# Global Flow Control for Wide Area Overlay Networks: A Cost-Benefit Approach

Yair Amir, Baruch Awerbuch, Claudiu Danilov, Jonathan Stanton

*Abstract*—**This paper presents a flow control protocol for multi-sender multi-group multicast and unicast in wide area overlay networks. The protocol is analytically grounded and achieves real world goals, such as simplicity, fairness and minimal resource usage. Flows are regulated based on the "opportunity" costs of network resources used and the benefit provided by the flow. In contrast to existing window-based flow control schemes, we avoid end-to-end per sender or per group feedback by looking only at the state of the virtual links between participating nodes. This produces control traffic proportional only to the number of overlay network links and independent of the number of groups, senders or receivers. We show the effectiveness of the resulting protocol through simulations and validate the simulations with live Internet experiments.**

*Keywords*— **Flow Control, Overlay Networks, Cost-Benefit.**

## I. Introduction

THIS paper [1] presents the *Cost-Benefit* framework, a new global flow control strategy for wide area overlay network multicast based on "economic" principles and competitive analysis. Our framework assigns costs to network resources, and benefits to achieving user goals such as multicasting a message to a group or receiving a message from a group.

Intuitively, the cost of network resources, such as buffers in routers, should go up as the resource is depleted. When the resource is not utilized at all, its cost should be zero. When the resource is fully utilized, its cost should be prohibitively expensive. Finding the best cost function is an open question. However, it has been shown theoretically that using a cost function that increases exponentially with the resource's utilization is *competitive* with the optimal offline algorithm[1]. This theoretical result was demonstrated to work in practice in the context of cluster computing[2]. In this paper, we use a similar cost function to price the buffers in the overlay routers.

Authors are with the Department of Computer Science at the Johns Hopkins University, Baltimore, MD 21218 USA. Email:{yairamir, baruch, claudiu, jonathan}@cs.jhu.edu Web: http://www.cnds.jhu.edu

This work was partially supported by DARPA and NSA under grants F30602-00-2-0626 and F30602-00-2-0550

[1] this is an updated version of CNDS-2001-1 tech report (see http://www.cnds.jhu.edu/publications/)

Our algorithm decides to allow use of resources if the benefit attached to that use is greater than the total cost of allowing the use. The choice of benefit function allows us to optimize for various goals. By adjusting the benefit function, performance issues such as throughput, as well as policy issues such as fairness, can be taken into account when making flow control decisions. In this way our framework allows adjustable trade-offs between conflicting properties such as fairness and throughput. For example, the benefit can be the number of packets sent (sending throughput), the number of packets received by all receivers (receiver throughput), or the average latency given some throughput constraints. In this paper we only use the sender throughput benefit function.

We define an overlay network as a virtual network constructed such that each link connects two edge nodes in an underlying physical network, such as the Internet. Each virtual link in the overlay network can translate into several hops on the underlying network. The characteristics of a virtual link, such as latency, bandwidth, and losses are the aggregate of the underlying network links over which it travels.

We use a standard congestion control protocol on each of the overlay links. This results in a dynamic capacity available to our flow control framework on every overlay network link. The global flow control problem is a dynamic optimization problem, optimizing some benefit subject to the dynamic capacity constraints of the overlay links. All of the traffic generated by our system on a link will be seen as one TCP flow on that link, regardless of the number senders or receivers. This provides a very conservative level of fairness between our multicast traffic and competing TCP flows.

Our Cost-Benefit framework is evaluated through both simulations and live tests on the Internet. The simulations use the ns2 simulator[3] and examine the behavior of several overlay network configurations. To conduct actual network tests we extended the available Spread group communication system[4] to implement our flow control protocols, and conducted experiments using this software on the CAIRN network[5].

Both the simulations and the actual CAIRN experiments show promising results. We show a fair sharing of individual congested links between both individual clients in a flow and between different flows. We show a fast and stable response to changing capacities on the network and to competing traffic. We demonstrate that senders can each achieve their own near optimal sending rate without being constrained by the ability (or lack thereof) of other senders.

The rest of this paper is organized as follows. After presenting the related work in Section II we describe the network model we use and the overall architecture of our protocol in Section III. We present our framework, the theoretical model supporting it, and its adaptation to practical settings in Section IV. Section V briefly discusses the issues of network and inter-flow fairness, and scalability. Simulation results and live Internet-based experiments are presented in Section VI and Section VII. Section VIII concludes the paper.

## II. Related work

Many different approaches exist in the flow control literature, including TCP like window based protocols [6], [7], one or two bit feedback schemes [8], [9], [10], and optimization based flow control [11], [12], [13], [14], [15], [16]. The economic framework for flow and congestion control used in many optimization based protocols[12], [14] has some similarity with the cost-benefit model used in our work. In both, the links have some cost and packets that are sent must have sufficient benefit to pay the cost of the network resources they require. A significant difference is that our cost-benefit model takes an algorithmic approach using a simple formula to decide when a packet can be sent, and is not based on economic theory. Unlike many economic models our cost-benefit model does not try to reach an equilibrium state or influence non-cooperative processes to behave, but rather optimizes the throughput under the assumption of minimally cooperative senders.

Research on protocols to support group communication across wide area networks such as the Internet has begun to expand. Recently, new group communication protocols designed for such wide area networks have been proposed [17], [18], [19], [20] which continue to provide the traditional strong semantic properties such as reliablity, ordering, and membership. These systems predominantly extend a flow control model previously used in local area networks, such as the Totem Ring protocol[19], or adapt a window-based algorithm to a multi-sender group[21], [20]. Our work presents a flow control algorithm designed explictly for wide-area overlay networks which is motivated more by networking protocols and resource optimization research, than by existing group communication systems.

The research in flow control specificially for group communication systems is sparse. A 1998 work by Mishra and Wu[22] surveyed the flow control used in existing systems and compared archetypal optimistic and conservative flow control algorithms. The flow control discussed in that work was used to globally limit the number of outstanding update messages that have not become stable yet. One other paper[21] discusses some general approaches to flow control for group communication and studies extending window-based protocols to groups using both local and global information.

Work on flow control for multicast sessions has occured mainly in the context of the IP-Multicast model. Much of this work has focused on the congestion control problem, avoiding extra packet loss and providing fairness, and has left flow control up to higher level protocols (such as reliability, ordering, or application level services). Research has explored the difficult problems associated with multicast traffic such as defining fairness[23], [24] and determining appropriate metrics for evaluation of multicast traffic[25]. A number of congestion control protocols have been developed with the goal of providing some level of fairness with TCP traffic, while taking advantage of the unique characteristics of multicast traffic. These include window based protocols[26], rate based protocols[27], [28], multi-layer based protocols[23], and protocols that use local recovery to optimize congestion control[29]. While IP-Multicast focuses on a single sender, single group approach that scales to many receivers and many intermediate routers, our approach addresses a multi-group multi-sender problem that scales with number of groups, senders and receivers, but is defined in an overlay network setting rather than on every router in the Internet.

