The One-to-Many TCP Overlay: A Scalable and Reliable Multicast Architecture

François Baccelli* Augustin Chaintreau* Zhen Liu† Anton Riabov†

Abstract— **We consider reliable multicast in overlay networks where nodes have finite-size buffers and are subject to failures. We address issues of end-to-end reliability and throughput scalability in this framework. We propose a simple architecture which consists of using distinct point-to-point TCP connections between adjacent pairs of end-systems, together with a back-pressure control mechanism regulating the transfers of adjacent TCP connections, as well as a back-up buffering system handling node failures. This architecture, that we call *the One-to-Many TCP Overlay*, is a natural extension of TCP to the one-to-many case, in that it adapts the rate of the group communication to local congestion in a decentralized way via the window back-pressure mechanism. Using theoretical investigations, experimentations in the Internet, and large network simulations, we show that this architecture provides end-to-end reliability and can tolerate multiple simultaneous node failures, provided the backup buffers are sized appropriately. We also show that under random perturbations caused by cross traffic described in the paper, the throughput of this reliable group communication is always larger than a positive constant, that does not depend on the group size. This scalability result contrasts with known results about the non-scalability of IP-supported multicast for reliable group communication.**

I. Introduction

With the proliferation of broadband Internet access, end-system multicast (also referred to as application-level multicast) becomes a practically feasible and appealing alternative to IP-supported multicast which has been experiencing deployment obstacles.

In the end-system multicast architecture, one forms an overlay tree by establishing directed point-to-point connections between end-systems, where each end-system duplicates and forwards data to downstream end-systems in a store-and-forward way. Note that the nodes of the multicast tree are end-systems rather than routers as in IP-supported multicast. Since this architecture may be less efficient than IP-supported multicast in terms of resource utilization, some protocols have been proposed for the construction of efficient overlay trees [1]–[4].

* INRIA & Ecole Normale Supérieure, 45 rue d'Ulm 75005, Paris, France, francois.baccelli@ens.fr, augustin.chaintreau@ens.fr

† IBM T.J. Watson Research Center, P.O. Box 704 Yorktown Heights, NY 10598, USA, zhenl@us.ibm.com, riabov@us.ibm.com

A natural way of improving the reliability and the TCP friendliness of end-system multicast is to use point-to-point TCP connections between nodes as proposed in RMX [5], Overcast [6] and ROMA [7]. There is an increasing interest in using TCP based multicast overlays for real time applications (see studies on multimedia streaming over TCP [8], [9]). TCP is indeed an appealing alternative of UDP for such applications due to a number of advantages such as fair bandwidth sharing, in-order delivery and passing through client imposed firewalls. TCP is also the best candidate for point to point connections for applications which require reliable data delivery as it provides local recovery from packet losses in network links.

An important remaining issue with such overlay networks is their *end-to-end reliability*. In case of a node failure, the nodes in the subtree rooted at the failed node need, on one hand, to be re-attached to the remaining tree and, on the other hand, to get TCP sessions established from where they are stopped. The former is referred to as the *resiliency* issue in the literature (see e.g. [10]), which, in this context, consists in the detection of failures and in the reconstruction of trees. When using TCP point-to-point connections, it is not clear how known resiliency mechanisms can be extended to achieve reliability for large groups. While it is relatively easy to find nodes of re-attachment and thus to reconstruct the tree, it is not guaranteed that the data flows can be restarted from where they are stopped. As the forwarding buffers of the nodes in the overlay network have finite size, it may happen that the packets needed by the newly established TCP sessions are no longer in the forwarding buffers. Another important reliability issue stems from the fact that even when nodes never fail, some packets may be lost in a node due to buffer overflow.

In this paper, we consider reliable multicast in overlay networks where nodes have finite-size buffers and can fail. We propose a simple architecture which consists of using distinct point-to-point TCP connections between adjacent pairs of end-systems, together with a back-pressure control mechanism (as in [11], [12]) regulating the transfers of adjacent TCP connections, as well as a back-up buffering system handling node failures. This architecture, that we call *the One-to-Many TCP Overlay*, is a natural extension of TCP to the one-to-many case, in that it adapts the rate of the group communication to local congestion in a decentralized way via the window back-pressure mechanism. Using theoretical investigations and experimentations in the Internet, we show that this architecture provides end-to-end reliability and can tolerate multiple simultaneous node failures, provided the backup buffers are sized appropriately.

The second issue that arises in reliable multicasting pertains to performance when the group size is large. Significant effort has been spent to evaluate the performance of IP-supported reliable multicast transport protocols [13]–[16]. In particular, it was shown in [15] and [16] that for IP-supported reliable multicast schemes using a TCP like control, the group throughput decreases and tends to zero when the group size increases.

Within the context of end-system multicast, this issue was studied in [17] for the case of infinite-size buffers in end-systems. It was shown that such a end-system multicast scales in the sense that the group throughput is lower bounded by a positive constant independent of the group size. In [7], the scalability issue is considered for a similar case where the downstream TCP connections are assumed not to affect the upstream TCP connections. The case of finite-size buffers was first studied in [12], where the authors investigated a TCP-friendly congestion control mechanism with fixed window-size for node-to-node transfers. Simulation results were presented showing a sharp decrease of throughput with the size of the group for moderate group sizes. One of the conclusions drawn from this observation is that the input/output buffer size plays a key role and should perhaps be increased to infinity with the size of the multicast group.

We prove that for finite node buffers, the group throughput of the one-to-many TCP overlay is also bounded from below by a strictly positive constant which does not depend on the group size. The assumptions, which are described in Section III-F, consist of statistical guarantees on each point to point route and of the boundedness of the fan-out degree of the multicast tree. This contrasts with the known result established about the non-scalability of IP-supported multicast based on TCP-like control. It also shows that the unbounded increase of the node buffer size which is suggested in [12] in order to cope with larger and larger groups is not necessary to guarantee a group throughput.

It is worthwhile noticing that in this One-to-Many TCP architecture, nodes influence each other both downstream (when the slow down of a node slows downs its offspring nodes through the delay incurred by forwarded packets) and upstream (via the back-pressure mechanism). This contrasts with the case without backpressure where nodes influence downstream nodes only. A consequence of this acyclicity is that the group throughput is here a function of the parameters of the *whole* tree (topology, buffer sizes, random packet ECN marking or random packet losses and random effects of cross traffic). Another new modeling difficulty comes from the fact that packets need to be relayed in sequence by a node.

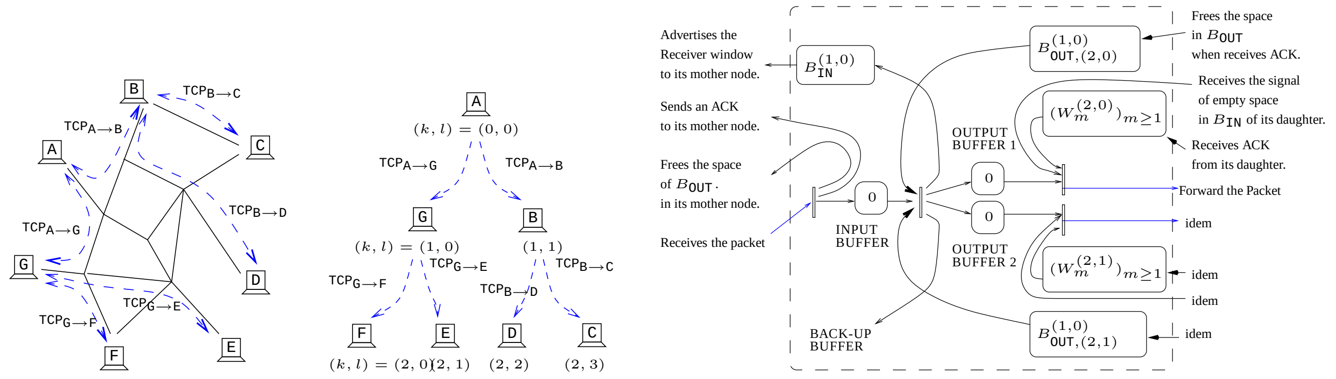
In order to address these modeling and performance evaluation issues, we introduce a new class of random graphs with random weight in vertices and a random set of edges and we show that the group throughput can be interpreted in terms of first passage percolation in such a random graph. The main practical conclusion concerning performance is based on experiments and simulation. It bears on the actual value of the group throughput that is achieved inside a group of large size. We compare this to some reference throughput, defined as the worst long term average rate which can be separately achieved by TCP on a route between two end-systems in a standalone mode. Compared to this reference, and for a wide range of parameters, the degradation of the group throughput induced by back-pressure is less than 20%.

