# On-Line End-to-End Congestion Control

Naveen Garg
Indian Institute of Technology
New Delhi, India
naveen@cse.iitd.ernet.in

Neal E. Young
Akamai Technologies
Cambridge, MA, USA
neal@young.name

## Abstract

*Congestion control in the current Internet is accomplished mainly by TCP/IP. To understand the macroscopic network behavior that results from TCP/IP and similar end-to-end protocols, one main analytic technique is to show that the the protocol maximizes some global objective function of the network traffic.*

*Here we analyze a particular end-to-end, MIMD (multiplicative-increase, multiplicative-decrease) protocol. We show that if all users of the network use the protocol, and all connections last for at least logarithmically many rounds, then the total weighted throughput (value of all packets received) is near the maximum possible. Our analysis includes round-trip-times, and (in contrast to most previous analyses) gives explicit convergence rates, allows connections to start and stop, and allows capacities to change.*

## 1. Congestion control and optimization

Congestion control in the current Internet is accomplished mainly by TCP/IP — 90% of Internet traffic is TCP-based [41]. Meanwhile the design and analysis of TCP and other end-to-end congestion-control protocols are only partially understood and are becoming the subject of increasing attention [25, 28]. One main analytic technique is to interpret the protocol as solving some underlying combinatorial optimization problem on the network — to show that the protocol causes the traffic distribution, over time, to optimize some global objective function [41, 29, 26, 2, 40, 17, 30, 52, 9, 46, 44, 54].

For example, a continuous analogue of TCP-Reno (under various assumptions about the network) maximizes $\sum_i \tau_i^{-1} \arctan(\tau_i x_i)$, where $\tau_i$ is the (constant) round-trip time of packets sent by the $i$th user and the variable $x_i$ is that user's transmission rate [41]. (Each term in the sum is a smoothed threshold function.) Similarly, a continuous analogue of TCP-Vegas maximizes $\sum_i \alpha_i d_i \log x_i$ where $d_i$ is the round-trip propagation delay of packets sent by the $i$th user and $\alpha_i$ is a protocol parameter. Typically these results concern a continuous analogue of the protocol. They analyze a system of differential equations where time is continuous and each rate $x_i$ is a continuous function of time. They show (e.g. using a Lyapunov function [22, 50]) that, as time tends to infinity, the vector $x$ tends to an equilibrium point that maximizes the objective function in question.

This approach is very general. It has been used to design and analyze protocols other than TCP, including protocols that require the network routers to explicitly transmit congestion information to the users by means other than packet loss and latency. (Typically the congestion signals are dual variables — Lagrange multipliers, or "shadow prices" — with interesting economic interpretations.) For a survey of results of this kind, see [41, 29]. A few specific technical papers include [26, 2, 40, 17, 30, 52, 9].

Our interest in this paper is in protocols that are both on-line and *end-to-end*: the number of packets sent on a path $p$ at time $t$ is determined solely by the number of packets sent and received on $p$ in previous rounds. (The protocol has no a-priori knowledge of the network or how the paths relate to it, and learns about the network only through packet loss.) Such protocols are implementable in the current Internet, without modifications to routers. The protocol we analyze in this paper, which we call the Linear MIMD Protocol, is an example (see Fig. 1). It can be implemented by modifying only the TCP server.

Generally, existing works (that formally analyze end-to-end protocols implementable in the current Internet) assume that all connections start at time 0 and continue indefinitely in a static network. They show that the objective function is optimized in the limit as time tends to infinity. (See Low's survey and Kelly's survey [29].) The only exceptions that we are aware of are for the special case of a single-bottleneck network [18, 9]. Thus, we do not yet have a complete theoretical understanding of speed of convergence (noted as important by Low in his survey [41]) or of the effects of dynamic connections and changing network conditions. (These issues have been studied empirically, e.g.

**Inputs for connection** $p$**:** starting rate $f_0(p) > 0$, active time interval $T_p = \{s_p, s_p + 1, \ldots, e_p\}$.
**Parameters:** $\alpha_p > 0$**,** $\beta_p \in (0, 1)$

At times $t = s_p, s_p + 1, \ldots, s_p + \tau_p$ take $\mathsf{sent}(p, t) := f_0(p)$.
At times $t = s_p + 1 + \tau_p, \ldots, e_p$, take

$$\mathsf{sent}(p,t) \;:=\; \mathsf{sent}(p, t-1-\tau_p) \times \big[1 + \alpha_p - \beta_p \mathsf{lsr}(p, t-1)\big].$$

**Figure 1. The Linear MIMD Protocol. In round $t$ on path $p$, $\mathsf{lsr}(p,t)$ is the observed packet loss fraction, $\tau_p$ is the round-trip time. For the main result, $\beta_p \in (0, \epsilon]$ and $\alpha_p = \beta_p \epsilon\, \mathsf{val}(p)$.**

[10].)