The Distributed Core Multicast (DCM)[30] routing protocol uses a network architecture similar to ours with tunnels through the backbone network, and focuses on similar scalability goals of supporting many small groups. DCM is complimentary to our work as it focuses on efficient routing for such groups, while we provide flow control algorithms.

## III. Architecture

The overlay network model used is a graph with nodes and overlay links. Each node on the graph represents a host running a daemon program. Each overlay link is a unicast link between two nodes, which may be a long path traversing multiple routers and physical links in the Internet as is seen in Figure 1. Each daemon chooses a tree from this graph based on the network topology, in which it will multicast messages. This tree is rooted at the daemon node and will not necessarily be the same as any other

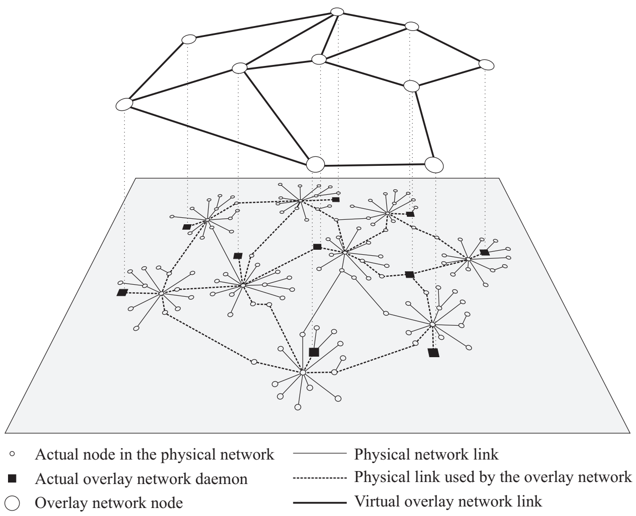


Fig. 1. Overlay Network Architecture

daemon's tree. The daemon provides multicast services to clients. As the daemon also acts as a router, we will refer to daemon and router interchangeably. Each daemon can have many clients connected to it.

Each client may join an arbitrary number of groups, and may send multicast messages to any number of groups, including ones it has not joined.

Clients connect to whichever daemon they like, usually the closest one, and that daemon handles the forwarding of their traffic and providing all required semantics. The connection from a client to a daemon is either a TCP/IP connection or a local IPC mechanism such as Unix Domain Sockets. Each client can multicast and receive messages at any time. In this approach each daemon may support many distinct clients who are actually running on many different hosts.

The performance of the entire system is improved if each client connects to the daemon closest, network-wise, to them, however clients may connect from anywhere. In practice, the clients connected to a specific daemon may actually use a local group, or local area reliable multicast protocol instead of unicast TCP connections to connect the clients to the daemons. This modification would improve the performance for the group of clients, but does not change the basic model as long as the protocol used allows individual control of each sender's sending rate. The Spread system that we utilize in our experiments actually uses such a protocol.

Each message carries some information about its source and destination nodes. When an intermediate node receives a message, it forwards it through its links that have downstream destinations.

In a multiple sender system, each sender may have a different rate at which it can reach the entire receiver group, and different senders may reach the group over different multicast trees. So, the bottleneck link for one sender may not be the bottleneck for other senders. The obvious goal is to allow each sender to achieve their highest sending rate to the group, not limit them by what other senders can send to that group. To achieve this, rate regulation must occur on a per-sender basis rather than on a single flow control limit for the entire group. The result is a flow control that provides fine granularity of control (per-sender, per-group).

**The Spread group communication toolkit:** We implemented our global flow control algorithm in the Spread wide area group communication system[4], [31]. The Spread system provides a similar architecture to our model with daemons running on end-hosts acting as routers in an overlay network. Spread provides strong semantics for messages including reliable multicast, message ordering guarantees (unordered, fifo, agreed), and a membership service supporting Extended Virtual Synchrony (EVS)[32] and Virtual Synchrony (VS)[33] models. It is designed to support a small to medium number of members of a group (1-1000's), with a large number of active groups and many senders. As such, it has different design goals than most IP-Multicast systems, which support larger groups but focus on the single-sender model and require state in every network router for every group.

Routing in Spread is based on shortest-path multicast trees rooted at each daemon. The routing is recalculated whenever the set of connected daemons changes (and not when clients join or leave groups, or connect or disconnect from the system).

The Spread system provides end-to-end reliablity by using a reliable point-to-point protocol for each link on the overlay network and requiring that the daemons do not drop any messages between links[20]. Spread provides a choice of several link protocols from which we chose to use TCP.

## IV. GLOBAL FLOW CONTROL FOR WIDE AREA OVERLAY NETWORKS

The algorithmic foundation for our work can be summarized as follows: We price links based on their "opportunity cost", which increases exponentially with link utilization. We compare different connections based on the total opportunity cost of the links they use, and slow down connections with large costs, by delaying their packets at the entry point.

### A. Algorithmic foundation

Whether a message is accepted or not into the system by a daemon is an online decision problem. At the time of acceptance we don't know how much data the sending client (or the other clients) will be sending in the future, nor at what specific times in the future.

**Evaluation of algorithms – Competitive analysis:** The general problem with online allocation of resources is that it is impossible to optimally make irreversible decisions without knowing the future nor the correlations between past and future. Thus, the goal is to design a "competitive" algorithm whose total accrued benefit is comparable to that achieved by the optimal offline algorithm, on *all* input instances. The maximum possible performance degradation of an online algorithm (as compared with the offline) is called the "competitive ratio". Specifically,

$$\rho = \max_{x} \frac{C(x)}{C^*(x)}$$

where $x$ is the input sequence, $C(x)$ is the cost of the online algorithm, and $C^*(x)$ is the cost of optimal offline algorithm on sequence $x$.

Our goal is to design an algorithm with a small competitive ratio $\rho$; such an algorithm is very robust in the sense that its performance is not based on unjustified assumptions about probability distributions or specific correlation between past and future.

**Theoretical background for the cost-benefit framework:** Our framework is based on the theoretical result in[1]. The following components are present in this framework.

- user benefit function is defined, defining how much benefit a given user extracts out of ability to gain resources, e.g., ability to communicate at a certain rate.
- resource opportunity cost is defined, based on utilization of the resource. The cost of unused resource is the lowest possible connection benefit, and the cost of fully used resource is the maximum connection benefit.
- a connection is admitted into the network if the opportunity cost of resources it wishes to consume is lower than its benefit.
- flow control is accomplished, conceptually, by dividing the traffic stream into smaller sub-streams and applying the above admission control framework for each sub-stream.

