Multipoint-to-Point Session Fairness in the Internet

Pradnya Karbhari, Ellen Zegura, Mostafa Ammar
Networking and Telecommunications Group, College of Computing
Georgia Institute of Technology, Atlanta, Georgia 30332
Email: {pradnya,ewz,ammar}@cc.gatech.edu

Abstract— **In the current Internet, many applications start sessions with multiple connections to multiple servers in order to expedite the reception of data. One may argue that such aggressive behavior leads to unfair sharing of bandwidth using the current per-connection rate allocation methods. Sessions with more connections get a higher total rate than competing sessions with fewer connections. In this paper, we explore the issue of fairness of rate allocation from a session point of view. We define a *multipoint-to-point session* as a set of point-to-point connections started from multiple servers to a client in order to transfer an application-level object. We present session fairness definitions, propose algorithms to achieve these definitions, and compare the resulting allocations with the traditional connection fair algorithm. It is clear from our evaluations that the session fair algorithms proposed achieve a more fair distribution of session rates than the connection fair algorithm, by redistributing the rates claimed by sessions with more connections. We present some initial thoughts on the challenges involved in implementing the session fair algorithms proposed.**

I. Introduction

Content delivery in the current Internet has evolved beyond a simple point-to-point connection from a server to a client. Many retrievals now involve parallel point-to-point connections from multiple servers to a single client. This shift is largely an artifact of the competition among content providers who are motivated to employ aggressive methods for content delivery in an attempt to give better delay and bandwidth performance than their competitors.

Content delivery networks (CDNs), replicated servers and caches, are some of the means used by content providers for distributing content from servers in close proximity to clients. These servers typically do not cache all the data from the origin site, but only a subset of the data. Thus, a single request for a page is satisfied by setting up multiple connections to different servers.

For example, when a user requests the webpage www.cnn.com, the data is retrieved from the origin server, an advertisement server, an image server, and a couple of CDN servers. Krishnamurthy et al. [1] have observed in a January 2001 study that the average number of CDN servers contacted by a client is somewhere between 3.4 and 10.3, with the median being around 6.

Recent proposals for content delivery of streaming media [2]–[4] take advantage of path diversity between multiple servers/peers and clients. These approaches use different forms of content coding (e.g. multiple description coding) to produce complementary descriptions of content, which are then served from multiple servers/peers to the same client.

Companies like PeerGenius.com [5], CenterSpan.com [6], digitalfountain.com [7] and parallel file-download applications such as those proposed in [8] start multiple connections to multiple servers in order to expedite the reception of data, thus improving user-perceived performance.

The conclusion we draw from these example applications is that data transfer is increasingly being performed over a set of point-to-point connections from multiple servers to a single client, and we can expect this trend to continue. We refer to such a set of connections as a *multipoint-to-point session*.

The long-standing max-min fair [9] rate allocation strategy, which proposes a fair allocation for competing connections, is based on a network with point-to-point connections. It allocates rates to connections, considering each connection independently. In the current Internet, this strategy favors sessions with more connections because a session with multiple connections from multiple servers to a single client will be given a higher total rate than a session with a single connection between one of the servers and the client.

For example, if two connections belonging to a single session are bottlenecked at the same link, the max-min fair rate allocation strategy says that each connection should get an equal share at that link. Thus, the session as a whole will receive twice as much bandwidth as any other single-connection session bottlenecked at that link. One may view this as being "unfair" from the point of view of a session with fewer connections. In this work, we explore the possibility of treating sessions "equally", irrespective of the number of connections they comprise[1].

In general, the data path of a multipoint-to-point session forms a tree with data flow from the leaves to the root, as shown in Figure 1. The whole network will thus have a set of senders and a set of receivers, with each session comprising some of these senders and a single receiver[2]. Each connection in the session might have different individual bottlenecks. The data path of a session might thus have multiple bottlenecks and sharing of these bottlenecks at a session level, as opposed to the current connection-level sharing, is a challenging problem.

This work was supported by NSF grant ANI-9973115 and the Georgia Tech Broadband Institute.

[1] The appropriate use of our mechanisms is a matter of policy.

[2] For simplicity, we assume a network with only point-to-point and multipoint-to-point connections. We expect that point-to-multipoint connections can be accommodated as a set of point-to-point connections.

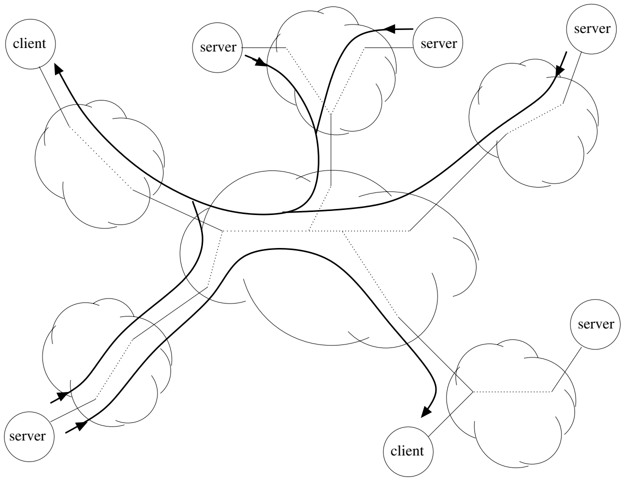


Fig. 1. Multipoint-to-point sessions in the Internet

Our goal in this paper is to explore this issue of fairness of rate allocations from a session perspective. This problem has been alluded to as an open problem by Balakrishnan et al. [10] and Padmanabhan [11]. In particular, we look at *multipoint-to-point sessions*, which are defined as *a set of point-to-point connections started from multiple servers to a client in order to transfer an application-level object*. We explore answers to the questions:

- What are "reasonable" definitions for session fairness?
- How do connection-fair allocations differ from session fair allocations?

We use Raj Jain's fairness index [12], variance of session rates, and mean, minimum and maximum session rates as quantitative metrics to compare rate allocations. We show that the fairness index for session rates improves with the session fair allocations, while maintaining or improving overall utilization, in comparison with the connection fair allocation. Also, the variance and minimum of the session rates achieved with the session fair allocations are lower than those with the connection fair allocation.

The outline of the paper is as follows. We start with related work in Section II. We elaborate on the problem statement in Section III. We then formalize the definitions of session fair allocations, and give algorithms to achieve the same in Section IV. The evaluation results comparing the session fair algorithms with each other and with the original connection fair algorithm are presented in Section V. Implementation issues are briefly outlined in Section VI, and we conclude in Section VII.