Here we use a relatively dynamic model: time is discrete, connections start and stop, network capacities vary with time. The objective function we study is *total weighted throughput* (the total value of all packets delivered).

Our main result is that, for some small $T$, *as long as each connection lasts at least $T$ rounds*, the protocol in Fig. 1 achieves a total weighted throughput of at least $(1-\epsilon)\mathsf{opt}$. Here $\mathsf{opt}$ is the maximum possible weighted throughput of any solution that respects capacity constraints and assigns a *fixed* rate $f(p)$ to each path $p$ while the path is active. $T$ is proportional to a logarithmic term over $\epsilon^3$. To sidestep the question of how the protocol finds a reasonable *starting* rate for each path, we analyze the speed of convergence given arbitrary (feasible) initial rates.

Most existing works assume instantaneous feedback about congestion. In practice feedback is delayed due to round-trip times, but delayed feedback is harder to analyze formally. Some recent works such as [46, 44, 54] study the effect of delayed feedback on convergence, and even suggest that some variants of TCP may become unstable as network capacity becomes large [42]. Here we do model delayed feedback, although we assume the delay on each path $p$ is a *fixed* constant $\tau_p$. We show that the convergence rate of the protocol grows linearly with delay.

Existing works model packet loss in the network in various ways. (Some also model variable latency due to queuing, which we do not.) Roughly, we assume that a network resource (switch, router, etc.) discards packets only if congested, and then in an approximately *fair* manner — so that no path incurs significantly disproportional loss *over time* (see Condition (2) later in the paper). We believe the packet-discard model is realistic in practice. (See the final section.)

Here is a formal statement of our result. Let $\tau_p$ be the round-trip delay on path $p$. Note that $\alpha_p$ and $\beta_p$ are parameters of the protocol and $f_0(p)$ is the initial sending rate on path $p$.

**Theorem 5** *Assume each* $\mathsf{val}(p) \le 1$. *Fix* $\epsilon > 0$. *Assume $\epsilon$-fair loss on each path. Let $U_p$ be the maximum amount received in any round on path $p$.*

*The weighted throughput achieved by the Linear MIMD Protocol with $\beta_p = O(\epsilon)$ and $\alpha_p = \epsilon\,\beta_p\,\mathsf{val}(p)$ is* $(1 - O(\epsilon))\mathsf{opt}$ *provided the duration $|T_p|$ of each connection is at least*

$$\Omega\left(\max_p \frac{(1+\tau_p)\ln(U_p/f_0(p))}{\epsilon^2 \beta_p \mathsf{val}(p)}\right).$$

In today's Internet, a generous upper bound for the ratio $U_p/f_0(p)$ is around $10^4$ — the ratio between 100 Mbps (the most a typical server can transmit) and 10Kbps (about one packet per second).

Protocols that maximize weighted throughput within a $1-\epsilon$ factor are necessarily *unfair* (they may allocate little or no bandwidth to some connections). Also, the protocol that we study is not innately *tcp-friendly*, although, with a proper setting of the parameters, the loss rates the protocol induces in the network can be made close to the "background" level of loss in the Internet (typically 2-4% in well-capacitated networks). For a discussion of other limitations, and future directions, see the final section.

The protocol can be tuned to network conditions. For example, in a network with low packet-loss rates, a particularly simple special case of the protocol — "for each packet received, send $1+\epsilon$ packets" — converges faster by a factor of $1/\epsilon$. The analysis also generalizes to the case when each path $p$ can use a resource $r$ to some extent $M_{pr} \in [0,1]$. (These results are omitted from the proceedings version.)

**Application: multi-path bandwidth testing.** Suppose a web site has multiple servers hosted in an Internet data center. The web site owner has a contract with the network provider for 100 Mbps access to the Internet. The servers send traffic through multiple, possibly shared, up-links, and then through the provider's local network, toward a backbone and/or toward the provider's local users. How can the web site owner find an optimal traffic distribution, one that maximizes the traffic sent from its servers while respecting the local bandwidth constraints? How can the site owner verify that the maximum possible traffic is at least 100 Mbps (or prove to the network provider that 100 Mbps is not possible, in violation of the contract)? This is the problem that motivated this work. It involves multiple paths, in contrast to the more-well studied problem of estimating bandwidth along a single path [36, 31, 38, 37, 8].

The Linear MIMD Protocol (or any protocol that maximizes global aggregate throughput) can be used to solve this problem as follows. Select a large representative sample of paths (from the servers to destination IP's that the servers serve). Run the Linear MIMD Protocol simultaneously on the paths (giving each path equal value) until near-convergence. The resulting aggregate throughput will be a good estimate of the maximum multi-path bandwidth.