**Model of the resource – Cost function:** The basic framework revolves around defining, for each resource, the current *opportunity cost*, which is, intuitively, the benefit that may be lost by *high-benefit* connections as a result of consuming the above resource by a *lower-benefit* connection.

Since the goal is to maximize the total benefit, it is "wasteful" to commit resources to applications (connections) that are not "desperate" for that resource, i.e., not enjoying the maximal possible benefit from obtaining this resource. On the other hand, it is equally dangerous to gamble that each resource can be used with maximal benefit gained without knowing the sequence of requests ahead of time.

For the purpose of developing the intuition, it is useful to consider a somewhat restrictive setting where the resources are either assigned forever, or rented out for specific time. For a given resource $l$, (e.g., bandwidth of a given link $l$), denote by $u_l$ the normalized utilization of the resource, i.e., $u_l = 1$ means the resource is fully utilized and $u_l = 0$ means that the resource is not utilized at all. Also, let $\alpha$ be the minimum benefit value of a unit of a resource used by a connection and $\beta$ be the maximum value. Let $\gamma = \beta/\alpha$. The opportunity cost is now defined as:

$$C(u_l) = \alpha \cdot \gamma^{u_l} \tag{1}$$

Each $1/\log_2 \gamma$ of the fraction of the utilized resource necessitates doubling the price.

Notice that the opportunity cost of an unused resource starts at $\alpha$ per unit, i.e. *any* connection can "afford" it, and the cost of fully utilized resource is $\beta$, i.e., no connection can afford it.

**Model of the user – Benefit function:** This is part of the users specifications, and is not part of our algorithms. Each user (connection) associates a certain "benefit function" $f(r)$ with its rate $r$. The simplest function $f(r) = r$ means that we are maximizing network throughput; a linear function means we are maximizing weighted throughput.

More interestingly, a concave function (second derivative negative, e.g. $\sqrt{r}$) means that there is curve of diminishing return associated with rate allocation for this user. For example, imagine that a traffic stream is encoded in a layered manner, and it consists of a number of streams, e.g., first 2KB is voice, next 10KB is black and white video, and last 50KB is color for the video stream. In this case, a concave benefit function may allocate \$10 for voice part, \$5 for video and \$2 for color.

Notice that concave functions enable one to implement some level of fairness: given 50KB of bandwith, it is most "beneficial" to provide voice component for 25 connections, rather than voice + black and white video + color for a single connection since \$10 x 25 = \$250 is the total benefit in the former case, and \$10 + \$5 + \$2 = \$17 is the total benefit in the latter case.

**Example: Online auction model** In fact, let us focus on the following simple case of auctioning off an arbitrary resource, say link capacity, in an online setting where the bids arrive sequentially and un-predictably.

The *input* to the problem is the sequence of bids $B_1, B_2, B_k$that are positive numbers in the range from $\alpha$ to $\beta$, generated online at times $t_1, t_2, t_k$; each bid requests fraction $r_i$ of the total resource.

The *output* is a sequence of decisions $D_i$ made online, i.e. at times $t_1, t_2, t_k$, so that $D_i = 1$ if bid is accepted and $D_i = 0$ otherwise. The total benefit of the auction is $\sum B_i{\cdot}r_i{\cdot}D_i$ and the inventory restriction is that$\sum D_i{\cdot}r_i \leq C$ where $C$ is the resource capacity.

The question is what is the optimal online strategy for decision making without knowing the future bids, given that decisions to accept "low" bids cannot be reversed after knowing about future higher bids. On the other hand, it is dangerous to wait for high bids since they may never arrive.

As shown in[1], the problem of designing a competitive online algorithm for allocating link bandwidth has a lower bound of $\Omega(\log \gamma)$ on the competitive ratio, where $\gamma$ is the ratio $\gamma = \beta/\alpha$ between maximal and minimal benefit, achievable if $1/\log_2 \gamma$ of the fraction of the utilized resource necessitates doubling the price of the resource.

Since $1/\log_2 \gamma$ of the fraction of the utilized resource necessitates doubling the price, then at least $1/\log_2 \gamma$ fraction of the utilized capacity has been sold at at least 1/2 of the final price $P$, i.e. online gains at least

$$ALG > P/(2 \cdot \log_2 \gamma)$$

The total "lost" benefit, i.e. benefit of all the connection accepted by prescient and rejected by online, is at most $P$, i.e.

$$OFF - ALG < P$$

It follows that

$$OFF/ALG < 1 + 2 \cdot \log_2 \gamma$$

**Admission control in a circuit-switched environment:** Admission control on a single link is exactly auctioning of bandwidth on that link. Consider, as a toy example of the cost-benefit framework, the problem of admission control in the case of a permanent virtual circuits environment where an online request for a point-to-point virtual circuit is either *accepted* (with the permanent bandwidth allocation) or *rejected*. The goal is to maximize the bandwidth of all accepted connections.

In this case, the opportunity cost of the path is the sum of opportunity costs of all the links which make up the path.

$$C_p = \sum_{l \in L_p} C(u_l)$$

Also, the minimal benefit of a connection *per link* needs to be adjusted to

$$\alpha' = \alpha/d$$

where $d$ is the number of hops on a path. Thus, the base of exponent becomes

$$\gamma = \frac{\beta}{d\alpha'}$$

### B. *Adapting the model to practice*

The above theory section shows how bandwidth can be rationed with a Cost-Benefit framework leading to a near-optimal (competitive) throughput in the case of managing *permanent* connections in circuit-switched environments.

The core theory has several assumptions which do not exactly match the reality in distributed systems and communication networks. We will examine these assumptions and how we adapted the ideas of the Cost-Benefit framework to work in real networks.

- The framework applies to permanent connections in circuit-switched environment, rather than to handling individual packets in packet-switched networks.
- The theoretical model assumes that the senders have instantaneous knowledge of the current costs of all the links at the instant they need to make a decision. It is also assumed that fine-grained clocks are available and essentially zero-delay responses to events are possible.
- The natural application of the framework, as in the case of managing permanent virtual circuits, is to use bandwidth as the basic resource being rationed. However, bandwidth is *not* under our control because competing flows may ocupy at any point an arbitrary and time-varying fraction of the link bandwidth. Therefore, while bandwidth is an essential component for performance, our protocols cannot meaningfully ration (save or waste) it, as its availability is primarily at the mercy of other applications. (Recall that our application has to share the link bandwidth "fairly" with other TCP flows.)

In fact, the underlying model does not specify which resources are to be managed by it. The most important issue is understanding what is the resource that is being controlled (rationed) since not all of the resources used are controllable. Figuratively speaking, available bandwidth to flow control is like wind to sailing: it is controlled by adversarial forces rather than by us. We must try to adapt as much as possible to changing circumstances, rationing the controlable resources.