II. Related Work

Previous work, such as Webmux [13], Ensemble-TCP [14], and Integrated Congestion Manager [15], looks at the problem of fairness of resource allocation to multiple connections between the same client and server pair. In our work, we extend this problem to the case of multiple servers sending to a single client. The challenge in this case is that each connection in the session might traverse different paths and hence might have different bottleneck links. Thus, connections cannot share congestion information with each other, as proposed in the above papers. In Section VI, we present some thoughts on the extension of some of these approaches to handle the multipoint-to-point session fairness problem.

Banchs [16] has recently proposed the notion of user-fairness, in which every sender is considered as a single user in the network. This fairness scheme, referred to as *user fair queueing (UFQ)*, suggests that all connections started by a user (sender) should be considered as a single entity for rate allocation. This proposal moves away from the notion of per-connection fairness, towards session fairness from a sender point of view. This approach might put the client at a disadvantage because it might contact a server that has many other open connections, each one of them thus receiving a low rate. We compute the allocation achieved by this proposal with our session fair allocations. Our algorithms have a more fair allocation from the client's point of view, compared to the user fairness approach.

In ATM research, there has been some work on multipoint-to-point session fairness. Fahmy et al. [17] give an informal definition for virtual-circuit based (VC-based) fairness, but without a formal definition or an algorithm to achieve the same. Moh and Chen [18] present an informal definition and an algorithm for multipoint-to-point multicast flow control. These approaches benefit from the ability of ATM switches to do complex processing and focus on intra-network merge point behavior. However, in the Internet we are limited in our ability to implement functionality in routers. We want the solution to be purely an end-to-end [19] solution.

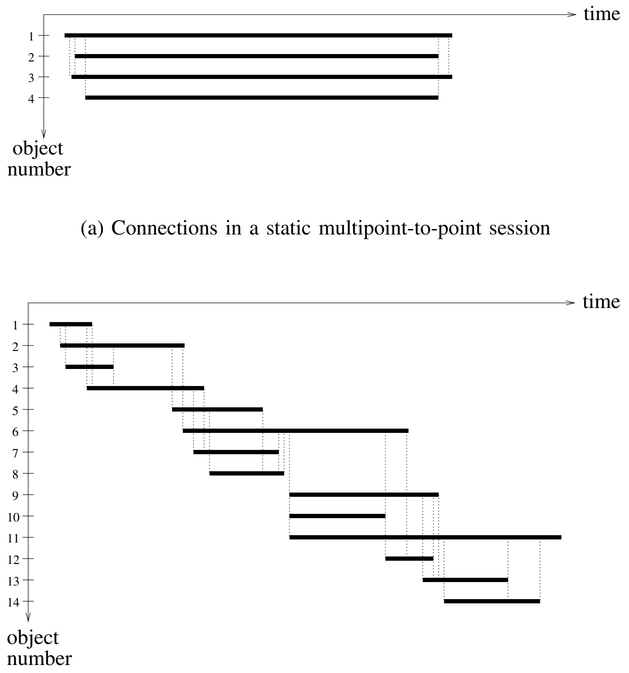
Point-to-multipoint session fairness has been discussed in the multirate multicast context in the paper by Rubenstein et al. [20]. In the multirate multicast scenario, the bandwidth used by each session on any link is the *maximum* of the receiving rate of any of the downstream receivers. In our case however, the session rate on any link is the *sum of the total rates* of the upstream senders that are sending to a particular client. Hence we cannot directly use their approach to solve the multipoint-to-point session fairness problem.

III. Discussion of Problem Statement

This section provides more detail regarding the problem of multipoint-to-point session fairness, and describes informally the proposed fairness definitions.

A. Static and Dynamic Sessions

We begin by differentiating between static and dynamic sessions. *Static sessions* are multipoint-to-point sessions comprising connections which, for the purposes of rate allocation, start and terminate at approximately the same time. On the other hand, connections in *dynamic sessions* start and terminate at different times. Thus, the composition of a dynamic session varies significantly over time. Clearly, the designation of static or dynamic must be based on the temporal granularity of resource allocation.



(a) Connections in a static multipoint-to-point session

(b) Connections in a dynamic multipoint-to-point session

Fig. 2. Static and Dynamic multipoint-to-point sessions

A typical static session is shown in Figure 2(a). The length of each horizontal line denotes the duration that the connection was active. The Multiple Description Streaming Media with Content Delivery Networks approach [2], the parallel-access approach in Rodriguez et al. [8], and retrieval by companies like PeerGenius.com [5], CenterSpan.com [6], digitalfountain.com [7], are examples of applications using static sessions.

Retrieval of webpages, on the other hand, is typically an example of a dynamic session. Figure 2(b) from Liston et al. [21] shows a typical object retrieval with a number of connections (possibly to different servers) starting and ending at different points in time.

In this work, we focus on static sessions. Static sessions are important in their own right, given the number of near-simultaneous parallel download applications that are increasingly under development. Further, the definitions and algorithms proposed here can be used as building blocks to solve the session fairness problem for dynamic sessions[3].

B. Inter-Session and Intra-Session Fairness

The notion of session fairness has two components—

- *Inter-session fairness* refers to the fairness of the total session rates with reference to each other.
- *Intra-session fairness* refers to the division of the session rate amongst the connections constituting the session.

Inter-session fairness advocates fair sharing of network resources between independent sessions on intersecting paths. To this end, we propose the notions of *normalized rate session fairness* and *per-link session fairness*, the details of which are presented in Section IV.

Normalized rate session fairness is based on the notion of *weight of a connection*, which is used to express, in some sense, the number of connections in the session. Each connection in a session is assigned a weight, such that the total weight of the connections in a session is 1. The rate allocation for each connection at every link in the network is based on the weight of the connection. The constraint on the total weight of the connections in a session ensures that the session rates are fair with reference to each other. In fact, if all connections in all sessions traverse the same bottleneck link, all sessions will get an equal session rate.

Per-link session fairness, on the other hand, looks at each link in the network and tries to ensure that each session shares the capacity of the link in a fair manner. The capacity of every link in the network is divided equally between the sessions traversing that link. The sessions divide the rate assigned to them at that link amongst the connections belonging to that session, that traverse the link[4].

Intra-session fairness options allow the client in each session to decide for itself as to how each connection in the session should be treated, in terms of rate allocation. An allocation is said to be *"source and session fair"* if each connection in the session is treated equally. That is, the rate allocation within the session is fair from the point of view of the sources in the session. If each connection in a session is given a rate allocation in proportion to the amount of data retrieved over that connection, the allocation is said to be *"data and session fair"*. This allocation has the appeal of allowing sources with more data to operate at higher speeds than sources with less data. If each connection in a session is given a rate allocation based on its path in the network (i.e. based on sharing of links between connections in the session), the allocation is said to be *"path and session fair"*. For simplicity, we describe algorithms that achieve "source and session fairness".