The paper is organized as follows. The next section describes the multicast overlay architecture. The throughput scalability results are presented in §III. Section IV describes the algorithms ensuring overall reliability and gives a proof of their end-to-end reliability properties. Simulation results and Planet-Lab experiments aiming at showing the joint scalability and reliability properties of this kind of architecture are gathered in §V.

II. The One-to-many TCP Overlay

A. Multicast Overlays

In contrast to native reliable IP multicast where the nodes of the tree are Internet routers and where specific routing and control mechanisms are needed, overlay multicast uses a tree where the nodes are end-systems and where the currently available point to point connections between end-systems are the only requirement. An arc in the overlay network represents the path between the two nodes that it connects. While this path may traverse several routers in the physical network (see Figure 1 (left and center), and the models introduced in §III), this level of abstraction considers the path as a direct link in the overlay network. The point-to-point communication between a mother node and a daughter node is carried out by TCP. We shall assume that in all TCP connections, Fast Retransmit Fast Recovery [18] is implemented. We consider both situations with and without Selective Acknowledgment (SACK). In the scalability analysis of Section III, we also consider the case of Early Congestion Notification (ECN) as an intermediate step.



A
B
C
D
E
F
G
TCP_{A→B}
TCP_{B→C}
TCP_{B→D}
TCP_{A→G}
TCP_{G→F}
TCP_{G→E}
(k, l) = (0, 0)
(k, l) = (1, 0)
(1, 1)
(k, l) = (2, 0)
(2, 1)
(2, 2)
(2, 3)
Advertises the Receiver window to its mother node.
Sends an ACK to its mother node.
Frees the space of B_OUT in its mother node.
Receives the packet
INPUT BUFFER
BACK-UP BUFFER
OUTPUT BUFFER 1
OUTPUT BUFFER 2
Frees the space in B_OUT when receives ACK.
Receives the signal of empty space in B_IN of its daughter.
Receives ACK from its daughter.
Forward the Packet
idem

Fig. 1. An overlay multicast tree: physical topology (left) and abstract topology (center), description of node with index $(1, 0)$ (right).

On each node (except for the root node), there is an input buffer, corresponding to the receiver window of the upstream TCP, and, except for the leaf nodes, there are several output buffers, also referred to as forwarding buffers, one for each downstream TCP connections. There is also a backup buffer in each of these intermediate nodes storing copies of packets which are copied out from the input buffer (receiver window) to the output (forwarding) buffers. These backup buffers are used when TCP connections are re-established for the daughter nodes after their mother node fails. Figure 1 (right) illustrates these buffering mechanisms. Throughout this paper we shall assume that all these buffers have finite sizes $B_{\mathtt{IN}}$, $B_{\mathtt{OUT}}$, $B_{\mathtt{BACK}}$ (for, respectively, input buffer, output buffer and backup buffer).

In this paper, we do not consider the tree construction/optimization issue. Rather, we assume that the tree topology is given, and that the out-degrees (or fan-out) are bounded by a constant D.

We consider several tree topologies, for which we introduce the following generic notation: we number end-systems by a pair (k, l). The first index k gives their distance to the root of the tree (or *level*). The second index l allows one to number end-systems with the same level. For the case of a complete binary tree, the end-systems with the same distance k from the root are labeled by numbers $l = 0, \ldots, 2^k - 1$. An example of complete binary tree with height equal to 2 is described on Figure 1 (middle). The mother node of end-system (k, l) is denoted $(k - 1, \mathsf{m}(k, l))$. The daughter nodes of end-system (k, l) is labeled $(k + 1, l')$ with $l' \in \mathsf{d}(k, l)$. For a complete binary tree, $\mathsf{m}(k, l) = \lfloor \frac{l}{2} \rfloor$ and $\mathsf{d}(k, l)$ is $\{2l, 2l + 1\}$. For end-system (k, l) we denote by $B^{(k,l)}_{\mathtt{IN}}$ the size (measured in packets; we assume that all packets have the same size for the sake of simple exposition, although this is not an essential assumption) of its input buffer. We denote by $B^{(k,l)}_{\mathtt{OUT},(k',l')}$ the size (also measured in packets) of the output buffer of end-system (k, l) in the socket corresponding to the TCP connection to the daughter end-system (k', l').

B. Reliable Transfer and Forwarding via Back-Pressure

There can be three different types of packet losses in the overlay multicast: (1) losses that occur in the path in-between the nodes (sender and receiver); and losses due to overflow in (2) input buffer and (3) output buffer. The first type of losses are recovered by the TCP acknowledgment and retransmit mechanisms (see §II-C)

The second type of losses will not occur thanks to the back-pressure mechanism of TCP. Indeed, the available space in the input buffer at the receiver node is advertised to the sender through the acknowledgments of TCP. The acknowledgment packet sent by the receiver of the TCP connection contains the space currently available in the receiver window. The sender will not send a new packet unless the new packet and those "in-fly" packets will have room in the receiver window. In addition, when the available input buffer space differs from the last advertised size by two Maximal Segment Size (MSS) or more, which can occur when packets are copied to the output buffers, the receiver sends a notification to the source via a special packet.

The last type of loss is avoided in our architecture. Indeed, a packet will be removed from the input buffer only when it is copied to all of the output buffers. The copy process is blocked when an output buffer is full and is resumed once there is room for one packet in that output buffer. Thus, due to this "blocking" back-pressure mechanism, there will be no overflow at the output buffers. In order to help the reader to identify the mechanisms used in end-systems, we have depicted (in Figure 1 (right)) a zoom of an end-system and of the information exchanged with its mother and daughter nodes. Each bar on this figure represents a sequence of events of a certain type (for instance the sequence of packets departure times from a buffer) and the labels on the arcs represent the lag in the packet sequence number between the events connected by this arc. The interpretation is as follows: for a bar with several incoming arcs, the m-th event takes place as soon as for all upstream bar, the event of order m minus the lag has taken place.

For instance packet m leaves the input buffer at the latest of the following 3 events: packet m arrived in the input buffer; packet $m - B^{(1,0)}_{\mathsf{OUT},(2,0)}$ has been acknowledged; packet $m - B^{(1,0)}_{\mathsf{OUT},(2,1)}$ has been acknowledged. These two types of back-pressure mechanisms guarantee that there will be no losses at the overlay nodes even if they have finite-size buffers. However, these mechanisms will also reduce the throughput of the group communication. It is crucial to understand the scalability issue, namely to check whether the throughput would vanish when the group size grows.