#### B.1 Practical cost function

We chose buffer space in each overlay network router as the scarce resource we want to control. Conceptually, we model our software overlay network router as a router with fixed size output link buffers where packets are placed into the appropriate output link queues as soon as the packet is received by the router. Note that the number of queues is equal to the number of outgoing links, and does not depend on the number of senders, receivers or groups in the system. If a link is not congested its corresponding queue will be empty. In this case, once a message arrives in the buffer it is immediately sent, maybe with only a short delay due to send scheduling. If the incoming traffic is more than the capacity of an outgoing link, some packets will accumulate in the corresponding outgoing link buffer.

Each router establishes the cost for each of its outgoing links and advertises this cost to all the other daemons using the overlay network. The price for a link is zero if its corresponding buffer is empty. This means that the cost is zero as long as the link is not congested, i.e. the link can accommodate all the incoming traffic. As the link gets congested and messages accumulate in the buffer, the cost of the link increases. The price can theoretically go as high as infinite when the buffer is full. In practice, the cost of the link will increase until a given value $S$ when nobody will be able to buy it.

Equation 1 from Section IV-A gives the basic form of our cost function. The utilization of a link is given by $x/M$ where $x$ is the average number of messages in the buffer, $M$ is the desired capacity of the buffer. In our implementation the minimum benefit is $\alpha = 1$. So the cost is:

$$C(x) = \alpha \cdot \gamma^{x/M} = \beta^{x/M}$$

This function ranges from 1 to $\beta$. We want to scale it to range from 0 to $S$ (the prohibitive cost). So the cost becomes:

$$C(x) = S \cdot \frac{\beta^{x/M} - 1}{\beta - 1}$$

The theory claims that the maximum benefit given to a client should be sufficient to fully utilize the network. If this maximum benefit is $Max_B$, and the minimum benefit is 1, then the base of the exponent in the cost function should be $\frac{Max_B}{1}$.

If we use $Max_B$ (say 20) as the base of the exponent, then the cost function stays near zero until the buffer utilization is almost 1. Then, the cost goes up very quickly. Since we don't have instantaneous feedback, this allows actual congestion to occur (used buffers $> 0$) without increasing the cost almost at all. Therefore a practical mechanism will provide incremental feedback on costs before the utilization becomes high. This calls for a small base of the exponent.

Using a small base for the exponent has real-world advantages and still fits the model. First, it provides an increase in costs almost immediately as the link first becomes congested. Second, it is still exponential, which is required by the theoretical model. We chose $e$ as the base of the exponent.

So finally we get:

$$C_l(x) = S \cdot \frac{e^{x/M} - 1}{e - 1}$$

It was interesting to see whether other functions that also react quickly could work. We tried functions such as $f(x) = x$ and $f(x) = x^2$, however, they were unstable and unable to provide consistent throughput.

Each router will periodically advertise the cost of its links by multicastig a cost update message to all the other daemons through the overlay network. We will discuss later in Section V how we minimize the control traffic while we maximize the information it contains.

From the cost of the multicast tree links a sending daemon can compute the cost for a packet. That cost is the sum of the costs on all the links that the packet will traverse plus a constant $\rho$ that will be discussed later.

$$MT = \{l | l \in \text{Multicast tree of p}\}$$

$$C_p = \rho + \sum_{l \in MT} C_l \qquad (2)$$

### B.2 Benefit assignment

The choice of benefit is tightly intertwined with the goal we want to achieve. If our goal is to maximize the sending throughput of the network, then the benefit function should reward each packet sent onto the network with a fixed benefit. If our goal is to maximize sending throughput while still allowing every sender to send at least some traffic (i.e. no one starves), then the benefit function could be a concave function (like $1/x$) of the number of packets already sent by this connection. The benefit function could also be based on the number of received packets, instead of sent packets. That would advantage senders who send to larger groups. In this paper we chose to maximize sending throughput so our benefit is one unit per packet sent.

Although one would like to handle the benefit as a pure rate, in practice giving several units of benefit to a client at a time is more efficient. We define a "salary" as the amount of dollars a client is given from time to time. A client is allowed to save up to $S$ dollars of its salary. This mechanism actually works like a token bucket of dollars.

The sending rate of the clients is regulated by the daemon a client is connected to. The daemon acts as the client's "agent", purchasing packets whenever possible for the sending client.

If the client wants to send a message that costs more dollars than it currently has in his budget, the daemon blocks the client by not reading from its socket anymore, and keeps the expensive message until the client affords to buy it. This hapens when the client receives more dollars, or when the cost for the sending tree goes down.

### B.3 Other considerations

A link that is not congested has a cost zero, and a client that sends through it will see it this way until the next cost update arrives. Therefore, any client would be allowed to send an infinite number of messages on a non congested link between two cost updates, obviously leading to high burstiness and congestion on these links. A solution for this problem is to have a minimum cost per link greater then zero, however this will not scale with the size of the network (long paths could have a very large cost even if idle). Our solution is to keep the minimum cost at zero, but add a constant cost per message (like a processing fee). This cost is the constant $\rho$ referred to in Equation 2, and in our implementation we define it to be \$1. Therefore we put a cap on the number of messages each client can send between two cost updates, even when the network cost is zero, because of the limited salary.

Since we do not know the network capacity (assumed known by the theory), we approximate such knowledge by adaptively probing for the current network bandwidth. We chose to do this in a way very similar to TCP. When a client receives a salary, the daemon checks whether the client was blocked during the last salary period (due to an intention to buy a message more expensive that it could afford). If the client was not blocked, it means that he was able to buy, and therefore send, all the messages he initiated, so the next salary period will be exactly the same as the previous one. However, if the client was blocked, the daemon compares the cost of the most expensive message that the client bought during the last salary period with a certain threshold $T$. If the maximum cost is less than the threshold, the time between two salaries is decreased, as the client might have been blocked due to processing fees on a relatively idle network. The new salary period is:

$$T_{new} = \frac{T_{old} \cdot T_{update}}{T_{old} + T_{update}} \quad (3)$$

where $T_{old}$ is the previous salary period and $T_{update}$ is the minimum time between two cost updates. If the maximum cost is larger then the threshold, it means that the client tries to buy expensive messages, therefore contributing to

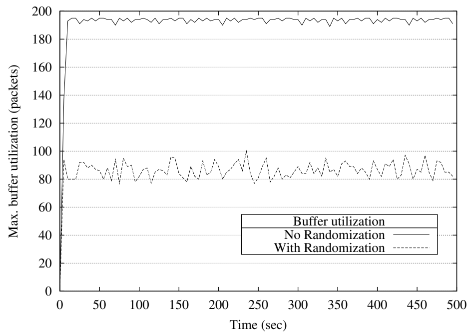


Fig. 2. Randomization effect on buffer utilization

congestion in the network. The new salary period will be:

$$T_{new} = 2 \cdot T_{old} \quad (4)$$

This algorithm resembles the TCP congestion control [7] where $T_{new}$ and $T_{old}$ would be the average time between two packets sent, and $T_{update}$ would be the round trip time. Equation 3 is algebraically equivalent to adding 1 to the congestion window in TCP, while equation 4 is equivalent to reducing the congestion window by half.

