

and

$$\bar{x}_{s_j(x(t)),j}(t) = Q_j - \sum_{i \in S, i \ne s_j(x(t))} \bar{x}_{ij}(t). \tag{28}$$

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 $P_{i,j}$ 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 $P_{i,j}$ 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 (26), 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 (19) and (20), 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 (17). Hence, the gradient projection algorithm is completely decentralized.

C. The Analysis of Convergence

In this subsection, we will adapt the results from [17] 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 (15)-(16). 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.

Assume that there is at least one feasible flow assignment. Define a set $\mathcal{X}_0 = \{x \in \mathcal{X} \mid f(x) \le \eta\}$ 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,

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

[6](26) 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 $\underline{a}$, $0 < \underline{a} \le 1$. 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 $A(t)$.

We state the convergence results for algorithm (26) - (28). The main proofs are adapted from the proofs in [17]. More details about the proofs are given in Appendix VIII.

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 (26)-(28) with $x(0) \in \mathcal{X}_0$ is optimal.

The constant δ_1 can be chosen as in Theorem 3. The proof is given in Appendix VIII.

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 wouldn't have been deployed or activated in the first place. Alternatively, it may be easy to spot those links that won't 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 (21).). 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 (22).). We can then apply the result in [17] 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 2: 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 VIII. ■

Define

$$F = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}. \tag{29}$$

Theorem 3 (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 (26)-(28) converges to an element of $\mathcal{X}^*$ and the convergence rate is linear (i.e., geometric) in the sense that for all $t \geq t_0$,

$$f(x(t+1)) - f^* \leq (1 - D_5 \delta)(f(x(t)) - f^*). \tag{30}$$

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 2. $\hat{\sigma}_2 \leq \hat{L}_2$ 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 VIII. ■

Theorem 4 (Global Convergence - Stronger Version): Suppose $q > 2$ and $A1$ holds. For $0 < \delta < \delta_1$, the sequence $\{x(t)\}$ generated by the synchronous gradient projection algorithm (26)-(28) 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 3, 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 (7) and keep the same constraints. With this modification, $\hat{f}_e^1(y_e) = (\frac{y_e}{u_e} + \kappa)^q$ and $(\hat{f}_e^1)''(y_e) = q(q-1)(\frac{y_e}{u_e} + \kappa)^{q-2}$, 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 [17] and conclude *global* geometric convergence.

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

IV. Diagonally Scaled Algorithm

Geometric convergence in theory may still be slow for practical problems. As Section V will show, the gradient projection algorithm in (26) - (28) 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.

A. The Ill-conditioned Problem

When the power, q, is large, the problem (15)-(16) 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. 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

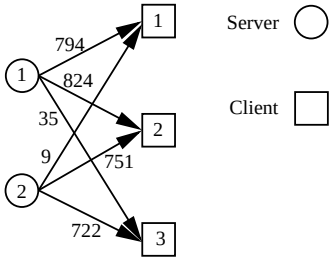


Fig. 4. A network that leads to an ill-conditioned problem; $q = 10$.

a large condition number of the Hessian of f. (See page 71 of [6] 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 (13) and (14), the step size upper bound δ_1 is about $5.6e+06$ (See Theorem 3.). 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 (26)-(28). 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 (27), 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 $x(t)$.

B. 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 [6].). Such scaling of the variables can also be interpreted as using different step sizes for different (unscaled) variables. The scaled algorithm is as follows.

$$x(t+1) = \alpha(t)\bar{x}(t) + (1-\alpha(t))x(t), \tag{31}$$

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

$$\bar{x}_{ij}(t) = [x_{ij}(t) - \delta \cdot d_{ij}^{-1}(t) \cdot (\frac{\partial f(x(t))}{\partial x_{ij}(t)} - \frac{\partial f(x(t))}{\partial x_{s_j(x(t)),j}(t)})]_+ \tag{32}$$

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

$$\bar{x}_{s_j(x(t)),j}(t) = Q_j - \sum_{i \in S, i \neq s_j(x(t))} \bar{x}_{ij}(t). \tag{33}$$

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 (17), (18), and (25), and $\alpha(t)$ is as described for the non-scaled gradient projection algorithm in Section III-B3. 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 (21) and (22), 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 (21) (22). Hence, the scaled algorithm is completely decentralized.