The normalized rate session fair algorithm is very flexible and can be easily extended to incorporate intra-session fairness definitions, based on the clients' choices, by adjusting the weights of the connections. It is also amenable to a distributed end-to-end implementation. On the other hand, it is difficult to extend the per-link session fair algorithm to incorporate the intra-session fairness definitions. This is because the algorithm has implicit weights embedded for each connection. There is no way for the client to specify (say with the weight of the connection) that it wants otherwise, in terms of rate allocation within the session, at any link.

In the next section, we discuss these definitions and algorithms in detail.

[3] Dynamic sessions may require additional or alternative strategies, particularly if the connections are highly dynamic.

[4] We conjecture that per-link session fairness is equivalent to normalized rate session fairness for some globally-coordinated assignment of weights. However, such an assignment is sufficiently complex that it makes sense to describe per-link session fairness separately.

IV. Definitions and Algorithms

In this section we formalize the definitions for different fairness criteria for rate allocation to multipoint-to-point sessions. For comparison, we define the connection max-min fairness criterion. We also present algorithms to achieve the allocations defined, and discuss properties of these allocations with respect to their session rates. The notations used are given in Table I[5].

TABLE I

Notations

l_j	jth link in the network
C_j	capacity of link l_j
c_j	residual capacity of link l_j
S_i	ith session in the network
m	number of sessions in the network
k_i	number of senders in session S_i
$s_{i,k}$	kth sender in session S_i
r_i	ith receiver in the network
$R_{i,j}$	set of receivers in S_i whose data path traverses l_j
R_j	set of receivers in all sessions whose data path traverses l_j
$a_{i,k}$	data rate of sender $s_{i,k}$
$\overline{a}_{i,k}$	data rate of sender $s_{i,k}$ using an alternate allocation
$w_{i,k}$	weight of sender $s_{i,k}$
$z_{i,k}$	normalized rate for sender $s_{i,k}$
$b_{i,k,j}$	link rate for kth connection in session S_i on l_j
$b_{i,j}$	link rate for session S_i on l_j
b_j	link rate for all sessions on l_j
a_i	data rate for session i (or session rate of S_i)
X, Y	set of unassigned receivers
$c_{i,j}$	residual capacity for session i on link l_j
$u_{i,j}$	number of unassigned connections in session i on link l_j
u_j	number of unassigned sessions on link l_j

A. Connection Fairness (Max-Min Fairness)

We first present the traditional connection max-min fairness definition, for comparison with our session fairness definitions.

Definition: An allocation of sender rates $a_{i,k}$ is said to be *connection fair* [9] if it is feasible and for any alternative feasible allocation of sender rates where $\overline{a}_{i,k} > a_{i,k}$, there is some other sender $s_{i',k'} \neq s_{i,k}$ such that $a_{i,k} \geq a_{i',k'} > \overline{a}_{i',k'}$.

An allocation is said to be *feasible* if each sender $s_{i,k}$ is assigned a sending rate $a_{i,k}$, subject to the constraint

$$\forall j : \{\sum a_{i,k} : s_{i,k} \in R_j\} \leq C_j$$

Informally, the rate of a connection at its bottleneck link is greater than or equal to the rate of any other connection at that link. The feasibility condition ensures that the link is not loaded beyond its capacity. The algorithm for achieving this allocation is given in Bertsekas and Gallager [9].

Example allocation: We describe the working of the algorithms with the help of an example. Figure 3(a) shows the topology considered. Session S_1 consists of sender $s_{1,1}$ sending to receiver r_1. Session S_2 consists of two senders, $s_{2,1}$ and $s_{2,2}$, sending to r_2. The capacities of the links are as shown in Figure 3(a).

Figure 3(b) shows the connection fair rate allocation. Link a-b is the bottleneck link for all three connections, and hence each connection gets a rate of 2 units. Session S_1 gets a total rate of 2 units, and session S_2 gets a total rate of 4 units.

Properties of the Allocation: We discuss properties of the allocation from a session point of view. These properties are in addition to the normal connection rate properties.

- If all connections in all sessions have the same bottleneck link, the session rates will be in proportion to the number of connections in the session.
- Two connections between the same sender and receiver, belonging to different sessions will be bottlenecked at the same link and will receive equal rates.

B. Normalized Rate Session Fairness (NRSF)

We now present the two session fair definitions and discuss them in detail. We also present algorithms to achieve the allocations and then discuss some properties of the allocations.

Definition: An allocation of sender rates $a_{i,k}$ is said to be *normalized rate session fair* if it is feasible and for any alternative feasible allocation of sender rates where the normalized rate $\overline{z}_{i,k} > z_{i,k}$, there is some other sender $s_{i',k'} \neq s_{i,k}$ such that $z_{i,k} \geq z_{i',k'} > \overline{z}_{i',k'}$.

The *normalized rate of a connection* is defined as

$$z_{i,k} = \frac{a_{i,k}}{w_{i,k}},$$

where $w_{i,k}$ is the *weight of the connection*, subject to the constraints:

$$\sum_k w_{i,k} = 1$$
$$w_{i,k} \leq 1$$

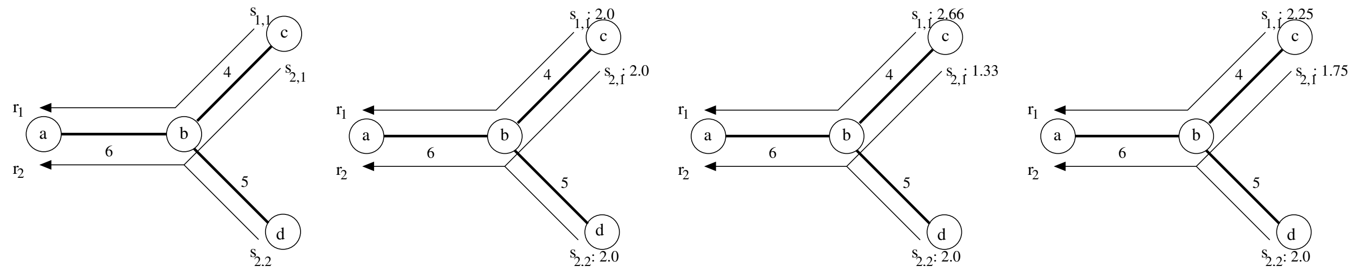
Informally, the normalized rate of a connection at its bottleneck link is greater than or equal to the normalized rate of any other connection at that link. The first constraint on the weights ensures that the set of connections in each session behaves as at most one connection throughout the data path of the session, and hence as at most one connection on each link that it traverses. The rate assigned to each connection at its bottleneck link will be in proportion to its weight[6]. The second constraint ensures that no connection behaves as more than one connection on any link.