In practice, it may suffice to send packets only over appropriate *prefixes* of the paths; this requires packet-programming techniques that we don't describe here. In our experience, the time it takes for a typical test is on the order of a minute. An advantage of this approach is that it works even in the presence of "hidden" bottlenecks, such as switches, whose presence can be known to the network user only through packet loss.

Note that a protocol such as TCP generally does not maximize aggregate throughput, and so doesn't work for this application.

**Other related work.** The protocol can be viewed as a new Lagrangian-relaxation algorithm (adapted to and implemented in the network setting) for the underlying packing/covering problem. It is most similar in spirit to recent works such as [15, 55], which are in turn part of a large body of work over the last decade [12, 19, 21, 27, 39, 32, 43, 48, 51]. The focus of all of these works are on Lagrangian-relaxation algorithms with provable convergence rates (i.e., running-time bounds). Those works, and this one, are technically related to algorithms for "boosting" and "following expert advice" in learning theory (e.g. [14]).

Within theoretical computer science, works focusing on distributed optimization of related problems include [7, 47, 6, 1, 35]. See also theoretical works on routing and related problems [4, 5, 3]. Works studying related issues of resource allocation in networks in a game-theoretical spirit include [23, 49, 11, 16, 34].

For a critique of the pricing-based research agenda for congestion control (which is closely related to the optimization paradigm), see [53].

## 2. On-line end-to-end network model

We model a network as a set of *resources* (e.g., cables, routers, switches). Each resource $r$ has a capacity $\mathsf{cap}(r,t) \geq 0$ that can vary over time. A *connection* in the network is identified with a path $p$, and has a value $\mathsf{val}(p) \in [0,1]$ and an *active time interval* $T_p$ (a finite contiguous subsequence of the times $\{0,1,2,\ldots\}$). The path $p$ represents the fixed route (determined by the routers) in the network from the source to the destination. We identify each path with the ordered sequence of resources that it uses, and we write $p \sim r$ or $r \sim p$ if path $p$ uses resource $r$.

To model feedback delay (round-trip times), we assume static latencies along the paths: $\tau_{pr} \in \{0,1,\ldots\}$ denotes the number of time intervals it takes a packet to reach its destination after going through resource $r$, while $\tau_p \in \{0,1,\ldots\}$ denotes the number of time intervals for a packet to travel the entire length of $p$. (We assume $\tau_p \geq \max_{r\sim p} \tau_{pr}$.) We use the following notations to count the number of the protocol's packets of various kinds during time $t$:

| | | |
|---|---|---|
| $\mathsf{sent}(p,t)$ | – | packets injected at path $p$'s source, |
| $\mathsf{rcvd}(p,t)$ | – | packets received at $p$'s destination, |
| $\mathsf{lost}(p,t)$ | – | $\mathsf{sent}(p,t-\tau_p) - \mathsf{rcvd}(p,t)$, |
| $\mathsf{into}(r,t)$ | – | packets entering resource $r$, |
| $\mathsf{lost}(r,t)$ | – | packets discarded at resource $r$. |

We assume for simplicity that all packets are the same size, and (adopting the standard "fluid" model) we allow the number of packets sent, received, lost, etc. to take on arbitrary real values (rather than integer values). We assume that the source of each path, by time $t$, knows $\mathsf{rcvd}(p,t-1)$ (by some mechanism such as TCP acknowledgments).

We assume that packet loss occurs at a resource $r$ at time $t$ only if the number of packets entering the resource at time $t$ exceeds the capacity $\mathsf{cap}(r,t)$. If this occurs, then the loss is adversarial (arbitrary) subject to a *fair loss* condition, which says that packet loss is not unduly biased against any particular user. To explain, note that if packet loss were perfectly distributed at each iteration, then for each path $p$ and time $t$ it would be the case that

$$\mathsf{rcvd}(p,t) \;=\; \mathsf{sent}(p,t-\tau_p)\prod_{r\sim p}\Big[1-\frac{\mathsf{lost}(r,t-\tau_{pr})}{\mathsf{into}(r,t-\tau_{pr})}\Big]. \tag{1}$$

The above would imply

$$\begin{aligned}\frac{\mathsf{lost}(p,t)}{\mathsf{sent}(p,t-\tau_p)} &= 1-\frac{\mathsf{rcvd}(p,t)}{\mathsf{sent}(p,t-\tau_p)}\\ &= 1-\prod_{r\sim p}\Big[1-\frac{\mathsf{lost}(r,t-\tau_{pr})}{\mathsf{into}(r,t-\tau_{pr})}\Big]\\ &\leq \sum_{r\sim p}\frac{\mathsf{lost}(r,t-\tau_{pr})}{\mathsf{into}(r,t-\tau_{pr})}.\end{aligned}$$