C. 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 [18] [17].

In the asynchronous algorithm, we replace (32) with

$$\bar{x}_{ij}(t) = [x_{ij}(t) - \delta \cdot \hat{d}_{ij}^{-1}(t) \cdot (\lambda_{ij}(t) - \lambda_{s_j(x(t)),j}(t))]_+, \tag{34}$$

and set $\bar{x}_{s_j(x(t)),j}(t)$ according to (33), 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

$$\lambda_e(t) = \sum_{\tau=t-B_2}^{t} h_e(t,\tau)(\hat{f}_e^1)'(y_e(\tau)), \tag{35}$$

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

$$\hat{d}_e(t) = \sum_{\tau=t-B_2}^{t} h_e(t,\tau)(\hat{f}_e^1)''(y_e(\tau)). \tag{36}$$

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

$$\lambda_i(t) = \sum_{\tau=t-B_2}^{t} h_i(t,\tau)(\hat{f}_i^2)'(z_i(\tau)), \tag{37}$$

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

$$\hat{d}_i(t) = \sum_{\tau=t-B_2}^{t} h_i(t,\tau)(\hat{f}_i^2)''(z_i(\tau)), \tag{38}$$

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 (19) - (22).

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

$$\lambda_{ij}(t) = \sum_{e \in P_{ij}} \lambda_e(t) + \lambda_i(t), \tag{39}$$

and

$$\hat{d}_{ij}(t) = \hat{d}_i(t) + \hat{d}_{S_j(x(t))}(t) + \sum_{e \in \hat{P}_{ij}(t)} \hat{d}_e(t), \tag{40}$$

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 (35)-(38). 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 (34) and (33). 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 (31). 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.

Theorem 5: (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 (15)-(16), the conditions of Theorem 5 are all met. Hence, we have the following conclusion.

Corollary 6: For problem (15)-(16), $f(x(t))$ 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.

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 (35) and (36), 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 (37) and (38), 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 (39), (40), respectively. Then, it computes $\bar{x}_{ij}(t)$ for all $i \in S$ according to (34) and (33). Client j chooses new rates $x_{ij}(t+1)$ for all $i \in S$ according to (31) and communicates the changed flow rates to the servers.

V. Experimental Results

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 paper 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.

A. 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 $i \in S$.

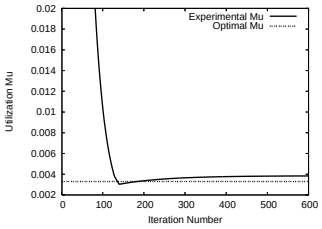
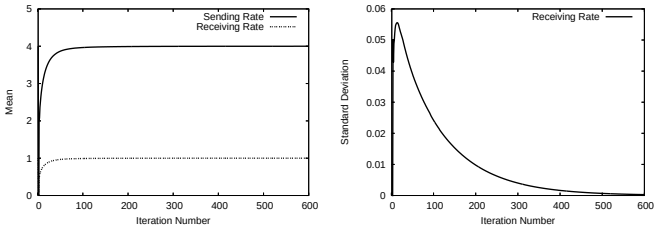
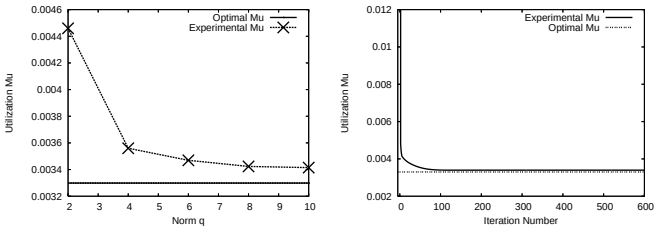


Fig. 5. Convergence of the subgradient algorithm



(a) (b)

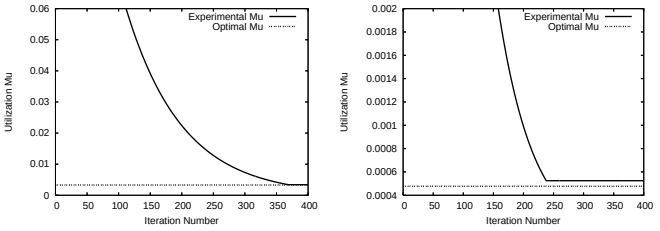
Fig. 6. Flow rate convergence of the subgradient algorithm: $K_i = 6.0, Q_j = 1.0$.