The session fairness definition can be extended to an *r-parallel connection fairness* definition if each session behaves as at most r connections throughout the data path of the session. This approach lies between the connection-fair and the session fair rate allocation approaches, with $r = 1$ corresponding to the session fair approach and $r =$ number of connections corresponding to the connection-fair approach. r-parallel connection fairness can be achieved by defining the *normalized rate of a connection* as

$$z_{i,k} = \frac{a_{i,k}}{w_{i,k}},$$

[5] For consistency, we adopt a notation similar to that used in Rubenstein et al. [20].

[6] Moh and Chen [18] also use the concept of weights for each connection, but their definition and algorithm differ from ours.



(a) Session Layout

(b) Connection fair allocation (S_1 : 2.0, S_2 : 4.0)

(c) Normalized rate session fair allocation (S_1 : 2.66, S_2 : 3.33)

(d) Per-link session fair allocation (S_1 : 2.25, S_2 : 3.75)

Fig. 3. (a)Example network, (b)Connection fair, (c)Normalized rate session fair, (d)Per-link session fair

subject to the constraints:

$$\sum_k w_{i,k} \leq r$$
$$w_{i,k} \leq 1$$

The second constraint ensures that no connection behaves as more than one connection on any link.

Algorithm: We first assign weights to each connection according to the intra-session fairness approach that the client wants, as described in Section III. For "source and session fairness", we assign equal weights to each connection, for "data and session fairness", we assign weights to each connection in proportion to the size of data retrieved from that server, and for "path and session fairness", we assign weights to each connection according to the sharing of the data path of the session. For simplicity, we describe the normalized rate session fair algorithm with all clients using "source and session fairness".

In every iteration, we compute the normalized rate attainable on every link and saturate the link with the minimum normalized rate. The connections which get saturated as a result are bottlenecked at that link. Thus, after every iteration, at least one link will be saturated, and the algorithm will terminate when all connections have a bottleneck link. We expect the session rates to be fair with respect to each other because we have assigned weights subject to the constraint:

$$\sum_k w_{i,k} = 1$$

In fact, the weight of each connection gives an idea about the number of other connections constituting the session it belongs to.

The algorithm is given below, followed by a discussion of the steps involved.

```
1) $X = \{s_{i,k} : i = 1..m, k = 1..k_i\};$
   $\forall j, c_j = C_j;$
   $\forall s_{i,k}, a_{i,k} = 0$
2) while $(|X| \ > \ 0)$
3) $\forall j, x_j = \frac{c_j}{\sum_{s_{i,k} \in R_j} w_{i,k}}$
4) $x_{min} = min\{x_j\}$
5) $\forall s_{i,k} \in X :$
       $a_{i,k} \mathrel{+}= \ w_{i,k} * x_{min}$
       $\forall l_j$ along the path from $s_{i,k}$ to $r_i$
           $c_j \mathrel{-}= \ w_{i,k} \ * \ x_{min}$
6) $\forall j$, if $(c_j \ == \ 0)$[7]
       $X = X - \{s_{i,k} \in R_j\}$
7) repeat from step 2
```

We assume that each connection has a weight $w_{i,k}$. In step 1, we initialize the unsaturated sender set (senders in a session with no bottleneck link) to all the sender-receiver pairs in the topology. Each connection is assigned a rate $a_{i,k}$ of 0. In step 3, x_j is calculated as the ratio of the residual capacity of the link to the sum of the weights of all unsaturated connections traversing that link, that do not have a bottleneck yet. In step 4, we compute the minimum of all these x_j's. In step 5, the rate for all unsaturated connections is incremented by its weight, times the minimum value, thus saturating at least one link. The residual capacity on all links along the path from the senders in set X to the receivers is then decremented by that amount. In step 6, all senders that are saturated as a result are removed from set X. We repeat steps 2 through 7 until all receivers have a bottleneck link. The rate allocation thus determined is the required normalized rate session fair allocation.

Example allocation: Figure 3(c) shows the allocation achieved by the normalized rate session fair algorithm. The connection in session S_1 is assigned a weight of 1.0, and the two connections in session S_2 are each assigned a weight of 0.5. In the first iteration, the total weight on link a-b is 2.0, hence a weight of 1.0 would correspond to a rate of 3 units. The total weight on link b-c is 1.5, hence a weight of 1.0 corresponds to a rate of 4/1.5 = 2.66 units. The total weight on link b-d is 0.5, hence a weight of 1.0 corresponds to a rate of 10 units. The minimum normalized rate is 2.66, hence the algorithm will saturate link b-c first. $s_{1,1}$, which has a weight of 1.0, therefore gets a rate of 2.66 units, and $s_{2,1}$, which has a weight of 0.5, gets a rate of 1.33 units. In the next iteration, link a-b is saturated, and $s_{2,2}$ gets a rate of 2.0 units. Thus, session S_1 gets a

[7] At least one link corresponding to x_{min} will get saturated.

total rate of 2.66 units and session S_2 gets a rate of 3.33 units.

Properties of the Allocation: We present some properties of the allocation that follow in a relatively straightforward manner from the definition of normalized rate session fairness.

- If all sessions have an equal number of connections, they will share bandwidth as in the connection max-min fair allocation. This is because, the weights of all connections in all the sessions will be equal, and they will share the capacity on bottleneck links equally.
- If all connections in all sessions traverse the same bottleneck link, the rates allocated to each session will be equal because we have imposed the constraint:
  $$\sum_k w_{i,k} = 1$$
  on each session. Each session will therefore be treated as a single connection on that link. The connections within a session will share this session rate.
- All connections (possibly from different sessions) bottlenecked at the same link will have the same normalized rate:
  $$z_{i,k} = \frac{a_{i,k}}{w_{i,k}}$$
- Two connections between the same sender and receiver (assuming they follow the same path), belonging to different sessions with different number of connections (and hence having different weights, to be more precise), will be bottlenecked at the same link and will receive different rates, in proportion to their weights. However, their normalized rates will be the same.

C. Per-Link Session Fairness (PLSF)

Definition: An allocation of sender rates $a_{i,k}$ is said to be *per-link session fair* if it is feasible and for any alternative feasible allocation of sender rates $\overline{a}_{i,k}$, where $\overline{b}_{i,j} > b_{i,j}$ for session S_i at some link l_j, there is some other session $S_{i'} \neq S_i$ such that $b_{i,j} \geq b_{i',j} > \overline{b}_{i',j}$.

Informally, this means that the rate of a connection can be increased only by decreasing the rate of a session with an already lower or equal rate on a link at which the higher rate session, which comprises the connection, is bottlenecked. This also ensures that each session behaves like at most one connection on each link that it traverses. The basic idea here is that the capacity of the bottleneck link is shared equally by all the sessions traversing that link (and not by the individual connections). In other words, we try to achieve session fairness on every link.