The expression on the left-hand side is the fraction of packets sent on $p$ at time $t-\tau_p$ that are lost. One can interpret the fraction on the right-hand side as the probability that any particular packet is lost at resource $r$ at the time the packets in question enter it. Under this interpretation, the equation above will hold if the fraction of packets lost on $p$ is at most the expected number. The fair loss condition ($\epsilon$-*fair loss*) is that this holds *approximately over time*:

$$\begin{aligned}(\forall p) \quad & \sum_{t \in T_p + \tau_p} \frac{\mathsf{lost}(p,t)}{\mathsf{sent}(p, t-\tau_p)} \\ \leq \quad & (1+\epsilon) \sum_{t \in T_p+\tau_p} \sum_{r \sim p} \frac{\mathsf{lost}(r, t-\tau_{pr})}{\mathsf{into}(r, t-\tau_{pr})}. \qquad (2)\end{aligned}$$

We are interested here in protocols that are both *online* and *end-to-end*: that is, they determine the packets sent on a path $p$ at time $t$ as a function of the values $\{\mathsf{sent}(p,s), \mathsf{rcvd}(p,s) : s < t\}$ (and possibly other path-specific information such as $\mathsf{val}(p)$). The protocols have no a-priori knowledge of the network.

The objective is to maximize the total value of the received packets $\sum_p \sum_{t \in T_p+\tau_p} \mathsf{val}(p)\mathsf{rcvd}(p,t)$. The *competitive ratio* (a.k.a. performance ratio) of the protocol is the worst-case ratio (over all inputs) of this quantity to opt — the maximum that could by assigning a fixed sending rate $f(p)$ to each path $p$ while its active (without exceeding capacity constraints).

## 3. Analysis of the Linear MIMD Protocol

**Definitions 1** *Let* $T'_p = T_p + \tau_p = \{s_p+\tau_p, \ldots, e_p+\tau_p\}$ *and lost-to-sent ratio* $\mathsf{lsr}(p,t) = \mathsf{lost}(p,t)/\mathsf{sent}(p,t-\tau_p)$.

First, we lower-bound the throughput in terms of the packet loss:

**Lemma 1** *The Linear MIMD Protocol satisfies, for each path* $p$,

$$\begin{aligned}\sum_{t \in T'_p} \mathsf{rcvd}(p,t) \quad \geq \quad & (\beta_p/\alpha_p - 1) \sum_{t \in T'_p} \mathsf{lost}(p,t) \\ & - \alpha_p^{-1}(\tau_p+1) f_0(p).\end{aligned}$$

**Proof:** Let $S = \sum_{t \in T'_p} \mathsf{sent}(p, t-\tau_p)$,
$R = \sum_{t \in T'_p} \mathsf{rcvd}(p,t)$, and $L = \sum_{t \in T'_p} \mathsf{lost}(p,t)$.

Expanding the product on the right-hand side in the definition of the protocol, for $t \in T'_p$,

$$\mathsf{sent}(p, t+1) \leq (1+\alpha_p)\mathsf{sent}(p, t-\tau_p) - \beta_p \mathsf{lost}(p,t).$$

Summing over $t \in T'_p$ gives

$$\sum_{t \in T'_p} \mathsf{sent}(p, t+1) \ \leq \ (1+\alpha_p)\, S - \beta_p L.$$

Subtracting $S$ from both sides and substituting $R + L$ for $S$ gives

$$\alpha_p R \geq (\beta_p - \alpha_p) L - (1+\tau_p) f_0(p)$$

$\square$

Next, we lower-bound the packet loss in terms of opt, but with a condition.

**Definitions 2** *Let* $\mathsf{rcvd}^*(p)$ *denote the number of packets the optimal algorithm would send on path* $p$ *when it is active. Define* $\mathsf{into}^*(r,t)$ *to be the number of packets the optimal algorithm would send through* $r$ *at time* $t$*. Let* $\mathsf{opt} = \sum_p \mathsf{val}(p)\mathsf{rcvd}^*(p)|T_p|$ *denote the optimum weighted throughput.*

**Lemma 2** *Assume that only congested resources discard packets. If, for some* $c$ *and each* $p$, $\sum_{t \in T'_p} \sum_{r \sim p} \frac{\mathsf{lost}(r, t-\tau_{pr})}{\mathsf{into}(r, t-\tau_{pr})} \geq c\,|T_p|\mathsf{val}(p)$, *then the number of packets lost,* $\sum_t \sum_r \mathsf{lost}(r,t)$, *is at least* $c\,\mathsf{opt}$.

**Proof:** We show that the loss rates at the resources implicitly define a solution to the dual linear program of the problem, and we use the dual solution to bound opt.