C. Packet Losses and Re-sequencing

Another factor that could significantly impact the throughput of the group communication is the packet re-sequencing delay due to losses. In case of a packet loss along a path, TCP retransmits the packet eventually. However, some packets with larger sequence number arrive before the duplicate (of the lost packet) arrives. These packets are not copied out to forwarding buffers until the duplicate arrives. Such a delay in packet processing has negligible impact to the throughput of the TCP connection in question thanks to the window inflation implemented by Fast Retransmit Fast Recovery [18]. However, it can create extra burstiness in packet arrival process of the downstream TCP connections. Such perturbations can cause significant performance degradation in these downstream TCP connections and has ripple effects in the corresponding subtrees. In turn, due to the back pressure mechanisms, these performance degradations impact the root sending rate, and therefore the group communication throughput. Thus, it is very important to understand how the scalability is affected by possible packet losses in the network.

D. End-to-End Reliability and Backup Buffers

With end-system multicast scheme, an important issue to address is resiliency, i.e., handling node failures and/or departures (possibly without prior notice). For this, one first needs to detect failures. Unfortunately TCP does not provide a reliable and efficient mechanism for detecting nodes that are not responding. Different methods can however be deployed for this purpose, e.g. heartbeat probes, keep-alive signals, etc. A heartbeat message is sent using UDP at regular time intervals to all neighbors of a node, and missing heartbeats from a neighbor indicate a failure. The keep-alive messaging system can be established in a similar way. In this paper, we assume that one of such mechanisms is deployed. The comparison of their efficiency is out of the scope of this paper.

Once a failure is detected, the tree needs to reconfigure in such a way that the daughter nodes of the failed node, as well as the subtrees they are rooted at, are re-attached to the original tree. A new TCP connection is established for each re-attachment. In other words, we need to find “step-mothers” for the daughter nodes of the failed node. There are a variety of ways to reconfigure the tree. We proposed and evaluated an algorithm in [19]. We will not discuss this issue in this paper due to space constraint.

In order to achieve end-to-end reliability, we need to ensure the data integrity while providing resiliency. In other words, we need to make sure that when these daughter nodes (of the failed node) are attached to the remaining tree, the new mother nodes have the data that are old enough so that these daughter nodes, as well as their offspring, receive the entire sequence of packets that the root sends out. For this purpose, we implement backup buffers in the end-systems so that whenever a new TCP connection is established, the packets in the backup buffer of the sender are sent out first via this new connection. In this way, the sender starts with these packets that have smaller sequence numbers than those in the input buffer of the sender.

We show in Section IV that if the size of the backup buffer is large enough compared to those of input and output buffers, then the end-to-end reliability is guaranteed even if there are multiple simultaneous failures.

III. Modeling and Scalability Analysis

The aim of this section is to prove that the reliable overlay multicast architecture described above is scalable in the sense that the throughput of the group is lower bounded by a positive constant irrespective of the size of the group. We show that even in a multicast tree of infinite size with the back-pressure mechanisms, the group throughput is positive, provided certain statistical assumptions hold on the individual point to point routes that basically guarantee the good behavior of each TCP connection in a stand-alone situation (e.g. bound on the number of hops, bound on the packet loss probability etc.). This is an unexpected result in view of the preliminary simulation results reported in [12], and also contrasts with the non-scalability results reported in the literature on IP-supported reliable multicast.

The packets of the multicast flow compete with those of other flows for the resources of each router-link in each TCP connection. On each such resource, this competition creates additional queuing delays between the processing of successive packets of the multicast flow. We model routers as single server queues with FIFO discipline and infinite buffer (losses are taken into account as an exogenous process, which controls the window evolution). In these queues, we only consider the packets of the multicast flow but we expand their processing times to random *aggregated service times* in order to take this effect into account (see [20], [16], [17] for more details on this representation of the influence of cross-traffic). The processing of packets inside the overlay network and the mechanisms already described can then be all explicitly described in a global discrete event dynamical systems, which is defined below.

The proof is made under a set of stochastic assumptions described below which covers both the case of ECN packet marking, as well as packet losses. We first present the model in the former case, and introduce later the more complex packet losses, retransmission and resequencing phenomena described in §II-C.

A. The Model for the ECN Packet Marking Case

We introduce the following notation:

- The *TCP connections* to end-system (k, l) is labeled with the index (k, l). It has a route that consists of a sequence of $H_{(k,l)}$ routers in series.
- *Routers* of TCP connection (k, l) are labeled by the index $h = 1, 2, \dots, H_{(k,l)}$. We denote accordingly the buffer for router h of connection (k, l) by (k, l, h). Each router is modeled as a single server queue containing only packets from the reference connection. The service time for these packets in this queue is a random variables describing the impact of cross traffic, also called *Aggregated Service Time*. It is denoted by $\sigma_m^{(k,l,h)}$ for packet m through router with index h in connection (k, l).
- We also introduce labels for the other *buffers* used by TCP connection (k, l): label $(k, l, \texttt{beg})$ denotes the output buffer of node $(k-1, \texttt{m}(k, l))$ on the socket corresponding to TCP connection (k, l) ; label $(k, l, \texttt{end})$ denotes the input buffer of node (k, l), that is the destination of this connection.

TCP Window flow control: Let $(W_m^{(k,l)})_{m\geq 1}$ denote the window size sequence for TCP connection (k, l). More precisely, $W_m^{(k,l)}$ is the window size seen by packet m. This sequence takes its values in the set $\{1, 2, \dots, W_{\max}\}$, where $W_{\max}$ is the maximal window size. We assume the following random evolution (corresponding to TCP RENO's congestion avoidance AIMD (Additive Increase Multiplicative Decrease) rule) for this sequence: when it is equal to w, the window increases of a size corresponding to one MSS after w packets (additive increase rule), if there is no packet marked with ECN ; when a packet is marked by one of the routers, the window is halved (multiplicative decrease rule); actually, an integer approximation of halving is used so as to keep the window in the set $\{1, 2, \dots, W_{\max}\}$; similarly, if the window is equal to $W_{\max}$, it remains equal to this value until the next packet marking. If one assumes packets to be marked independently with probability $p_{(k,l)}$, then $(W_m^{(k,l)})_m$is an aperiodic and ergodic Markov chain [20].
Remark: In this model, we are not including the Slow-Start phase, and Time-Outs that occur and reinitialize the window after a source starvation or a large variation of delay. Both can be taken into account in simulations but for the cases that we are presenting in this paper, we have observed that they do not impact the long-term throughput. Also, we assume that each packet is acknowledged. In TCP implementations, an acknowledgment is sent for every second segment. This is taken into account by saying that a packet transmission in the model represent the transmission of two MSS in the TCP connection.

B. Evolution Equation

The system that we described above can still be represented exactly using (max,plus) linear recursions, that we know discuss in detail. We shall see that they are instrumental for the scalability analysis and for simulations of large networks presented in §V-A.

Let T_m, $m \geq 0$, denote the time when packet m is available at the root node. In this work we assume a saturated root, where all packets are ready at the root from the beginning of the communication; namely $T_m = 0$ for all m. Let $x_m^{(k,l,h)}$ denote the time when m has completed its transmission and leave the buffer with label (k, l, h). In particular for $h = 1, \dots, H_{(k,l)}$, this is the time when router h in connection (k, l) has completed the service for packet m. $x_m^{(k,l,\texttt{beg})}$ is the time when packet m departs from the output buffer of the source node of TCP connection (k, l), and arrives in the input buffer of router $h = 1$. Finally $x_m^{(k,l,\texttt{end})}$ is the time when packet m departs from the input buffer of the receiver node of TCP connection (k, l), and is transmitted into each output buffer of node (k, l).