Finally, the coarse granularity of cost updates causes a high degree of synchronization between clients at the time an update is received. This synchronization phenomenon could cause oscillations in the router buffers as every client buys at the same time, then the cost goes up, then everybody waits for the price to go down, etc. To prevent the synchronization, each client may *not* send a message even though it has sufficient funds to do so. It will choose to send only with a probability $1/C_p$, where $C_p$ is the total cost of the message. This bad behavior and the benefit of randomization are shown in Figure 2, which represents the maximum buffer utilization for a 2Mb link when 100 clients compete on sending through it.

## V. FAIRNESS AND SCALABILITY

What definition of fairness is best in a multicast environment is an area of active research. For this work we chose a conservative approach of considering each link on our overlay network as one TCP flow. So, we will fairly share each link with all the external competing traffic. Some might argue that this is too conservative, as many people may be using our multicast service at once, and each one would receive their own TCP flow if they were using a separate unicast service, but here they will share only one TCP flow. This is true. However, the purpose of this paper is not to argue for a particular model of fairness, but rather to provide an overlay network flow control that works in any environment.

The difference between looking at the receiving throughput and looking at the sending throughput when comparing a multicast protocol with TCP is big, as there can be more than one receiver for one sender. However, we try to be very conservative by taking into account the worst case scenario and analyze only the sending throughput.

Giving a “fair” amount of traffic to all the senders, regardless of their intended use of network resources, is at odds with maximizing throughput of the network as a whole. We choose, by default, to provide a fair share of our overlay network resources to all senders who have the same cost per messsage. That could be because their messages travel over the same multicast tree, or just by random coincidence. However, senders who have higher costs, say because they cross more congested links, will be allowed to send at a lower rate. This is depicted in Section VI Scenario 3, where sender A-F who uses all the network links receives much less then its “fair” share of the resources.

We provide a per-group, per-sender, per-message flow control in a multi-group multi-sender multicast environment, without keeping any per-flow state in the intermediate routers. Morever, the amount of control traffic does not depend of the number of groups, senders, or receivers in the system, neither it carries any information about them. The cost updates carry information only about the state (buffer sizes) of the links - edges in the overlay network graph.

We consider the edges of the overlay network to be wide-area, high latency virtual links, in the order of milliseconds to hundreds of milliseconds delay. Therefore, the diameter of the overlay network can not be more than some tens of hops - it is limited by the distances achievable on our planet - so a maximum number of daemons in the order of hundreds is realistic. The number of clients or groups, however, can be much higher, in the order of thousands, or tens of thousands.

Each daemon in the overlay network multicasts a cost update at every $T_{max}$ interval as long as its outgoing links are not congested, or their costs did not change significantly. However, if at least one of its links becomes congested - the link cost increases - the daemon will send cost updates faster, at $T_{min}$ intervals. This mechanism is based on the observation that, in general, in one multicast tree there are only a few bottleneck links that will impose the speed of the entire tree. Moreover, it is likely that the bottleneck links for different senders or groups will be the same. Therefore, only the daemons that control bottleneck links will send frequent cost updates, while the others will not contribute much to the control traffic. Since the cost updates are very small (64 bytes), they are piggy-backed with the data packets whenever possible. In our implementation we chose $T_{max}$ to be 2.5 seconds, and $T_{min}$ 50 milliseconds. For an overlay network with the average link

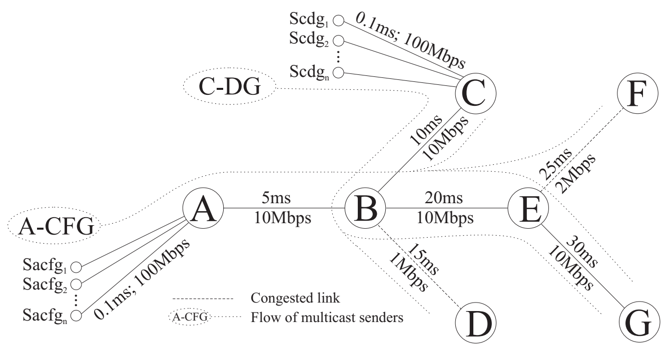


Fig. 3. Scenario 1: Network Configuration

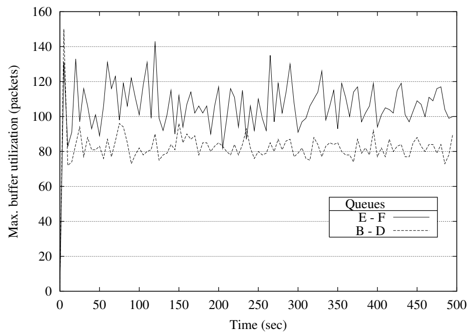


Fig. 4. Scenario 1: Buffers

bandwidth of 2Mbps this leads to a control traffic of about 0.5 percent per congested link, and 0.01 percent per non-congested link.

## VI. Simulation results

We used the ns2 network simulator[3] to evaluate the performance and behavior of our flow control protocol. The main issues we focused on are:

- optimal network resource utilization;
- automatic adjustment for dynamic link capacities;
- optimal sharing of network resources to achieve maximum throughput;
- fairness between flows using the same congested links;
- scalability with number of clients, groups and diameter of the network;

**Scenario 1 – one bottleneck link per flow:**

We used the multicast tree shown in Figure 3, with the link capacities and latencies as shown in the figure. All the intermediate buffers in the network have a soft limit of 100 packets. Clients receive a \$10 salary, and they can save up to \$20 in their budget. The processing fee is \$1/packet.