(a) (b)

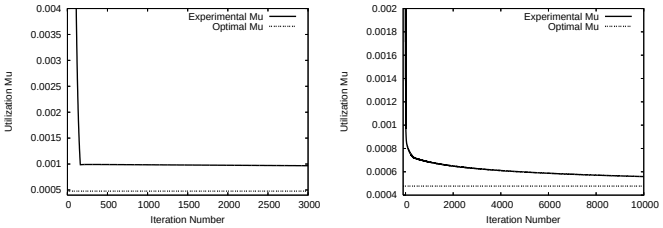
Fig. 7. Convergence of gradient projection algorithm: (a) Optimal utilization with different q; (b) Convergence of the algorithm. $q = 10$.

In the subgradient algorithm, the step size δ is chosen manually. Fig. 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. 6 (b) shows), μ converges to the optimal value. Fig. 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,



(a) (b)

Fig. 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. $q = 10$.

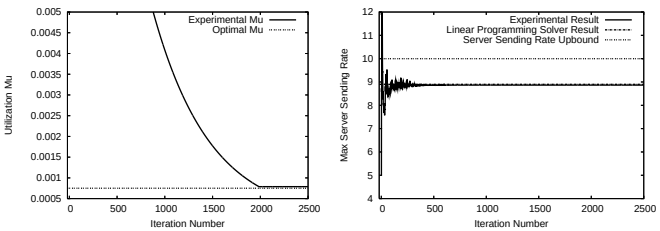


(a) (b)

Fig. 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. $q = 10$.

the algorithm is devoted to balancing the link utilization. Fig. 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. 7 (a), as q grows, the optimal value μ^* for the approximate problem (15)-(16) 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. 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.



(a) (b)

Fig. 10. The 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. $K_i = 10$.

B. Scaled vs. Unscaled Algorithm

The diagonally scaled gradient projection method is surprisingly effective for our problem. With the same network of 50 nodes, Fig. 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. 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. 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. 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. 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 won't 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. 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.

VI. Other Related Work

References to convex network optimization have been given throughout the paper. 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 [19], GridMedia [20], PRO [21], and PRM [22]. Examples of bulk data distribution systems include BitTorrent [2], FastReplica [23], SplitStream [24], Bullet′ [25], ChunkCast [26], and CoBlitz [27]. 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 paper 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 [24] and FastReplica [23], 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 [2], Bullet′ [25], ChunkCast [26] and CoBlitz [27] 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 [28]. More relevant to this paper, BitTorrent, Bullet′ and Slurpie [29] 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 [30].

The next two are content distribution systems that pay attention to network performance. Julia [31] 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 [32], 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 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 [33], 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, [34] presents a dynamic selection scheme based on instantaneous measurement of the RTT and available bandwidth. Similar approaches are also reported in [35]–[37].

VII. Conclusion

In this paper, 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.

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

There exist two classes of solutions to the optimization problem, the subgradient algorithm and a special gradient projection algorithm. The paper 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 VI. 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 paper, 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 paper 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.

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

VIII. 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 (26) - (28) 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 [17]. One key difference is that, in [17], 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, [17] 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 III-C, we can prove Lemma 2. 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 [17], this will lead to the conclusion of a local geometric convergence rate.