The network (except for the root) is assumed to be empty initially of any packets of the communication. This can be represented by taking $x_m^{(k,l,h)} = -\infty$ for any $m < 0$. Then, with the above assumptions, the dynamics of the model presented in the last subsection is given by the following evolution equations at the root node :

$$\begin{aligned} x_m^{(0,0,\texttt{beg})} &= T_m \vee x^{(0,0,\texttt{end})}_{m-B_{\texttt{IN}}^{(0,0)}} \\ x_m^{(0,0,\texttt{end})} &= x_m^{(0,0,\texttt{beg})} \vee \Big(\bigvee_{l\in \texttt{d}(0,0)} x^{(1,l,H_{(1,l')})}_{m-B^{(0,0)}_{\texttt{OUT},(1,l)}} \Big) \end{aligned}$$

where $\vee$ denotes the maximum.

For a node $(k,l),\ k \geq 1,\ l \geq 0$:

$$\begin{aligned}
x_m^{(k,l,\mathsf{beg})} &= x_m^{(k-1,\mathsf{m}(k,l),\mathsf{end})} \vee x_{m-B_{\mathtt{IN}}^{(k,l)}}^{(k,l,\mathsf{end})} \vee x_{m-W_m^{(k,l)}}^{(k,l,H_{(k,l)})}\\
x_m^{(k,l,1)} &= \left(x_m^{(k,l,\mathsf{beg})} \vee x_{m-1}^{(k,l,1)}\right) + \sigma_m^{(k,l,1)}\\
&\dots\\
x_m^{(k,l,H_{(k,l)})} &= \left(x_m^{(k,l,H_{(k,l)}-1)} \vee x_{m-1}^{(k,l,H_{(k,l)})}\right) + \sigma_m^{(k,l,H_{(k,l)})}\\
x_m^{(k,l,\mathsf{end})} &= x_m^{(k,l,H_{(k,l)})} \vee \bigvee_{l' \in \mathsf{d}(k,l)} x_{m-B_{\mathtt{OUT},(k+1,l')}^{(k,l)}}^{(k+1,l',H_{(k+1,l')})}.
\end{aligned}$$

C. Path of Maximal Weight in a Random Graph

Let G be the graph where the set of vertices is:

$$\begin{aligned}
V = &\{(0,0,\mathsf{beg},m),(0,0,\mathsf{end},m),\ m \in \mathbb{Z}\} \cup \{(k,l,h,m),\\
&k \geq 1,\ l \geq 0,\ h \in \{\mathsf{beg},1,\dots,H_{(k,l)},\mathsf{end}\},\ m \in \mathbb{Z}\}
\end{aligned}$$

and the set of edges E is: $E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$ with:

$$\begin{aligned}
E_1 = &\{(0,0,\mathsf{end},m) \to (0,0,\mathsf{beg},m) | \forall m \in \mathbb{Z}\} \cup \{(k,l,1,m)\\
&\to (k,l,\mathsf{beg},m),\ (k,l,\mathsf{end},m) \to (k,l,H_{(k,l)},m)\\
&| \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\} \cup \{(k,l,h,m) \to (k,l,h-1,m)\\
&| 2 \leq h \leq H_{(k,l)}, k \geq 1, l \geq 0, m \in \mathbb{Z}\} \cup \{(k,l,\mathsf{beg},m)\\
&\to (k-1,\mathsf{m}(k,l),\mathsf{end},m) | \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\}\\
E_2 = &\{(k,l,h,m) \to (k,l,h,m-1) | \forall h, k \geq 1, l \geq 0, m \in \mathbb{Z}\}\\
E_3 = &\{(k,l,\mathsf{beg},m) \to (k,l,H_{(k,l)},m-W_m^{(k,l)})\\
&\ |\ \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\}\\
E_4 = &\{(k,l,\mathsf{beg},m) \to (k,l,\mathsf{end},m-B_{\mathtt{IN}}^{(k,l)})\\
&\ |\ \forall k \geq 0, l \geq 0, m \in \mathbb{Z}\}\\
E_5 = &\{(k,l,\mathsf{end},m) \to (k+1,l',H,m-B_{\mathtt{OUT},(k+1,l')}^{(k,l)})\\
&\ |\ \forall l' \in \mathsf{d}(k,l) \text{ and } \forall k \geq 0, l \geq 0, m \in \mathbb{Z}\}\ .
\end{aligned}$$

The weight of vertex (k,l,h,m) is given by $\sigma_m^{(k,l,h)}$ for $h \in \left\{1,2,...,H_{(k,l)}\right\}$ and $m \in \mathbb{Z}$, and is equal to zero for $h \in \{\mathtt{beg},\mathtt{end}\}$. The weight $\mathtt{Wei}(\pi)$ of a path π in G is defined as the sum of the weights of the vertices of π.

All the buffers in the end-systems are initially empty. We define accordingly the restriction of this graph denoted by $G^{[0]}$, in which the weight of vertex (k,l,h,m) is taken equal to $-\infty$ for all vertices with $m < 0$. It is then possible to show by an induction argument based on Equations shown in §III-B that for all k,l,h,m

$$x_m^{(k,l,h)} = \max_{\pi \text{ a path in } G^{[0]},\ (k,l,h,m) \rightsquigarrow (0,0,\mathsf{beg},0)} \{\mathtt{Wei}(\pi)\}\ .$$

D. Model for Packet Losses

Our aim in this section is not to build an exact model for the case with packet losses, but rather to describe a simplified and tractable model obtained via a set of conservative transformations. For proving the scalability in this case (namely the positiveness of throughput in the exact model for an infinite tree), it will hence be enough to prove that the simplified conservative model scales in the same sense.

Imagine packet with index m is lost. Before TCP is aware of this loss, some of the packets $m+1,\dots,m+W_m$ might have left from the source. They have been in fact allowed by the congestion window mechanism, but they may not have left the source due to the back-pressure mechanism, or because they are not available at this time in the source node. In all cases, one can see that the following simplified window evolution:

- the window is set to $\max((W_m-1)/2,1)$ for $m+1, m+2, \dots, ..., m+W_m+\max((W_m-1)/2,1)-1$;
- the additive increasing evolution of the window is resumed from packet $m+W_m+\max((W_m-1)/2,1)$ on

is then conservative in that the true system using Fast Retransmit Fast Recovery (see [18]) will have larger windows at all times and hence better throughput than this considered simplified model (more details in [19]).

In addition to that, the loss of packet with index m has two major impacts. First, retransmissions are sent by the sources. We can choose to include them at the last possible step of the communication (between packet $m+W_m$ and $m+W_m+1$). This tends to overload the network at intuitively the worst time (after the self clocking mechanism have resumed with a half window), and this is what is done in our simulations. Another even more conservative choice is to include retransmissions at every possible step (after each packet $m, m+1, \dots, m+W_m$) to cover each possible case of the exact model. This is what we choose in our model, as the scalability result can still be proven under this assumption. The number of retransmissions at each step is W_m in our model by default, again corresponding to a conservative assumption, it is only one packet if SACK is implemented by TCP. The second consequence of packet m being lost is that some packets $m+1, m+2, \dots$ are blocked in the input buffer of the destination node, as packets need to be processed by the node according to the order defined by the sequence number. In our model, we have chosen to include this blocking for all possible packets involved (i.e. packets $m, m+1, \dots, m+W_m$ are not processed in the destination node until packet $m+W_m$, and the retransmissions, arrive).