Two classes of 20 separate clients each initiate multicast messages, $S_{acfg}$ and $S_{cdg}$. Receiver clients are connected to nodes C, D, F and G. For simplicity we do not show the receiving clients, but only the daemons they are connected to. The $S_{acfg}$ clients multicast to receivers connected to

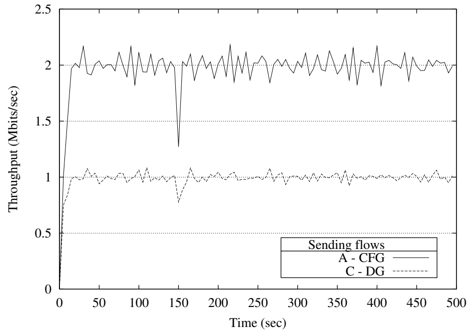


Fig. 5. Scenario 1: Throughput

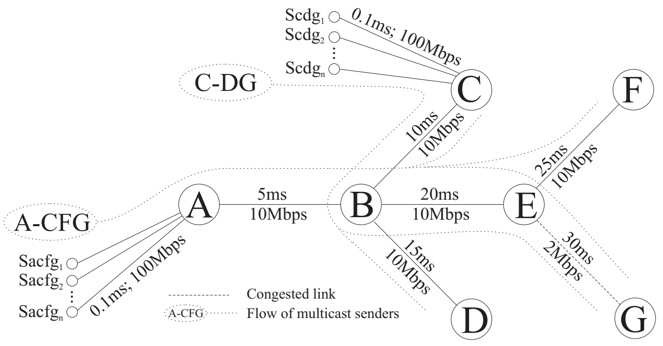


Fig. 6. Scenario 2: Network Configuration

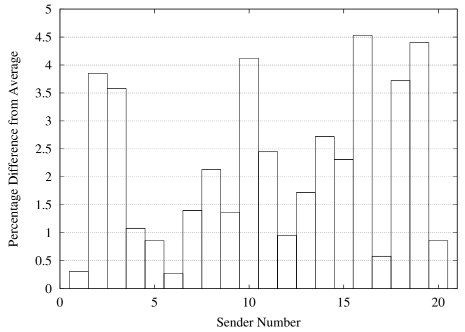


Fig. 7. Scenario 2: Fairness

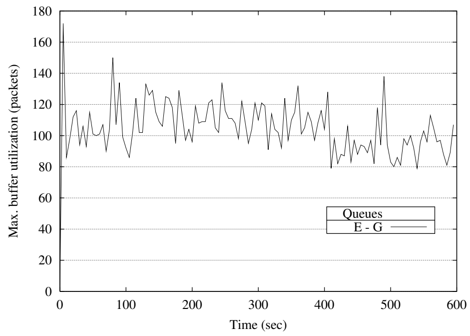


Fig. 8. Scenario 2: Delayed senders, buffers

nodes C, F and G, and and the $S_{cdg}$ clients multicast to receivers connected to nodes D and G, sharing the links B-C, B-E and E-G. $S_{acfg}$ clients are limited by the 2Mb bottleneck link $E-F$, and $S_{cdg}$ clients are limitted by the 1Mb link $B-D$. There are no other bottleneck links in the system.

Rather than looking at the instantaneous buffer occupancy which is very dynamic and depends on the sampling frequency, we chose to analyze the evolution of the maximum buffer utilization over the last sampling period in Figure 4.

The reason for a higher buffer utilization on link $E-F$ is that there is a higher network response delay for the $A-CFG$ clients. This also explains the increased variation in sending throughput for these clients, in Figure 5. However, the aggregate sending throughput is optimally divided between senders for maximal network usage, $S_{acfg}$ clients getting on average 1.977 Mbps and $S_{cdg}$ getting 0.992 Mbps.

**Scenario 2 – one shared bottleneck link:** To examine the effect of a congested link, the network shown in Figure 6 is used. Here, link $E-G$ forms the bottleneck for both flows. Each flow represents 20 sending clients.

The two flows share the bottleneck link fairly equal. Flow $S_{acfg}$ gets an average of 997.3 Kbps and flow $S_{cdg}$ gets an average of 997.6 Kbps, while the buffer of the link $G-E$ stays below 150 packets. The various clients who make up each flow also share the bandwidth fairly. In Figure 7, we show the percentage difference of each of the 20 $S_{acfg}$ clients throughput from the average throughput achieved by the group as a whole. This shows that the variance of the clients throughput is less then 4.6%.

A second experiment used the same tree configuration as Figure 6 but started the second group of senders $S_{acfg}$ after 200 seconds, and also changed the bandwidth of the link E-G to 1Mbps after 400 seconds. Figure 8 shows the maximum buffer utilization on the links E-G, B-D and E-F. After 200 seconds, as well as after 400 seconds we do not see any major change in the buffer utilization on the bottleneck link. Specifically, there is no large spike in maximum utilization when the second group of clients begins sending all at once, or when the bottleneck link reduces its capacity by half. This is because the link has an existing non-zero cost and so the clients must pay that cost before sending. Figure 9 shows how the throughput of the two groups of clients responds very quickly to the new load, or change in the available bandwidth.

**Scenario 3 – chain network:**

Our flow control tries to maximize throughput by allowing low cost messages to pass, and reducing high cost traffic. The very simple way to show this is to set up a chain network in which some clients try to send their messages

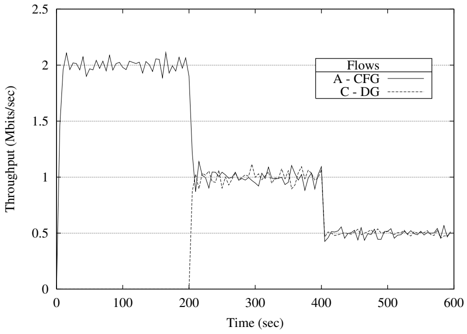


Fig. 9. Scenario 2: Delayed senders, throughput

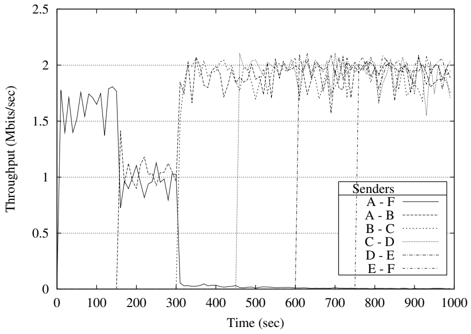


Fig. 11. Scenario 3: Throughput

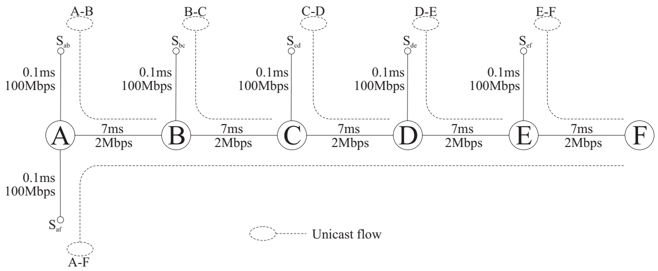


Fig. 10. Scenario 3: Network Configuration

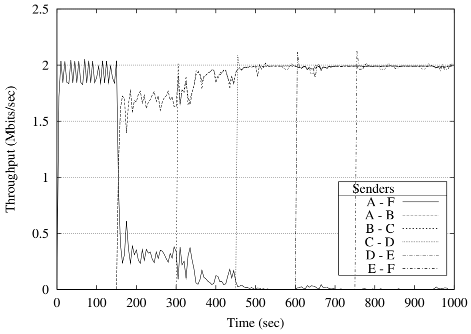


Fig. 12. Scenario 3: TCP throughput

across the entire network, while other clients use only one link in the chain. Figure 10 shows such a network with 5 links connected in series. One client sends from node A to node F, and 5 other clients send only over one link, i.e. from B to C or from E to F.