For each of the protocol's packets lost at a resource $r$ at time $t$ where $\mathsf{into}^*(r,t) > 0$, allocate a charge of one credit uniformly across each of the $\mathsf{into}^*(r,t)$ packets sent through $r$ by opt at time $t$. The assumption that packets are discarded only by congested resources means that $\mathsf{lost}(r,t) > 0$ only if $\mathsf{into}(r,t) \geq \mathsf{cap}(r,t) \geq \mathsf{into}^*(r,t)$, so the charge to each packet that opt sends through $r$ at time $t$ is

$$\frac{\mathsf{lost}(r,t)}{\mathsf{into}^*(r,t)} \geq \frac{\mathsf{lost}(r,t)}{\mathsf{into}(r,t)}.$$

The number of lost packets is at least the total charge to opt's packets, which by the above (and the supposition in the lemma about each path $p$) is at least

$$\begin{aligned}& \sum_p \mathsf{rcvd}^*(p) \sum_{r \sim p} \sum_{t \in T'_p} \frac{\mathsf{lost}(r, t-\tau_{pr})}{\mathsf{into}(r, t-\tau_{pr})} \\ \geq \quad & \sum_p \mathsf{rcvd}^*(p)\, c\, |T_p| \mathsf{val}(p) \\ = \quad & c\,\mathsf{opt}.\end{aligned}$$

$\square$

**Lemma 3** *Let* $U_p$ *be the most received in any round on path* $p$*. The Linear MIMD Protocol satisfies, for each path* $p$,

$$\begin{aligned}\sum_{t \in T'_p} \mathsf{lsr}(p,t) \quad \geq \quad & \frac{\alpha_p(1-\beta_p)}{\beta_p(1+\alpha_p)}(|T_p| - 1 - \tau_p) \\ & - \frac{1-\beta_p}{\beta_p}(1+\tau_p) \ln \frac{\beta_p U_p}{(\beta_p - \alpha_p) f_0(p)}.\end{aligned}$$

**Proof:** We use that (for $0 \leq a \leq \alpha$ and $0 \leq b \leq \beta < 1$),

$$1 + a - b \ \geq \ (1+a)(1-b) \ \geq \ \exp[a/(1+\alpha) - b/(1-\beta)].$$

Fix a path $p$. Recall $\mathsf{lsr}(p,t) = \mathsf{lost}(p,t)/\mathsf{sent}(p,t-\tau_p)$. By the definition of the protocol and the stated inequality, for $t = s_p + 1 + \tau_p, \ldots, e_p$,

$$\begin{aligned}\frac{\mathsf{sent}(p,t)}{\mathsf{sent}(p,t-1-\tau_p)} &= 1 + \alpha_p - \beta_p \mathsf{lsr}(p,t-1) \\ &\geq \exp\left[\frac{\alpha_p}{1+\alpha_p} - \frac{\beta_p \mathsf{lsr}(p,t-1)}{1-\beta_p}\right].\end{aligned}$$

Taking logs and summing over $t$,

$$\begin{aligned}&\sum_{t=s_p+1+\tau_p}^{e_p} \ln \frac{\mathsf{sent}(p,t)}{\mathsf{sent}(p,t-1-\tau_p)} \\ &\geq \sum_{t=s_p+1+\tau_p}^{e_p} \frac{\alpha_p}{1+\alpha_p} - \mathsf{lsr}(p,t)\frac{\beta_p}{1-\beta_p}.\end{aligned}$$

The sum on the left-hand side telescopes. Since at most $U_p$ is received on $p$ in any round, a proof by induction shows that at most $U_p\beta_p/(\beta_p - \alpha_p)$ is ever sent on $p$ in any round. Thus,

$$\begin{aligned}&(1+\tau_p)\ln\frac{U_p\beta_p}{(\beta_p-\alpha_p)f_0(p)} \\ &\geq (|T_p| - \tau_p - 1)\frac{\alpha_p}{1+\alpha_p} - \sum_{t=s_p+1+\tau_p}^{e_p} \mathsf{lsr}(p,t)\frac{\beta_p}{1-\beta_p}.\end{aligned}$$

The lemma follows. □

Next we combine the three lemmas.

**Lemma 4** *Fix $\epsilon > 0$. Assume $\epsilon$-fair loss. Let $U_p$ be the maximum amount received in any round on path $p$. Define $b = \min_p(\beta_p/\alpha_p - 1)\mathsf{val}(p)$. Define*

$$\begin{aligned}c &= \min_p \frac{\alpha_p(1-\beta_p)}{\beta_p(1+\alpha_p)\mathsf{val}(p)}(1 - \frac{1+\tau_p}{|T_p|}) \\ &\quad - \frac{1-\beta_p}{\beta_p \mathsf{val}(p)}\frac{1+\tau_p}{|T_p|}\ln\frac{\beta_p U_p}{(\beta_p - \alpha_p)f_0(p)}.\end{aligned}$$