1) Random Graph Model: In this section, vertices of the graph associated with the index m will refer either to packet m itself, or to a retransmitted packet that was sent after packet m and before packet $m+1$. In the random graph associated with the loss model, we denote the vertex for end-system (k,l), packet m and index h by $\mathsf{v}(k,l,h,m)$. For all $k, l, h = 1,\dots,H_{(k,l)}$ and m, we add a vertex $\mathsf{v}'(k,l,h,m)$ on top of $\mathsf{v}(k,l,h,m)$, which represents the potential retransmissions of packets just between packets m and $m+1$. The weight associated with this vertex is independent with the same law than $\sigma_m^{(k,l,h)}$ in the case where TCP is implementing SACK. If this is not the case, it should be equal to a sum of $2 \times W_m$ independent variables with this law, as several packets may be retransmitted.

We also add the following edges to link this vertex to the vertical and horizontal structure.

- Horizontal edges: $\mathsf{v}'(k,l,1,m) \to \mathsf{v}(k,l,\mathsf{beg},m)$ and $\mathsf{v}'(k,l,h,m) \to \mathsf{v}'(k,l,h-1,m)$ for $h = 2 \dots H$,
- Vertical edges: $\mathsf{v}'(k,l,h,m) \to \mathsf{v}(k,l,h,m)$ for $h = 1 \dots H$.

and, to represent the effect of the loss and the retransmission of packet m on the TCP connection (k,l) :

- Edge E_6: $\mathsf{v}(k,l,\mathsf{end},m) \to \mathsf{v}'(k,l,H_{k,l},m+W_m-1)$ in order to represent the re-sequencing constraints on packets $m, m+1, \dots, m+W_m-1$.
- Edges E_7: $\mathsf{v}(k,l,h,m''+1) \to \mathsf{v}'(k,l,h,m'')$ for all $h = 1, \dots, H_{k,l}$ and $m'' = m, \dots, m+W_m$ to represent the retransmission of packet m (as the extra packet in between indices $m+W_m-1$ and $m+W_m$) which delays the following packets.

The complete graph (including all types of edges $E_1, \dots, E_7$) is presented in Figure 2 in the case of two TCP connections in tandem with $B_{\mathtt{IN}} = B_{\mathtt{OUT}} = B$.. Edges belonging to E_7 are the vertical local edges in red. For readability purpose, edges belonging to other classes than E_1 and E_2 have been represented only when they depart from station k and packet m.

E. The Last-Mile Effect

The following issue should be taken into account in the models: if the out-degree of some node of the tree is large, then the access link from this node may become the actual bottleneck due to the large number of simultaneous transfers originating from this node. Hence the throughput of the transfers originating from this node may in fact very well be significantly affected by the other transfers originating from the very same node. This ”last-mile link” effect can be incorporated in our model. The extra traffic created by the transfers not located on the reference path can be represented by an increase of the aggregated service times (we remind that aggregated service times represent the effect of cross traffic on the reference TCP transfer). In order to keep this under control, the general idea is then to keep a deterministic bound, say D, on the out degree of any node in the tree. Using arguments similar to those in [17], it is easy to show that provided bandwidth sharing is fair on the last-mile link, then the system where all aggregated service times on this link are multiplied by D is a conservative lower bound system in terms of throughput.

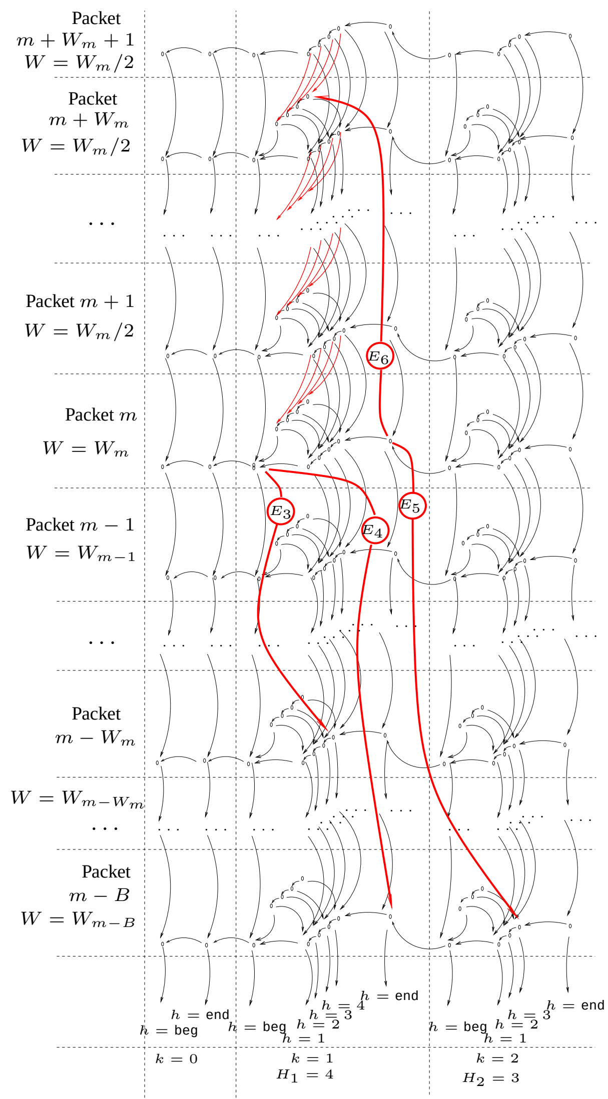


Fig. 2. Random Graph Representing two TCP Connections in tandem with Back-Pressure and Re-sequencing Constraints

Hence, whenever the out-degree of any node is bounded by a constant, the proof of the scalability of throughput for the case without this last-mile effect extends to a proof of scalability with this effect taken into account.

F. Throughput Scalability Result

In this section we are interested in the throughput of the group communication when the size of the tree gets large. For this, we directly consider infinite trees.

We call the *homogeneous* model (resp. the *non-homogeneous* model) the case where: Each node in the tree has a *fan-out degree* fixed D (resp. bounded from above by D); All back-pressure *buffers* have the same size for any end-system in the tree (resp. they are bounded from below by constants $B_{\mathtt{IN}}$ and $B_{\mathtt{OUT}}$); The *routes* used by all TCP connections are structurally and statistically equivalent (resp. offer minimal structural and statistical guarantee). More precisely, the number of hops H is the same for all connections (resp. bounded from above by H) ; the ECN packet marking or packet losses process is independent and identically distributed in all connections with probability p (resp. independent with a probability that is smaller than p) ; and finally the aggregated service times are independent in all routers, and identically distributed with law σ (resp. it is bounded in the strong ordering sense from above by a random variable σ) with a finite mean.

1) Definition of Group Throughput: Using Kingman subadditive ergodic theorem, it can be shown (see [19]) that in the homogeneous case, there exists a constant $\gamma \in \mathbb{R} \cup \{+\infty\}$ such that the almost sure (a.s.) convergence holds for any (k,l,h): $\lim_{m\to\infty} \frac{m}{x_m^{(k,l,h)}} = \gamma$.
We call γ the *asymptotic (long-term) group throughput*.

For the non-homogeneous case, we can show that $\liminf_{m\to\infty} \frac{m}{x_m^{(k,l,h)}} \geq \gamma$, where γ is the group throughput of the homogeneous model defined with the bounds.