The above definition does not lead to a unique allocation. This is because the per-link session fairness definition advocates the sharing of the session rate at each link in a fair manner. The session then splits this rate amongst the connections that traverse that link. This leads to residual capacity on some links. The manner in which this residual capacity is shared amongst the sessions can lead to multiple allocations which satisfy the above definition. We present an algorithm which will lead to one such allocation.

Algorithm: In this algorithm, we make the distinction between *virtual bottleneck links* and *physical bottleneck links*. Each link is considered to be a set of virtual links, one for each session on that link. In every iteration, at least one virtual link is saturated. Thus, at the end of one run of the algorithm, each connection has a virtual bottleneck link. But it might be the case that a connection does not have a physical bottleneck link on its path, and hence it can send at a higher rate. We therefore reassign the available capacities amongst all sessions without physical bottleneck links and reiterate through the algorithm. We do this until each connection has a physical bottleneck link on its path.

The algorithm is given below, followed by a discussion of the steps involved.

```
1)  $\forall j, c_j = C_j$;
    $\forall s_{i,k}, a_{i,k} = 0$
2)  $Y = \{s_{i,k} : i = 1..m, k = 1..k_i\}$
3)  while ($|Y| > 0$)
4)  $X = Y$
    $u_j$ = number of sessions on link $l_j$
    $u_{i,j}$ = number of connections in session $i$ on link $l_j$
5)  $\forall i, j$, if ($u_j \neq 0$)
        $c_{i,j} = \frac{c_j}{u_j}$
6)  while ($|X| > 0$)
7)  $x_{i,j} = \frac{c_{i,j}}{u_{i,j}}$
8)  $x_{min} = min\{x_{i,j}\}$
9)  $\forall s_{i,k} \in X$:
        $a_{i,k}$ += $x_{min}$
        $\forall l_q$ along the path from $s_{i,k}$ to $r_i$
            $c_{i,q}$ −= $x_{min}$
            $c_q$ −= $x_{min}$
            if ($c_{i,q}$ == $0$)[8]
                $X = X - \{s_{i,k} \in R_{i,q}\}$
                $\forall l_q$ on the path from these $s_{i,k}$'s to $r_i$,
                    $u_{i,q}$−−
            if ($u_{i,q}$ == $0$)
                $u_q$−−
            if ($c_j$ == $0$)
                $Y = Y - \{s_{i,k} \in R_j\}$
10) repeat from step 6
11) repeat from step 3
```

Set Y is the set of receivers without a physical bottleneck link. Set X is the set of receivers without a virtual bottleneck link in the current iteration. In step 5, each session is assigned the rate it is allowed to use on that link. This is the ratio of the residual capacity of that link to the number of sessions traversing that link. In step 7, each session then determines the rate of each connection it comprises as the ratio of the above rate to the number of unsaturated connections constituting that session on that link. In step 8, we determine the minimum of these rates, and increment the rates of all unsaturated connections by that amount. We thus saturate at least one virtual link in this step, and remove these connections from consideration in this iteration. If any

[8] At least one virtual link corresponding to x_{min} will get saturated.

physical edge has a residual capacity of zero, we remove those connections from consideration permanently. We repeat steps 6 through 10 until all connections are saturated by a virtual link. We then reassign the set X to the set Y and continue steps 3 to 11 until each connection has a physical bottleneck link. The resulting rate allocation is the per-link session fair allocation.

Example allocation: Figure 3(d) shows the allocation achieved by the per-link session fair algorithm. On link a-b, sessions S_1 and S_2 can each use a rate of at most 3 units. Session S_2 can therefore use at most 1.5 units for each connection, assuming that it shares the rate equally between the connections in the session. On link b-c, sessions S_1 and S_2 can each use at most 2 units. On link b-d, session S_2 can use at most 5 units. Since 1.5 units is the minimum rate that can be used by a connection in this scenario, the virtual link for session S_2 on link a-b is saturated in this iteration, and $s_{2,1}$ and $s_{2,2}$ each get a rate of 1.5 units. These two connections are then removed from contention on all the links that they traverse. In the next step, $s_{1,1}$ is assigned 0.5 units more, thus saturating its virtual link on b-c. Note that we did not reallocate the extra 0.5 units not used by session S_2 on link b-c, because we can reuse it in successive iterations. In the next run, we therefore reconsider all senders that are not saturated by a physical link. All three senders satisfy this condition— the residual capacity on link b-c is 0.5 units, that on link a-b is 1 unit, and that on link b-d is 3.5 units. $s_{1,1}$ and $s_{2,1}$ get 0.25 units each, in this round, thus saturating physical link b-c. $s_{2,2}$ also gets 0.25 units, thus saturating the virtual link for session S_2 on link a-b, but the physical link is not yet saturated. In the next round, the only connection in contention is $s_{2,2}$, and it gets the residual 0.25 units, thus saturating link a-b. Session S_1 therefore gets a rate of 2.25 units under the per-link session fair allocation, and session S_2 gets a total rate of 3.75 units.

Properties of the Allocation: The properties presented are for the allocation in general, and not just for the allocation achieved by this particular algorithm. They follow in a relatively straightforward manner from the definition of per-link session fairness.

- If all connections in all sessions traverse the same bottleneck link, the rates allocated to each session will be equal by definition. That is, each session will be treated as a single connection on that link. The connections within a session will share this session rate.
- Two connections between the same sender and receiver, belonging to different sessions will be bottlenecked at the same link if they are part of sessions with the same number of connections at the bottleneck link, and they will receive the same rates. If the connections belong to sessions with different number of connections on that link, they will receive different rates.

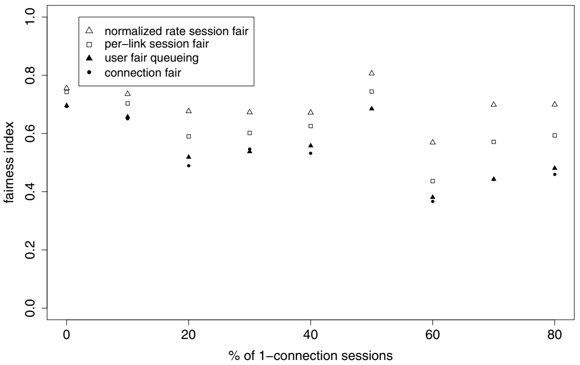


Fig. 4. Fairness index for sessions rates

To summarize this section, we have proposed two definitions for session fairness, and presented algorithms to achieve those allocations. We have also discussed some properties of these allocations.

V. Evaluation Results

This section describes an evaluation of the proposed session fair algorithms using simulations of a wide area network. Our goal is to evaluate the advantages offered by the proposed algorithms in an environment like the Internet, with data paths of sessions intersecting arbitrarily, and to ensure that the session fair allocations do not just degrade to the existing connection fair allocation. We compare the two session fair algorithms with each other, and with the original connection fair algorithm. We also compare these algorithms with the user fair queueing algorithm presented by Banchs [16].