*Then the total weighted throughput achieved by the protocol by round $T$ is at least*

$$b\,c\,\mathsf{opt}/(1+\epsilon) - \sum_p \frac{(1+\tau_p)f_0(p)\mathsf{val}(p)}{\alpha_p}$$

**Proof:** Applying Lemma 1 and the choice of $b$, the total weighted throughput is

$$\begin{aligned}&\sum_p \sum_{t\in T'_p} \mathsf{val}(p)\mathsf{rcvd}(p,t) \\ &\geq b\sum_p\left(\sum_{t\in T'_p}\mathsf{lost}(p,t) - \frac{(1+\tau_p)f_0(p)\mathsf{val}(p)}{\alpha_p}\right) \quad (3)\end{aligned}$$

Applying Lemma 3 and the choice of $c$,

$$\sum_{t\in T'_p} \mathsf{lsr}(p,t) \geq c|T_p|\mathsf{val}(p). \qquad (4)$$

$\epsilon$-fair loss over period $T'_p$ means that, for each $p$,

$$\sum_{t\in T'_p}\mathsf{lsr}(p,t) \leq (1+\epsilon)\sum_{t\in T'_p}\sum_{r\sim p}\frac{\mathsf{lost}(r,t-\tau_{pr})}{\mathsf{into}(r,t-\tau_{pr})}. \qquad (5)$$

Combining (4) and (5),

$$\sum_{t\in T'_p}\sum_{r\sim p}\frac{\mathsf{lost}(r,t-\tau_p)}{\mathsf{into}(r,t-\tau_p)} \geq c|T_p|\mathsf{val}(p)/(1+\epsilon).$$

This means the condition of Lemma 2 is met, so

$$\sum_t\sum_r \mathsf{lost}(r,t) \geq c\,\mathsf{opt}/(1+\epsilon)$$

Since every lost packet is eventually observed lost on a path,

$$\sum_p\sum_{t\in T'_p}\mathsf{lost}(p,t) \geq \sum_t\sum_r \mathsf{lost}(r,t).$$

Substituting the previous two inequalities into (3), the total weighted throughput is at least

$$b\,c\,\mathsf{opt}/(1+\epsilon) - \sum_p \alpha_p^{-1}(\tau_p+1)f_0(p)\mathsf{val}(p).$$

□

**Theorem 5** *Assume each $\mathsf{val}(p) \leq 1$. Fix $\epsilon > 0$. Assume $\epsilon$-fair loss on each path. Let $U_p$ be the maximum amount received in any round on path $p$.*

*The weighted throughput achieved by the Linear MIMD Protocol with $\beta_p = O(\epsilon)$ and $\alpha_p = \epsilon\,\beta_p\,\mathsf{val}(p)$ is $(1 - O(\epsilon))\mathsf{opt}$ provided the duration $|T_p|$ of each connection is at least*

$$\Omega\left(\max_p \frac{(1+\tau_p)\ln(U_p/f_0(p))}{\epsilon^2\beta_p\mathsf{val}(p)}\right).$$

**Proof:** Apply Lemma 4. With these choices for $\alpha_p$ and $\beta_p$, in that lemma, $b = \epsilon(1-O(\epsilon))$, $c = \epsilon^{-1}(1-O(\epsilon))$, and $\sum_p(1+\tau_p)f_0(p)\mathsf{val}(p)/\alpha_p = O(\epsilon\mathsf{opt})$. Thus, the lower bound that lemma gives on the weighted throughput is $[1 - O(\epsilon)]\mathsf{opt}$. □

## 4. Discussion

**Limitations of the model.** This paper assumes static packet latencies (round-trip times). In practice, a U.S.

coast-to-coast packet round-trip time is on the order of tens of milliseconds. A typical router queue might hold about 1/10 seconds worth of packets. So, the variation in round-trip time due to queuing can be larger than the round-trip time itself — our assumption of static latencies is not realistic. How will this affect the protocol in practice? Can one generalize the analysis to show that some amount of queuing doesn't hurt, or can queuing destabilize the protocol? (For one discussion of the effects of large queues in the Internet, see [45].) One workaround is to take the time per iteration in the protocol to be on the order of 1/10 second to 1 second — long enough so that, even with queuing, round-trip times are likely to be only one iteration.

Here we've assumed all packets are the same size. This hides two practical issues: typically routing has a per-packet cost, and large packets tend to be more likely to be dropped by congested routers (as routers queue packets in different queues by size).

In the current Internet, although long-lasting connections carry most of the traffic, most connections are short-lived. Do short-lived connections interfere?

The "fluid packet model" – in which we assume that arbitrary sending rates are possible, is unrealistic in the case that the sending rate on a path becomes very low. To have confidence that the analysis in this paper is realistic, each path should be sending at least one packet per iteration. (In practice, it may be useful to modify the protocol so that each path sends at least this minimal rate.) Can one modify the analysis to model discrete packets?