2) Scalability under Light Tailed Assumptions: We consider the non-homogeneous model, and we assume that the random variable σ is light tailed, i.e. such that $\exists \tau > 0$ such that $\forall 0 \leq t \leq \tau$, $\mathbb{E}[e^{t\sigma}] \leq A(t) < +\infty$.

The next result is based on the following fact : for a given (k,h), the length of any path of $G^{[0]}$ involved in the computation of $x_m^{(k,l,\mathsf{end})}$ is bounded from above by a linear function of m. As the number of neighbors of any node in the graph is uniformly bounded, this leads to an exponential bound on this number of paths, and hence on the weight of the maximal weight path. The complete argument can be found in [19].

Theorem 1 *Consider an overlay multicast tree with infinite height $k = 0,1,2,\ldots$. Under the assumptions that the law of σ is light tailed, we have*

$$\liminf_{m\to\infty} \frac{m}{x_m^{(k,l,end)}} \geq \mathtt{Const}(H,D) > 0 \;\; a.s. \, .$$

uniformly in (k,l), both for the ECN and packet losses cases, where D is the bound on the degree and H that on the hop number. In the particular case of a homogeneous network, the group communication throughput γ is bounded from below by a positive constant that does not depend on the size of the tree.

IV. End-to-End Reliability

As we mentioned in Section III, using TCP for point-to-point connections guarantees reliable transfers between the nodes of the group, but does not provide uninterrupted transmission in cases when a transit node suddenly stops functioning. Daughter nodes of the failing node must restore connection to the multicast group, and they should receive all stream packets.

Avoiding interruption in packet sequence may not be trivial, especially for nodes distant from the root, since the packets that these nodes were receiving at the time of failure may have been already processed and discarded by all other group members, except for the failed node. We employ *backup buffers* to create copies of stream content which could be otherwise lost during node failure. Figure 1 (right) illustrates our approach. While data is moved from the input buffer to the output buffers, a copy of data leaving input buffer is saved in the backup buffer. The backup buffer can then be used to restore packets which were lost during node failure.

We will show below that this end-to-end reliability can be achieved through the backup buffers, provided they are sized appropriately. We will formally derive a formula for the size of the backup buffer. We shall also present leave/join algorithms so as to keep the group throughput scalable.

Definition of End-to-End Reliability. We define overlay multicast system to be *end-to-end reliable with tolerance to r failures*, if after removing *simultaneously* r nodes from the multicast tree and restoring connectivity, transmission can be continued, and all remaining nodes receive entire transmission in the same sequence. In other words, failure of r nodes does not lead to any changes in the sequence or content of the stream received at the remaining nodes. However recovering from failure may incur a delay, which is required to restore connectivity.

During the time when the system is recovering from r failures, it is not guaranteed to recover correctly from any additional failures. However if l, for some $1 \leq l \leq r$, failures occur, the system will be able to recover from additional $(r-l)$ failures even if the failures happen before the system has completely recovered. In such situations new failures occurring during recovery will increase total recovery time.

Let $B_{\mathtt{OUT}}^{\max}$ and $B_{\mathtt{IN}}^{\max}$ be the maximum sizes of output and input buffers in the system, respectively. A *backup buffer of order r* has size $(r \cdot (B_{\mathtt{OUT}}^{\max} + B_{\mathtt{IN}}^{\max}) + B_{\mathtt{OUT}}^{\max})$.

Failure Recovery and Leave/Join Operations. We use the following simple algorithm to recover from failures and restore connectivity when nodes leave. We shall call node (k',l') *surviving ancestor* of node (k,l), if the mother of node (k,l) did not survive the failure, and (k',l') is the first surviving node on the path from (k,l) to the root. Each disconnected end-system (k,l) must be reconnected to a node that belongs to the subtree of the surviving ancestor (k',l'). After connection is restored, the node (k', l') stops receiving and retransmits all packets contained in its backup buffer. Then it continues the transmission as usual. Intermediate nodes on the new path from (k', l') to (k, l), as well as all nodes in the entire subtree of (k, l), must be able to ignore the packets that they have already received, and simply forward them to downstream nodes. Proof of the following theorem and a more detailed description of leave/join procedures are presented in [19].

Theorem 2 *An overlay multicast system with backup buffer of size* $(r \cdot (B_{OUT}^{\max} + B_{IN}^{\max}) + B_{OUT}^{\max})$ *is end-to-end reliable with tolerance to* r *simultaneous and consecutive failures in a chain of the tree.*

Note that this result is slightly stronger than the tolerance to r failures. A backup buffer of order r can tolerate r consecutive failures in a chain.

In the case with random failures, an interesting question is that of the evaluation of the stationary probability of *network failures* (defined as node failures which cannot be recovered), in function of the backup buffer size (r), the node failure rate and the recovery rate. The simplest model is when we assume that the times in between node failures are independent and identically distributed (i.i.d.) exponentials, and that node recovery times are i.i.d. exponentials too, then the network failure probability can be upper bounded by the loss probability in the $M/M/r/r$ queue, where the service times are the recovery times and the arrivals correspond to node failures. Of course, a realistic model would have an arrival rate that depends on the size of the tree, and possibly more general statistics than exponential. The analysis of such scenarios will be the topic of a companion paper.

V. Simulation and Experimental Results

We have carried out extensive simulations and a number of experiments in order to support and complement our theoretical investigations presented in the previous two sections. In this section, we report and comment on these empirical results.

A. Scalability Analysis using Simulation

We first report the simulation studies on the scalability issue. In particular we analyze the throughput obtained for long file transfers in large groups. We use for this purpose a (max,plus) simulator based on the evolution equations of §III-B. Its main advantage of this equation-based simulator compared to the traditional discrete-event simulators is that it allows one to handle much larger trees.

We have chosen MSS=100B, so that a packet is 200B (see the discussion at the end of §III-A). In each simulation run, we simulate the transmissions of 10M packets (or 2GB of data). We only report results on the complete binary tree case. Each TCP connection involved goes through 10 routers in series, and all the packets transmitted on this connection have an independent probability p to get a negative feedback (loss or ECN marking). By default, $p = 0.01$.

In this paper we choose $B_{\mathtt{IN}} = 50$ packets (i.e. 10KB), and $B_{\mathtt{OUT}}$ varies as 50,100,1000 and 10 000 packets (resp. 10KB, 20KB, 200KB, 2MB). Consequently, for each connection $W_{\max} = \min(B_{\mathtt{IN}}, B_{\mathtt{OUT}}) = 50$ packets.

As is described in § III, the cross traffic is characterized by the Aggregated Service Times in each router. In these simulations we have considered both exponential (the default option) and Pareto random values with mean equal to 10ms for each router/link.

Throughput Scalability. We have simulated complete binary trees of sizes up to 1023 nodes, with different variants of handling of losses: TCP RENO type (with fast retransmit), TCP SACK and TCP over ECN. We also considered the impact of output buffer size. Figure 3 (left) illustrates the throughput as a function of the group size in the case of TCP SACK. It is easy to see that, quite intuitively, the group throughput is a decreasing function of the group size and an increasing function of the output buffer size. Observe that when the output buffer is large (say more than 1000 packets), the throughput flattens out very quickly with small groups (less than 10 nodes). For smaller output buffers, the convergence to the asymptotic throughput can be observed when the group size reaches 100 nodes. The two other variants of TCP exhibit similar behavior: with the same configuration, TCP without SACK has a throughput that is about 8% less than that of TCP SACK; whereas TCP over ECN has slightly better throughput with about 2% improvement over TCP SACK.