Figure 11 shows the throughput on the chain network as short connections start up every 150 seconds. The client A-F starts trying to use the entire capacity of the network. When the client A-B starts, they share the congested link, AB, about equally. When the third client, B-C, starts at time 300, the long flow A-F slows down letting short flows use the available bandwidth. As we add more congested links by starting more short connections, the throughput of the flow A-F goes almost to zero, thus almost maximizing the global throughput of the system. If the flow control had been fair, the aggregrate throughput would be 6 Mbps, 1 Mbps for each client. We achieved an aggregate throughput after all clients have started of 9.677 Mbps, while the theoretical maximum is 10 Mbps.

The results of the previous simulation show a definite bias towards short flows and show how such a bias can increase network throughput. Actually, this is the same behavior as end to end connections using TCP would show given the same situation. Figure 12 shows the throughput on the same chain network, only instead of hop-by-hop connections regulated by our flow control, we run *end-to-end* TCP connections. With end-to-end TCPs, the long A-F connection is biased against in the same way as our flow control. Moreover, when competing with only one other TCP flow A-B, the longer flow A-F receives less bandwidth. We believe this is because TCP is biased against both long RTT connections as well as having to cross multiple congested links. So even when only one link is congested, the longer RTT of the A-F flow causes it to receive lower average bandwidth then the short RTT A-B flow.

**Scenario 4 – scalability with number of nodes and groups:** In order to see how a large number of clients multicasting to many different groups share a network resource we set up the network shown in Figure 13. The overlay network consists of 1602 nodes, and there are 1600 clients, each of them connected to a separate daemon, joining 800 differnt groups. We could not run a bigger scenario due to memory limitation using the ns simulator on our machines.

Each of the clients $S_1$ to $S_{800}$ multicasts to three different receivers. $S_1$ sends to $R_1$, $R_2$ and $R_3$, $S_2$ sends to $R_2$, $R_3$ and $R_4$, and so on, until $S_{800}$ that sends to $R_{800}$, $R_1$ and $R_2$. All the senders share the same bottleneck link, A-B.

We ran the simulation with different number of senders, from 5 to 800. As shown in Figure 14 the maximum buffer utilization on the bottleneck link A-B stays about the same until the number of senders reaches to the buffer soft limit (in our case, 100), and then it starts increasing. However, the Cost-Benefit framework kept the buffer size under controllable limits (under 170 packets for 800 senders). The

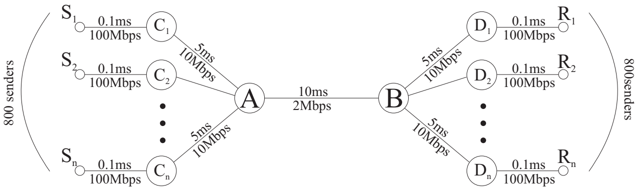


Fig. 13. Scenario 4: Network Configuration

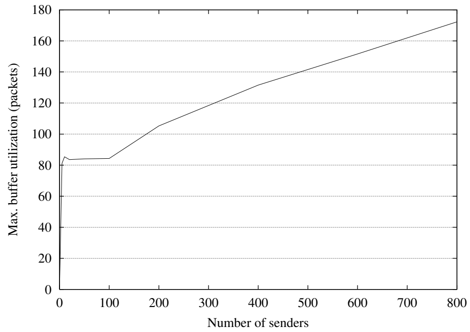


Fig. 14. Scenario 4: Buffers

aggregate throughput was not affected by the number of senders, getting an average of 1.979Mbps for the aggregate sending rate of 800 senders.

## VII. Simulation validation with real-life experiments

We ran our experiments over a portion of the CAIRN network[5]. This is a wide-area network that crosses the entire United States, and consists of links that range from 1.5Mbps to 100 Mbps. The routers are Intel machines that run FreeBSD 3.4. Figure 15 shows the portion of the CAIRN network that we used for our experiments. We measured individual link latencies using ping under zero traffic, and the available bandwidth with simple point to point TCP connections between each two ends of a link. Note that our flow control uses the available bandwidth given by the underlying TCP link protocol, and not the physical bandwidth of the network.

We extended the Spread toolkit[4] to use our Cost-Benefit framework for global flow control. Spread has its own overhead of about 15% of data sent due to headers required for routing, ordering, and safety guarantees as well as to provide user-friendly group names. What we measured in our results is *actual user data sent and received* by clients connected to Spread. For these experiments we gave each client a \$10 salary, and allowed up to \$20 of savings. The processing fee was \$1. All the overlay network links had a soft buffer limit of 100 packets.

Sender $S_1$ multicasts messages to a group A joined by the receivers $R_A$, while senders $S_2$ and $S_3$ multicast to a

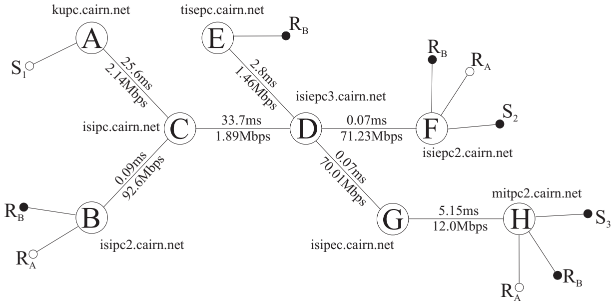


Fig. 15. CAIRN: Network Configuration

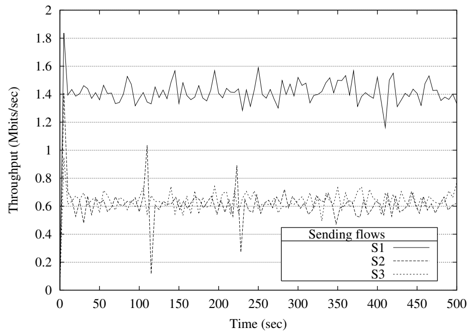


Fig. 16. CAIRN: Real-life results

group B joined by the receivers $R_B$. All the clients run directly on the overlay network machines, connected to the daemons through Unix Domain Sockets. Obviously, $S_1$ was limited by the bottleneck link C-D, while $S_2$ and $S_3$ had to share the bottleneck ling D-E. Taking into account the data overhead in Spread, we can see in Figure 16 that the sender clients use the network resources optimally and share them fairly between senders $S_2$ and $S_3$, $S_1$ getting 1.417 Mbps, while $S_2$ and $S_3$ got 0.618 and 0.640 Mbps. We then created a simulation of the CAIRN network, including the same bandwidth, latency and network characteristics. We ran the same experiment described above on the simulated network. After adjusting the average sending throughput with the Spread overhead, the simulation got 1.495Mbps for $S_1$, 0.626Mbps for $S_2$ and 0.630Mbps for $S_3$ in the simulation, which differs from the actual CAIRN results by less than 5.5% for $S_1$, and less than 1.6% for $S_1$ and $S_2$.