We do consider the "fair loss" assumption here realistic (modulo the problem of low sending rates). We note that existing theoretical analyses of end-to-end protocols typically assume fair loss at each point in time (that is, that (1) holds exactly or approximately in each round). Other models are plausible — for example, random discards [13] as in random early detection (RED) — and would allow similar assumptions.

We proved performance guarantees with respect to the *static* optimum (i.e., the sending rate on a path during its active interval is constant). How much does this restrict opt in a network where capacities change slowly?

**Interesting directions.** Prove upper and lower bounds on best-possible convergence rates of on-line, end-to-end protocols that approximately optimize various objective functions.

For example, it is straightforward to prove that any protocol requires $\Omega(\log[\log(U/f_0)/\epsilon])$ rounds to estimate the capacity of a single link of capacity $U$ within a factor of $1+\epsilon$, and $1/\epsilon$ times this many rounds to achieve an average throughput of $1-\epsilon$ times $U$. If we restrict the protocol so that it can't send more than $1+O(\epsilon)$ times what it has successfully transmitted, then these lower bounds can be improved to $\Omega(\log(U/f_0)/\epsilon)$ rounds to estimate the capacity of the link and $1/\epsilon$ times that to achieve an average throughput of $1-\epsilon$ times $U$. Strengthen these lower bounds.

Lower bounds on related algorithms have been proven in the contexts of static optimization [33] and learning theory [14]. For some lower bounds for particular protocols (e.g. a comparison of MIMD and AIMD convergence on a single-path network), see [18, 9].

An important objective function to study is *proportional fairness* (e.g. $\sum_p \log f(p)$). For upper bounds, a natural starting point would be to first develop a Lagrangian-relaxation algorithm that solves the underlying optimization problem with a provably good running time [24, 20].

Find a protocol that simultaneously maximizes aggregate throughput within a constant factor *and* maximizes proportional fairness within an additive constant.

A natural objection to the model in this paper is that, in practice, most connections are open not for a given period of time, but rather until a given number of bits have been transmitted. What is a reasonable objective function to model this? Perhaps an objective functions related to scheduling — e.g. minimize the total wait (sum of transfer times).

## Appendix

**Connection to existing algorithms.** We briefly sketch the connection between the protocols described here and existing packing and covering algorithms [15, 55]. Understanding this connection may be useful to aid further research. We designed the protocol in this paper by starting with Lagrangian-relaxation algorithms for the dual problem (a covering problem — fractional weighted set cover) of the underlying static multicommodity flow problem, and then adapting it to the current setting (e.g. with the fair loss assumption, round-trip times, etc.). The resulting algorithm is by no means a direct translation of an existing algorithm — in fact, appropriately parameterized, it runs faster than existing algorithms as they are implemented in the literature. However, some connections remain. Briefly, each round of the protocol corresponds to an iteration of the algorithm. Each path $p$'s sending rate is proportional to a path variable $x_p$ in the algorithm. Each resource $r$'s cumulative packet loss is proportional to a resource variable $\ell_r$ in the algorithm. Roughly, the algorithm ensures that (by time $T$) we get something like the familiar invariant $x_p \approx (1-\epsilon)^{\sum_{r\sim p}\ell_r}$. When a resource discards a packet, the resulting adjustments in the sending rates correspond to incrementing the corresponding path variable; that only congested resources lose packets means that resource variables get incremented only if their corresponding constraints are "sufficiently violated".

## References

[1] Y. Afek, Y. Mansour, and Z. Ostfeld. On the convergence complexity of optimistic rate based flow control algorithms. In *Symposium on Principles of Distributed Computing*, page 212, 1996.

[2] S. Athuraliya and S. H. Low. Optimization flow control with Newton-like algorithm. *Journal of Telecommunication Systems*, 2000.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *IEEE Symposium on Foundations of Computer Science*, pages 32–40, 1993.

[4] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 321–327, Jan. 1995.

[5] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competive non-preemptive call control. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 312–320, 1995.

[6] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. In *INFOCOM (3)*, pages 1350–1357, 1998.

[7] Y. Bartal, J. W. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *IEEE Symposium on Foundations of Computer Science*, pages 303–312, 1997.

[8] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evauation*, 27(8):297–318, Oct. 1996.

[9] D.-M. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[10] S. F. Deepak Bansal, Hari Balakrishnan and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *SIGCOMM 2001*, San Diego, CA, August 2001.

[11] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of muliticast transmissions. In *ACM Symposium on Theory of Computing*, pages 218–227, 2000.

[12] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, Nov. 2000.

[13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[14] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[15] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[16] A. Goel, A. Meyerson, and S. A. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. In *ACM Symposium on Theory of Computing*, pages 670–679, 2000.