Comparison between Asymptotic Throughput and Single Connection Throughput. In [17] it was shown for the case without back pressure that the group throughput is equal to the minimum of those of the single connections without constraint on the packet availability at the senders (such throughput is referred to as *local throughput*). Thus, for the homogeneous case, this translates into the fact that the group throughput is identical to the local throughput. In our case, there is no hope that such a relation holds due to the back pressure mechanisms. It is however interesting to know how far the group asymptotic throughput is from the local throughput. In the following table, we report the ratio of these two quantities:

Buffer (Pkts)	50	100	1,000	10,000
TCP RENO	.83	.90	.98	.99
TCP SACK	.86	.92	.99	.99
TCP ECN	.87	.92	.99	.99

It is worthwhile observing that the group throughput with large output buffers is very close to the local throughput. In other words, large output buffers alleviate in a very significant way the effect of the back pressure mechanisms. Even if the output buffer is small, say 50 packets (identical to the input buffer), the degradation of the group throughput due to back pressure mechanisms is moderate (less than 18%).

Remark: The previous result, which shows that back-pressure leads to moderate throughput degradation, is not restricted to the homogeneous case. Under the heterogeneous model assumptions, a simple stochastic monotonicity argument shows that the group throughput is larger than that of the homogeneous case where each connection is replaced by a connection stochastically equivalent to the one with the worst long term average. So, if we consider the same metric as above, namely the group throughput degradation compared to the reference throughput (the minimum long term average throughput among all point to point connections in a stand-alone mode), then the degradation in the heterogeneous case is less than that of the homogeneous one.

Impact of Cross Traffi c Distribution. In our model, cross traffic at the routers is represented through aggregated service times. The impact of the distribution of aggregated service times is studied in Figure 3 (middle) via simulation. In this figure, we plot the group throughput as a function of the group size for exponential and Pareto distributions with different parameters. We can see that the heavier the tail of this distribution, the smaller the throughput. We also observe that even for heavy tail distributions like Pareto, when the second moment is finite (e.g. for the case with parameter 2.1) the throughput curve has a shape similar to that of the exponential distribution. However, when the second moment no longer exists, the throughput curve decays much faster. This suggests that the light tail distribution assumption in Theorem 1 might possibly be relaxed and replaced by some moment conditions (see §VI).

Impact of Packet Loss Probability. As we pointed out earlier in this section, the asymptotic group throughput is relatively close to the throughput of a single connection (i.e. local throughput) when the output buffer is large. We have done extensive simulations which suggest that, even with the back pressure mechanisms, the group throughput has a similar shape as that of the single-connection throughput. Figure 3 (right) illustrates the group throughput as a function of packet loss probability in a particular case. One immediately notices that the single connection throughput (i.e. local throughput) is very close to those of the group of size 126.

B. Experimental Results of Scalability and Reliability

In order to evaluate practicality of our models, we have implemented a prototype of TCP overlay multicasting system. We used Planet-Lab network [21] which gives access to computers located in universities and research centers over the world. Our implementation runs a separate process for each output and input buffer, which are synchronized via semaphores and pipes. As soon as data is read from input buffer, it is available for outgoing transmissions. A separate semaphore is used to ensure that data is not read from input socket, if it can not be sent to output buffers, which creates back-pressure. A dedicated central node was used to monitor and control progress of experiments.

Scalability Analysis. To analyze scalability of throughput, we constructed a balanced binary tree of 63 nodes connected to the Internet. We started simultaneously transmissions in balanced sub-trees of sizes 15, 31 and 63 with the same root. Running experiments simultaneously allowed us to avoid difficulties associated with fluctuation of networking conditions. In this way, link capacities are always shared between trees of different sizes in roughly equal proportions across the trees. We measured throughput in packets per second, achieved on each link during transmission of 10MB of data. Throughput of a link was measured by receiving node. In the following table, we summarize group throughput measurements for 3 different tree sizes and 3 different settings for output buffer size. Group throughput is computed as the minimum value of link throughput observed in the tree. Similarly to our simulations presented above, size of each packet is 200 bytes, size of the input buffer is 50 packets, and size of the output buffer is variable.

Group size:	15	31	63
Buffer=50 Pkts	95	86	88
Buffer=100 Pkts	82	88	77
Buffer=1000 Pkts	87	95	93

One can observe that the group throughput changes very little in the group size. This is consistent with the simulation results reported above, although as is quite expected, the absolute numbers are different.

Reliability Analysis. To verify our approach to recovery after failures, we implemented a failure-resistant chain of 5 nodes running on Planet-Lab machines. During the transmission of 10 megabytes of data, two of

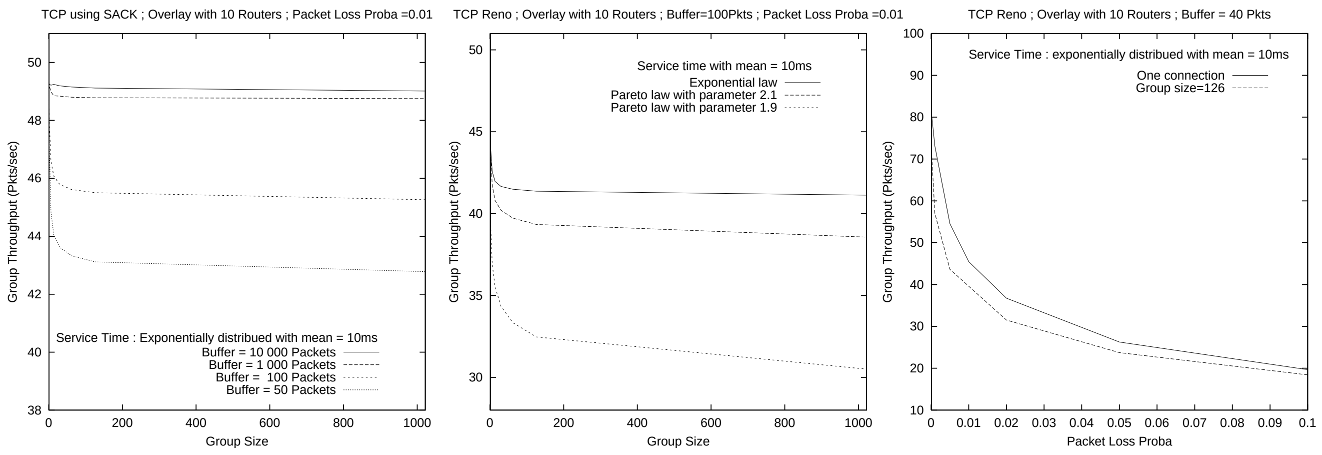


Fig. 3. Group throughput: as a function of group size, with TCP SACK, and exponential cross traffic (left), for several laws of cross traffic with TCP Reno (middle) as a function of packet loss probability for TCP Reno (right).

5 nodes fail. The failures are not simultaneous, and the system needs only to be resistant to one failure. In this experiment we limit both input and output buffer size to 50 packets. As in the previous experiment, size of each packet is 200 bytes (MSS=100 bytes). Our failure recovery algorithm needs a backup buffer of size 150 in this case. We have performed 10 runs of this experiment and measured group throughput, reconnection time and the number of redundant packets that are retransmitted after the connection is restored. Recall that in our architecture, the packet sequence numbers do not need to be advertised during the re-attachment procedure. Thus the daughter nodes of the failed node may receive duplicated packets after the connections are re-established. These redundant transmissions can impact the group throughput.