## VIII. Conclusions

This paper presented a new global flow control approach for wide-area overlay networks based on sound theoretical foundations. Our Cost-Benefit framework provides a simple and flexible way to optimize flow control to achieve several desirable properties such as near optimal network throughput and automatic adjustment to dynamic link capacities. The resulting algorithm provides fairness between equal cost internal flows and is fair with outside traffic, such as TCP. We implemented the framework in the ns2 simulator and showed results similar to those predicted by theory. We then implemented the framework in the Spread group communication system and conducted live experiments on the CAIRN network to validate the simulations and show the real-world performance of the framework.

## REFERENCES

[1] Baruch Awerbuch, Yossi Azar, and S. Plotkin, "Throughput-competitive on-line routing," in *Proceedings of 34th IEEE Symposium on Foundations of Computer Science*, 1993.

[2] Yair Amir, Baruch Awerbuch, Amnon Barak, Ryan Borgstrom, and Arie Keren, "An opportunity cost approach for job assignment and reassignment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 760–768, July 2000.

[3] "ns2 network simulator," Available at http://www.isi.edu/nsnam/ns/.

[4] "Spread group communication system," http://www.spread.org/.

[5] "Cairn network," Information available at http://www.cairn.net/, 2001.

[6] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.

[7] Van Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, 1988.

[8] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Proceedings of ACM SIGCOMM*, 1988.

[9] Sally Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, October 1994.

[10] K. K. Ramakrishnan and Sally Floyd, "A proposal to add explicit congestion notification (ECN) to IP," RFC 2481, January 1999.

[11] R. G. Gallager and S.Jamaloddin. Golestani, "Flow control and routing algorithms for data networks," in *Proceedings of 5th International Conference on Computers and Communication*, 1980, pp. 779–784.

[12] R. J. Gibbens and Frank P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, vol. 35, December 1999.

[13] S. Jamaloddin Golestani and S. Bhatacharyya, "End-to-end congestion control for the Internet: A global optimization framework," in *Proceedings of International Conference on Network Protocols*, October 1998, pp. 137–150.

[14] Frank P. Kelly, A. K. Maulloo, and David. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[15] David Lapsley and Steven Low, "An IP implemention of optimization flow control," in *Proceedings of IEEE Globecom*, 1998, pp. 3023–3028.

[16] David E. Lapsley and Steven Low, "Random early marking for Internet congestion control," in *Proceedings of IEEE Globecom*, 1999, vol. 3, pp. 1747–1752.

[17] Idit Keidar, Jeremy Sussman, Keith Marzullo, and Danny Dolev, "A client-server oriented algorithm for virtually synchronous group membership in wans," in *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan, April 2000, pp. 356–365, IEEE Computer Society Press, Los Alamitos, CA.

[18] Idit Keidar and Roger Khazan, "A client-server approach to virtually synchronous group multicast: Specifications and algorithms," in *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan, April 2000, pp. 344–355, IEEE Computer Society Press, Los Alamitos, CA.

[19] D.A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia, "The totem multiple-ring ordering and topology maintenance protocol," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 93–132, May 1998.

[20] Yair Amir, Claudiu Danilov, and Jonathan Stanton, "A low latency, loss tolerant architecture and protocol for wide area group communication," in *Proceeding of International Conference on Dependable Systems and Networks*. June 2000, pp. 327–336, IEEE Computer Society Press, Los Alamitos, CA, FTCS 30.

[21] Takako M. Hickey and Robbert van Renesse, "Incorporating system resource information into flow control," Tech. Rep. TR 95-1489, Department of Computer Science, Cornell University, Ithaca, NY, 1995.

[22] Shivakant Mishra and Lei Wu, "An evaluation of flow control in group communication," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 571–587, October 1998.

[23] Dan Rubenstein, Jim Kurose, and Don Towsley, "The impact of multicast layering on network fairness," in *Proceedings of ACM SIGCOMM*, October 1999, vol. 29 of *Computer Communication Review*, pp. 27–38.

[24] Thomas Bonald and Laurent Massoulie, "Impact of fairness on Internet performance," Submitted, December 2000.

[25] Robert C. Chalmers and Kevin C. Almeroth, "Developing a multicast metric," in *Proceedings of GLOBECOM 2000*, 2000, vol. 1, pp. 382–386.

[26] Huayan Amy Wang and Mischa Schwartz, "Achieving bounded fairness for multicast and TCP traffic in the Internet," in *Proceedings of ACM SIGCOMM*, 1998.

[27] Todd Montgomery, "A loss tolerant rate controller for reliable multicast," Tech. Rep. NASA-IVV-97-011, West Virginia University, August 1997.

[28] Tetsuo Sano, Nagatsugu Yamanouchi, Teruji Shiroshita, and Osamu Takahashi, "Flow and congestion control for bulk reliable multicast protocols – toward coexistance with TCP," in *Proceedings of INFOCOM*, March 1998.

[29] Shuming Chang, H. Jonathan Chao, and Xiaolei Guo, "Tcp-friendly window congestion control with dynamic grouping for reliable multicast," in *Proceedings of GLOBECOM 2000*, 2000, vol. 1, pp. 538–547.

[30] Ljubica Blazevic and Jean-Yves Le Boudec, "Distributed core multicast (DCM): a multicast routing protocol for many groups with few receivers," *Computer Communications Review*, vol. 29, no. 5, pp. 6–21, October 1999.

[31] Yair Amir and Jonathan Stanton, "The spread wide area group communication system," Tech. Rep. 98-4, Johns Hopkins University Department of Computer Science, 1998.

[32] L. E. Moser, Yair Amir, P. M. Melliar-Smith, and D. A. Agarwal, "Extended virtual synchrony," in *Proceedings of the IEEE 14th International Conference on Distributed Computing Systems*. June 1994, pp. 56–65, IEEE Computer Society Press, Los Alamitos, CA.

[33] R. Vitenberg, I. Keidar, G. V. Chockler, and D. Dolev, "Group communication specifications: A comprehensive study," Tech. Rep. CS99-31, Institute of Computer Science, The Hebrew University of Jerusalem, 1999.