[17] S. J. Golestani and S. Bhattacharyya. A class of end-to-end congestion control algorithms for the Internet. In *Proceedings of the International Conference on Network Protocols*, 1998.

[18] S. Gorinsky and H. Vin. Additive increase appears inferior. Technical Report TR2000-18, Department of Computer Sciences, University of Texas at Austin, May 2000.

[19] M. Grigoriadis and L. Khachiyan. Approximate minimum-cost multicommodity flows in $o(\epsilon^{-2}knm)$ time. *Math. Programming*, 75:477–482, 1996.

[20] M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1), 1994.

[21] M. D. Grigoriadis and L. G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Mathematics of Operations Research*, 21:321–340, 1996.

[22] H. Hochstadt. *Differential Equations, A Modern Approach*. Dover Publications, 1975.

[23] K. Jain and V. V. Vazirani. Applications of approximation algorithms to cooperative games. In *ACM Symposium on Theory of Computing*, pages 364–372, 2001.

[24] K. Jansen and H. Zhang. Approximation algorithms for general packing problems with modified logarithmic potential function. In *Proceedings 2nd IFIP International Conference on Theoretical Computer Science*, August 2002.

[25] R. Johari. Mathematical modeling and control of Internet congestion. *SIAM News*, 33, 2000.

[26] R. Johari and D. K. H. Tan. End-to-end congestion control for the Internet: Delays and stability. *IEEE/ACM Transactions on Networking*, 9:818–832, 2001.

[27] D. Karger and S. A. Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *ACM Symposium on Theory of Computing*, pages 18–25, 1995.

[28] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Optimization problems in congestion control. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[29] F. P. Kelly. Mathematical modelling of the Internet. In *Proc. of Fourth International Congress on Industrial and Applied Mathematics*, 1999.

[30] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: Shadow prices, proportional fairness, and stability. *Journal of the Operational Research Society*, 49(237-252), 1998.

[31] S. Keshav. A control-theoretic approach to flow control. *Proceedings of the Conference on Communications Architecture and Protocols*, pages 3–15, 1993.

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

[33] P. Klein and N. E. Young. On the number of iterations for Dantzig-Wolfe optimization and packing-covering approximation algorithms. In *Lecture Notes in Computer Science*, number 1610, pages 320–327, 1999. IPCO '99.

[34] J. M. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *IEEE Symposium on Foundations of Computer Science*, pages 568–578, 1999.

[35] Y. Korilis and A. Lazar. Why is flow control hard: Optimality, fairness, partial and delayed information. In *Proc. 2nd ORSA Telecommunications Conference*, March 1992.

[36] K. Lai and M. Baker. Measuring bandwidth. In *INFOCOM (1)*, pages 235–245, 1999.

[37] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *SIGCOMM*, pages 283–294, 2000.

[38] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proceedings of USENIX Symposium on Internet Technologies and Systems.*, Mar. 2001.

[39] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *J. Comput. Syst. Sci.*, 50(2):228–243, Apr. 1995.

[40] S. H. Low and D. E. Lapsley. Optimization flow control I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[41] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control: An analytical perspective. *IEEE Control Systems Magazine*, Jan. 2002.

[42] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of tcp/red and a scalable control. *IEEE Infocom*, June 2002.

[43] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *ACM Symposium on Theory of Computing*, pages 448–457, 1993.

[44] L. Massoulie. Stability of distributed congestion control with heterogeneous feedback delays. Technical Report MSR-TR-2000-111, Microsoft Corporation, November 2000.

[45] R. T. Morris. *Scalable TCP Congestion Control*. PhD thesis, Harvard University, 1999.

[46] F. Paganini, J. Doyle, and S. Low. Scalable laws for stable network congestion control. In *Proceedings of Conference on Decision and Control*, December 2001.

[47] C. H. Papadimitriou and M. Yannakakis. Linear programming without the matrix. In *Proc. 25th Annual ACM Symp. on Theory of Computing*, pages 121–129, 1993.

[48] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

[49] T. Roughgarden and E. Tardos. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.

[50] D. A. Sánchez. *Ordinary Differential Equations and Stability Theory: An Introduction*. W. H. Freeman and Company, San Francisco, 1968.

[51] F. Shahroki and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37:318–334, 1990.

[52] S. Shenker. A theoretical analysis of feedback flow control. In *ACM/SIGCOMM*, 1990.

[53] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in computer networks: Reshaping the research agenda. *Communications Policy*, 20(1), 1996.

[54] G. Vinnicombe. On the stability of end-to-end congestion control for the Internet. Technical Report CUED/F-INFENG/TR.398, University of Cambridge, December 2000.

[55] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *IEEE Symposium on Foundations of Computer Science*, 2001.