Proof: (of Lemma 2) 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

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

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

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

When $q = 2$, $\nabla^2 \hat{f}^1(y) = \text{diag}[\frac{q(q-1)}{u_1^q}, \frac{q(q-1)}{u_2^q}, \cdots, \frac{q(q-1)}{u_{|E|}^q}]$, for any $y = G_1 x$, where $x \in \mathcal{X}$. Let $\hat{\sigma}_1 = \min_{e \in E} \frac{q(q-1)}{u_e^q}$. We again have for all $x \in \mathcal{X}$,

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

■

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 (15) can be written as

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

Corollary 7: 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 2. $\hat{\sigma}_2$ 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 8: 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 7. $\hat{L}$ 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 [17]. [17] 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 7, we are able to have the above inequality in a neighborhood x^*. The rest of the proof is nearly identical to that in [17]. ■

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 III-C on how to do this.), Theorem 8 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 [17] involves substitution of appropriate constants.

Lemma 9: For $0 < \delta \leq \delta_1$, we have for all t that $x(t) \in \mathcal{X}_0$ and

$$f(x(t+1)) - f(x(t)) \leq -(\frac{\underline{a}}{\delta|S|} - \frac{L}{2})||x(t) - \bar{x}(t)||^2. \quad (41)$$

Lemma 10: For $0 < \delta \leq \delta_1$, we have for all t that

$$\begin{aligned} & f(\bar{x}(t)) - f^* \\ \leq \; & \tfrac{5L+1}{2}\phi(x(t))^2 + (\tfrac{3L}{2} + \tfrac{1}{\delta} + \tfrac{1}{2\delta^2})||x(t) - \bar{x}(t)||^2. \end{aligned} \quad (42)$$

Lemma 11: For $0 < \delta \leq \delta_1$, we have for all t that

$$\begin{aligned} & f(x(t+1)) - f^* \\ \leq \; & \underline{a}[f(\bar{x}(t)) - f^*] + (1 - \underline{a})[f(x(t)) - f^*] \\ & + L(2 - \tfrac{\underline{a}}{2})||x(t) - \bar{x}(t)||^2. \end{aligned} \quad (43)$$

Upon combining the preceding three lemmas and Theorem 8, Theorem 3 can be shown similarly as in [17].

A. Global Convergence without Condition $A1$

We next prove Theorem 1.

Proof: From Lemma 9, 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 (26)-(28) 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

$$\tilde{x}_{ij}^* > 0 \text{ only if } [\frac{\partial f(\tilde{x})}{\partial x_{kj}} \geq \frac{\partial f(\tilde{x})}{\partial x_{ij}}, \forall k \in S]. \quad (44)$$

which is exactly the optimality condition in (24). (Proposition 2.3.2 and Example 2.1.2 in [6]). ■

REFERENCES

[1] 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.

[2] BitTorrent Website, http://www.bittorrent.com/.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proceedings of ACM SIGMETRICS 2004*, June 2004.

[23] L. Cherkasova and J. Lee, "FastReplica: Efficient large file distribution within content delivery networks," in *Proceedings of the 4th USITS*, Seattle, WA, March 2003.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating systems principles*, 2003, pp. 282–297.

[29] R. Sherwood, R. Braud, and B. Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," in *Proceedings of IEEE Infocom*, Hong Kong, March 2004.

[30] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Sigcomm'04*, 2004.

[31] 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.

[32] P. A. Padmavathi Mundur, "Optimal server allocations for streaming multimedia applications on the Internet," *Computer Networks*, vol. 50, pp. 3608–3621, 2006.

[33] 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.

[34] 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.

[35] 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.

[36] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of IEEE Infocom*, Anchorage, AK 2001.

[37] 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.

Xiaoying Zheng is a PhD candidate in the Department of Computer and Information Science and Engineering at the University of Florida. She 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 include applications of optimization theory in networks, peer-to-peer overlay networks, content distribution and congestion control.

Ye Xia is an assistant professor at the Computer and Information Science and Engineering department at the University of Florida, starting in August 2003. He has a PhD degree from the University of California, Berkeley, in 2003, an MS degree in 1995 from Columbia University, and a BA degree in 1993 from Harvard University, all in Electrical Engineering. Between June 1994 and August 1996, he was a member of the technical staff at Bell Laboratories, Lucent Technologies in New Jersey. His research interests are in computer networking area, including performance evaluation of network protocols and algorithms, congestion control, resource allocation, and load balancing on peer-to-peer networks. He is also interested in probability theory, stochastic processes and queueing theory.