A. Evaluation model

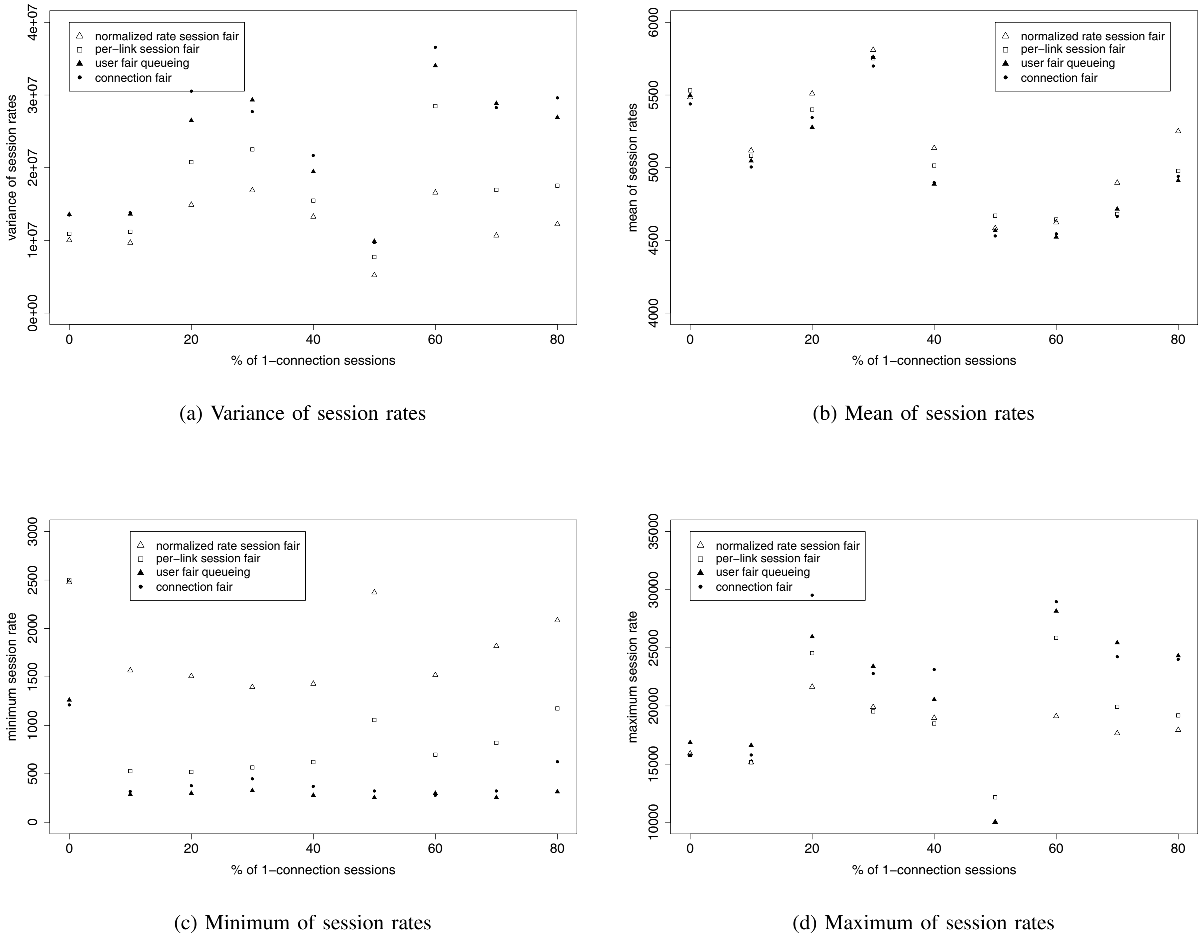
We implemented the four algorithms (normalized rate session fair, per-link session fair, connection fair, and user fair queuing) and computed the allocations achieved for various session configurations in the topologies discussed below.

We constructed transit-stub topologies of 100 nodes and 600 nodes using GT-ITM [22]. A number of sessions were then simulated on top of this topology, with varying percentages of clients and servers. These sessions comprised 1, 4 or 15 connections from multiple servers to a single client. We allowed for a single client to have multiple co-located sessions.

Based on a study by Krishnamurthy et al. [1], we varied the percentages of 1, 4 and 15-connection sessions, such that the average number of servers contacted by a client is between 4 and 9. We varied client and server percentages and their locations in the network, computed the allocations with the four algorithms and plotted the performance metrics discussed in Section V-B. A typical plot for the performance metrics is analyzed in Section V-C.

B. Performance metrics

The performance metrics used for comparison of the algorithms, and their significance are discussed below.



(a) Variance of session rates

(b) Mean of session rates

(c) Minimum of session rates

(d) Maximum of session rates

Fig. 5. Performance metrics for session rates

- *Raj Jain's fairness index*: This is a commonly accepted fairness performance measure, and is defined as:
$$f(x_1, x_2, \ldots x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} {x_i}^2}$$
It accounts for variability in the x_i's of all the users. The index assumes a range of values between 0 and 1. A fairness index of 1 implies an equal distribution of x_i's. The lower the value of the fairness index, the more unfair the distribution is. For calculation of the fairness index for session rates, the x_is are the total data rates of each session. For calculation of the fairness index for connection rates, the x_is are the rates allocated to each connection by the algorithm.
- *Variance* of session rates: A higher variance in session rates implies that the session rates are unevenly distributed across all sessions. Hence, a lower variance indicates a more fair distribution.
- *Mean* session rate: A higher mean session rate indicates a higher utilization of available bandwidth resources.
- *Minimum* and *Maximum* session rates: The minimum and maximum session rates are just two extreme points in the distribution. Since the session with the lowest rate is least satisfied, a fair algorithm would try to increase the minimum session rate. Also, a fair algorithm would try to redistribute the excess rate in the session with the maximum rate amongst sessions with lower total rates. Hence, an algorithm with a higher minimum session rate and a lower maximum session rate can be considered more fair.

C. Comparison of the Algorithms

Figures 4 and 5 show typical plots of the performance metrics discussed in the previous section. These plots are for a 100-node topology with 30 servers and 20 clients. 20% sessions have 15 connections, and we vary the percentage of 1-connection and 4-connection sessions between 0% and 80%. The x-axis for each graph is the percentage of sessions with 1 connection, and therefore, the percentage of sessions with 4 connections is $(80 - x)\%$.

1) Comparison of the two session fair algorithms and the connection fair algorithm: From Figure 4, it is clear that the fairness indices for the normalized rate session fair algorithm are higher than the per-link session fair and connection fair algorithms. It thus achieves the most fair allocation of session rates for this configuration.