In our implementation, the failing node closes all its connections, and failure is detected by detecting dropped connections. After the failure is detected, the orphaned daughter node listens for an incoming connection from the surviving ancestor. We measure the interval between the time when failure is detected, and the time when connection is restored. This time interval is measured separately at the two participating nodes: surviving mother (M), and daughter (D). The results of our measurements are summarized in Figure 4 (top). The average reconnection time in seconds and number of retransmitted packets per one failure are given per one failure. The average group throughput is given per experiment. In these experiments, the average number of retransmitted packets is about half of the backup buffer size. The TCP sessions are re-established in a few seconds, in the same order as the TCP timeout. As the failure detection can be achieved in a few seconds as well, our experiment results show that the entire procedure of failure detection and reconnection can be completed in a few seconds.

Scalability vs. Reliability. Simulation results presented above have shown that when there are no failures, the larger the buffers the more scalable the group throughput is. However, with larger buffers, the backup buffer size has to be increased proportionally in order to guarantee the end-to-end reliability. The above experiment showed that when failures do occur, the redundant transmissions will be increased as a consequence of larger backup buffers. These redundant transmissions will in turn reduce the group throughput. We here investigate into this issue. We consider a chain of 10 nodes and we generate 2, 4 and 6 failures (in a sequential way, so that the system just need to tolerate 1 failure). Figure 4 (bottom) reports the throughput measurements obtained with these settings and with different output buffer sizes. The backup buffer size is set to the input buffer size and twice the output buffer size. It is interesting to see that when the buffer sizes increase, the group throughput can actually decrease. While we cannot make general claims based on these observations alone, these experiments do show that the throughput monotonicity in buffer size no longer holds in the presence of failures. The more frequent the failures are, the more severe (negative) impact large buffers would have on the group throughput.

	min	average	max
Thput (Pkts/sec)	49.05	55.24	57.65
# of Retransmitted Pkts	34	80.5	122
Reconnection time (D)	0.12	3.53	5.2
Reconnection time (M)	0.27	3.81	5.37

Buffer (Pkts)	50	200	500	1000
for 2 failures	25.6	26.8	45.2	31.5
for 4 failures	29.2	28.8	36.4	27.2
for 6 failures	30.9	28.8	30.8	24.0

Fig. 4. End-to-End Reliability Experiments in Planet-Lab (top), Scalability vs End-to-End Reliability: Throughput in KB/s (bottom).

VI. Conclusions

Our first conclusion is that reliable multicast overlays can be deployed on top of the current TCP/IP by adding a light set of application layer back-pressure mechanisms that guarantee both end-to-end flow control and reliability. The one-to-many TCP overlay architecture is TCP based and hence TCP friendly; in particular it adapts the local rate in each part of the tree to the state of congestion of this part. It is also fully decentralized in that the control actions taken by any given point to point TCP connection propagate to the whole tree via neighboring end-systems only; in particular, there is no need for end-systems to send information back to the root of the tree as in rate control based architectures.

A second important observation concerns the fear that as more and more TCP connections get interconnected in such a multicast overlay, some slow down experienced by distant connections might propagate to the root via back-pressure, leading the group throughput to vanish as the number of end-system grows. We have shown that such a fear has no grounds, provided all point to point connections that are used within the overlay use routes that offer minimal quality guarantees and provided the fan-out degree of the multicast tree is bounded. Such architectures can be used for group communications of arbitrarily large sizes and still provide a group throughput that is close to that of a single point to point connection with these minimal guarantees.

Surprisingly, this conclusion holds true even in the case of moderate input and output buffers. Moderate buffers even seem to be a good tradeoff within this context: they allow more efficient recovery mechanisms in case of failures and, according to our simulation results, they do not affect too severely the group throughput if not too small. An optimal buffer size offering a good compromise between throughput and reliability could in principle be advertised to the group.

One possible extension of this work is to consider structures other than a single multicast tree to achieve higher throughput and resiliency. Multiple distribution trees can be used simultaneously, see for example SplitStream [22] and CoopNet [23]. SplitStream splits the data into a number of sections and dispatches the sections to different edge-disjoint trees. Indeed, the framework that we propose can readily be applied in this context. Each multicast tree can be implemented using the one-to-many TCP Overlay, thus reaching throughput larger than that of a single tree, while preserving the scalability and reliability characteristics of our architecture.

Another direction of future investigation is the relaxation of the light-tail assumption made in Theorem 1. Indeed, in the case of an infinite chain (a special case of tree), it can be shown, by applying a result of [24], that when the aggregated service times have finite moments of order more than two, then the group throughput is lower bounded by a strictly positive constant. Our simulation results also suggest that such a moment condition might be sufficient for the scalability in the tree case.

References

[1] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *ACM Sigmetrics*, 2000.

[2] B.Zhang, S.Jamin, and L.Zhang, "Host multicast: A framework for delivering multicast to end users," *IEEE Infocom*, 2002.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *ACM Sigcomm*, 2002.

[4] J. Liebeherr and M. Nahas, "Application-layer multicast with delaunay triangulations," *IEEE Globecom*, 2001.

[5] Y. Chawathe, S. McCanne, and E. A. Brewer, "RMX : reliable multicast for heterogeneous networks," *IEEE Infocom*, 2000.

[6] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable multicasting with an overlay network," *4th OSDI*, 2000.

[7] G. Kwon and J. Byers, "ROMA: Reliable overlay multicast with loosely coupled TCP connections," *IEEE Infocom*, 2004.

[8] P. H. Hsiao, H. T. Kung, and K. S. Tan, "Active delay control for TCP," *IEEE Globecom*, 2001.

[9] P. Mehra, A. Zakhor, and C. D. Vleeschouwer, "Receiver-driven bandwidth sharing for TCP.," *IEEE Infocom*, 2003.

[10] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays.," *ACM Sigmetrics*, 2003.

[11] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure," *3rd USITS*, 2001.

[12] G. Urvoy-Keller and E. Biersack, "A congestion control model for multicast overlay networks and its performance," *NGC 2002*.

[13] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM ToN*, vol. 5, no. 6, 1997.

[14] B.N. Levine and J.J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols," *ACM Multimedia Systems*, 1998.

[15] S.Bhattacharyya, D.Towsley, and J.Kurose, "The loss path multiplicity problem in multicast congestion control," *IEEE Infocom*, 1999.

[16] A. Chaintreau, F. Baccelli, and C. Diot, "Impact of TCP-like congestion control on the throughput of multicast group," *IEEE/ACM ToN*, vol. 10, no. 4, 2002.

[17] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu, "Scalability of reliable group communication using overlays," *IEEE Infocom*, 2004.

[18] M. Allman and V. Paxson, "RFC 2581 - TCP congestion control," 1999, at `www.ietf.org/rfc/rfc2581.txt`.

[19] F. Baccelli, A. Chaintreau, Z. Liu, and A. Riabov, "The one-to-many TCP overlay: A scalable and reliable multicast architecture," *INRIA Research Report number 5241*, 2004.

[20] F. Baccelli and D. Hong, "TCP is max-plus linear and what it tells us on its throughput," *ACM Sigcomm*, 2000.

[21] "Planetlab, an open platform for developing, deploying, and accessing planetary-scale services," `www.planet-lab.org`.

[22] M.Castro, P.Druschel, A-M.Kermarrec, A.Nandi, A.Rowstron, and A.Singh, "SplitStream: High-bandwidth multicast in a cooperative environment," *SOSP 2003*.

[23] V.Padmanabhan, H.Wang, P.Chou, and K.Sripanidkulchai, "Distributing streaming media content using cooperative networking," *ACM NPSSDAV 2002*.

[24] J. Martin, "Large tandem queueing networks with blocking," *Queueing Systems, Theory and Applications*, vol. 41, 2002.