The variances in session rates, as seen from Figure 5(a), are much higher with the connection fair algorithm than the per-link session fair algorithm and the normalized rate session fair algorithm. In some cases, the connection fair algorithm has a variance as much as two or three times that of the normalized rate session fair algorithm, thus giving a rather unfair allocation.

Although the mean session rates, as seen in Figure 5(b), are almost the same for all three algorithms, in most cases the normalized rate session fair algorithm achieves a higher mean than the connection fair algorithm and the per-link session fair algorithm. Note that a higher mean session rate implies a higher overall bandwidth utilization in the network.

As seen in Figure 5(c), the minimum session rate achieved by the normalized rate session fair algorithm is always greater than that achieved by the per-link session fair algorithm, which is again greater than that achieved by the connection fair algorithm. On the other hand, as seen from Figure 5(d), which plots the maximum session rate, the connection fair algorithm has a higher maximum session rate most of the times. The session fair algorithms try to decrease the maximum rate achieved by any session, and redistribute that rate amongst sessions with lower rates, thus increasing the minimum rate. Thus, the normalized rate session fair algorithm is more fair than the connection fair algorithm.

Thus, we can conclude that the session rates achieved by the session fair algorithms are more fair than those achieved by the connection fair algorithm, while maintaining or increasing overall bandwidth utilization.

2) Comparison with the User Fair Queueing (UFQ) Algorithm: In the normalized rate session fair algorithm, we need to be careful when we assign weights to the connections in each session. Our algorithm assigns weights in a session at the client or receiver end, subject to the condition $\sum_k w_{i,k} = 1$. However, if the weights are distributed amongst all connections at the sender, as suggested in the user fair queueing algorithm [16], the session-rate distribution may not be fair.

In Figure 5 the fourth algorithm plotted is the user fair queueing algorithm, which is basically the normalized rate session fair algorithm, with weights assigned at the server. As can be seen, the minimum session rate achieved by this algorithm is often lower than even that achieved by the connection fair algorithm. The variance is almost as high as that with the connection fair algorithm, or sometimes higher, and the mean session rate (and hence utilization) is lower than that with the connection fair algorithm at times. This is because, in user fair queueing, if a sender has many clients (N), the weight assigned to each connection is $1/N$. Effectively, some multipoint-to-point session might get a total weight of more than 1, and if two such sessions share a bottleneck link for all their connections, one session will get a higher share than the other session. The normalized rate session fair algorithm performs better than the user fair queueing algorithm in all cases.

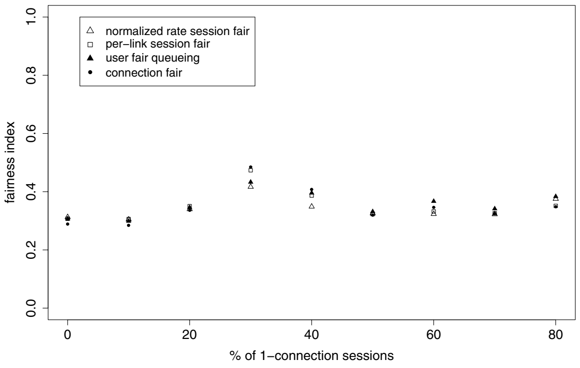


Fig. 6. Fairness index for connection rates

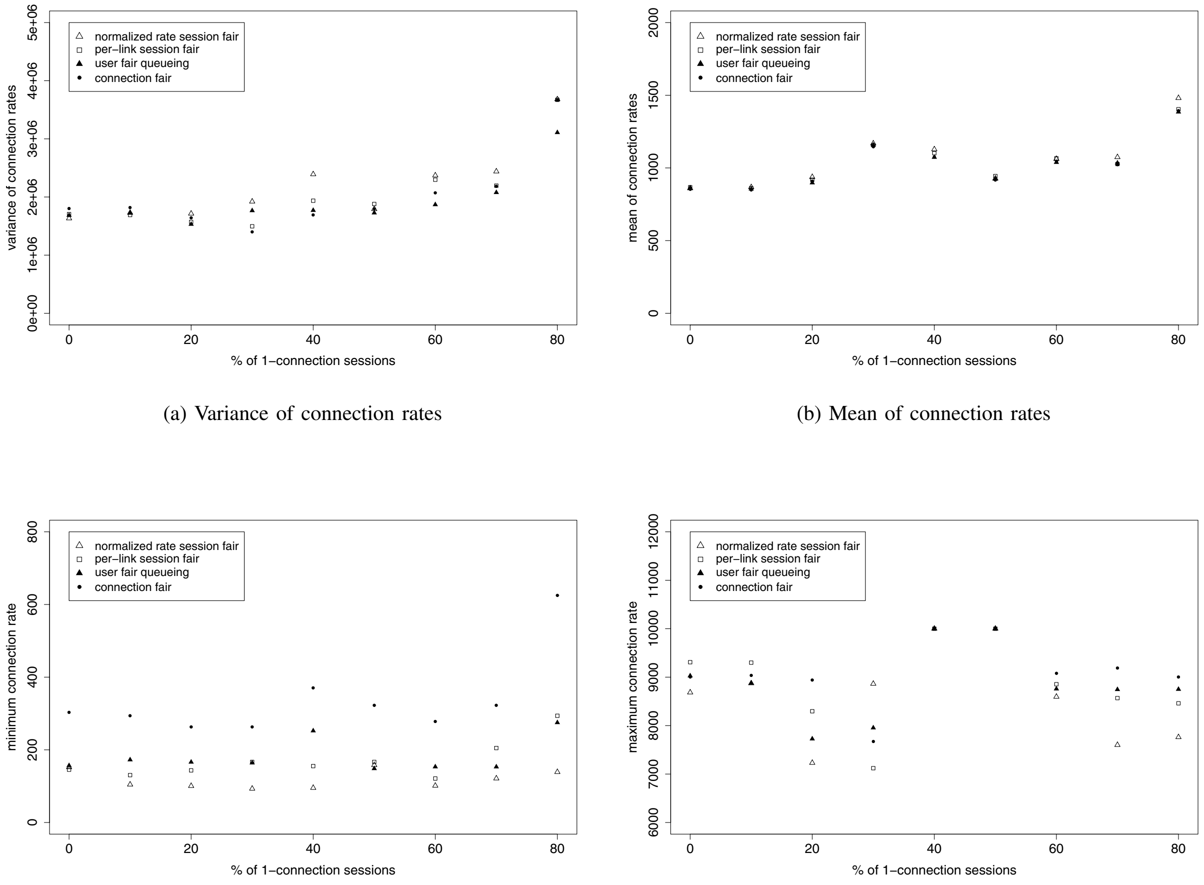
3) Comparison of Connection Rates: The improved session fairness does not come without a penalty. The performance metrics for the *connection rates* are shown in Figures 6 and 7. The x-axis shows varying percentages of 1-connection sessions in the network. On the y-axes we plot minimum, maximum, variance and mean of the connection rates in the network. It is clear that the minimum connection rates for the session fair algorithms are lower than those for the connection-fair algorithm and the variances are higher. This is the tradeoff seen due to our proposal of achieving fairness at the session level, as opposed to fairness at the connection level.

Thus, we conclude that the session fair algorithms, while achieving the same or higher total utilization as the connection fair algorithm, distribute the rate allocations more evenly among sessions.

VI. Implementation Issues

We present some preliminary thoughts on the challenges involved in implementing the session fair algorithms proposed. Both the algorithms assume global knowledge of the residual capacities of each link, and information about all connections in all sessions. Thus, these algorithms are inherently centralized and do not scale well. Modifications to these algorithms are necessary in order to implement them in a distributed and scalable manner.

The normalized rate session fair algorithm is amenable to a distributed end-to-end implementation. On the other hand, because of the need to compute the session rate at every link, the per-link session fair algorithm appears to require a more complex implementation. Also, as seen from the evaluations, the normalized rate session fair allocation performs better than the per-link session fair allocation. Hence, we focus on the normalized rate session fair algorithm in the following discussion.



(a) Variance of connection rates

(b) Mean of connection rates

(c) Minimum of connection rates

(d) Maximum of connection rates

Fig. 7. Performance metrics for connection rates

The three important issues in the implementation of the session fair algorithms are as follows:

- Identification of the transport connections belonging to the same session.
- Communication between connections within the same session about the existence of other connections in the session.
- Estimation of the sending rate of each connection, in accordance with the session fair algorithms, and then achieving that rate.

The issue of session identification of a connection, and communication between connections in a session can be implemented explicitly by each application, or by extending the Session Control Protocol [23] or Session Initiation Protocol [24]. The Integrated Congestion Management architecture [15] includes the ability to communicate information between different connections, though clearly the appropriate type and use of information must be modified to meet our needs. More work is needed to develop techniques for estimating the correct sending rate and incorporating rate adjustment into existing protocols (e.g. by modifying TCP acknowledgement behavior).

Our future work will continue to explore these options with the goal of designing and building a workable system for session fair rate control.

VII. Concluding Remarks

In the current Internet, many applications start sessions with multiple connections to multiple servers in order to expedite the reception of data. Such aggressive behavior can be viewed as unfair sharing of available bandwidth. This has been our motivation to propose the notion of session fairness when allocating rates to connections. In particular, we looked at *static multipoint-to-point sessions* which comprise multiple connections from multiple senders to a single client, starting and terminating at approximately the same time. We explored the session fairness space and proposed and evaluated two definitions and algorithms to achieve the definitions. The *normalized rate session fair* algorithm achieves a higher level of session fairness, while achieving the same or higher network utilization, as compared to the connection fair algorithm. The *per-link session fair* algorithm also performs better than the connection fair algorithm, but not as well as the normalized rate session fair algorithm. We discussed some of the issues involved in implementing these algorithms and presented some preliminary thoughts on options for implementation. The normalized rate session fair algorithm appears to be easier to implement than the per-link session fair algorithm.

Our future work will involve implementation of a protocol to realize session fair algorithms, as well as investigation of fairness issues for *dynamic multipoint-to-point sessions.*

REFERENCES

[1] B. Krishnamurthy, C. Wills, and Y. Zhang, “On the Use and Performance of Content Distribution Networks,” in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[2] J. Apostolopoulos, T. Wong, S. Wee, and D. Tan, “On Multiple Description Streaming with Content Delivery Networks,” in *Proceedings of IEEE Infocom*, 2002.

[3] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, “Informed Content Delivery Across Adaptive Overlay Networks,” in *Proceedings of ACM Sigcomm*, 2002.

[4] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, “Distributing Streaming Media Content Using Cooperative Networking,” in *Proceedings of NOSSDAV*, 2002.

[5] “Peergenius,” http://www.peergenius.com/.

[6] “Centerspan,” http://www.centerspan.com/.

[7] “Digital fountain,” http://www.digitalfountain.com/.

[8] P. Rodriguez, A. Kirpal, and E. Biersack, “Parallel-Access for Mirror Sites in the Internet,” in *Proceedings of IEEE Infocom*, 2000.

[9] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ, Prentice-Hall, 1992.

[10] H. Balakrishnan and S. Seshan, “The Congestion Manager,” Internet Engineering Task Force, Request for Comments 3124, 2001.

[11] V. Padmanabhan, “Coordinating Congestion Management and Bandwidth Sharing for Heterogenous Data Streams,” in *Proceedings of NOSSDAV*, 1999.

[12] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley and Sons Inc., 1991.

[13] J. Gettys and H. Nielsen, “The WEBMUX protocol, Internet Draft (work in progress),” http://www.w3.org/Protocols/MUX/WD-mux-980722.html, 1998.

[14] L. Eggert, J. Heidemann, and J. Touch, “Effects of Ensemble-TCP,” *ACM Computer Communication Review*, vol. 30, no. 1, pp. 15–29, January 2000.

[15] H. Balakrishnan, H. Rahul, and S. Seshan, “An Integrated Congestion Management Architecture for Internet Hosts,” in *Proceedings of ACM Sigcomm*, 1999.

[16] A. Banchs, “User Fair Queueing: Fair Allocation of Bandwidth for Users,” in *Proceedings of IEEE Infocom*, 2002.

[17] S. Fahmy, R. Jain, R. Goyal, and B. Vandalore, “Fairness for ABR multipoint-to-point connections,” in *Proceedings of SPIE Symposium on Voice, Video and Data Communications,*, November 1998.

[18] M. Moh and Y. Chen, “Design and Evaluation of Multipoint-to-Point Multicast Flow Control,” in *Proceedings of SPIE Symposium on Voice, Video and Data Communications,*, November 1998.

[19] J. Saltzer, D. Reed., and D. Clark, “End-to-End Arguments in System Design,” *ACM ToCS*, Nov. 1984.

[20] D. Rubenstein, J. Kurose, and D. Towsley, “The Impact of Multicast Layering on Network Fairness,” in *Proceedings of ACM Sigcomm*, 1999.

[21] R. Liston and E. Zegura, “Using a Proxy to Measure Client-Side Web Performance,” in *Proceedings of the 6th International Web Caching Workshop and Content Delivery Workshop*, June 2001.

[22] K. Calvert, M. Doar, and E. Zegura, “Modeling Internet Topology,” *IEEE Communications Magazine*, June 1997.

[23] S. Spero, “Session Control Protocol,” http://www.w3.org/Protocols/HTTP-NG/http-ng-scp.html, 1998.

[24] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, “SIP: Session Initiation Protocol,” Internet Engineering Task Force, Request for Comments 2543, 1999.