HAL
open science

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

François Baccelli, Augustin Chaintreau, Zhen Liu, Anton Riabov

To cite this version:

François Baccelli, Augustin Chaintreau, Zhen Liu, Anton Riabov. The One-to-Many TCP Overlay: A Scalable and Reliable Multicast Architecture. [Research Report] RR-5241, INRIA. 2004, pp.41. inria-00070757

HAL Id: inria-00070757

https://inria.hal.science/inria-00070757v1

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

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

François Baccelli — Augustin Chaintreau — Zhen Liu — Anton Riabov

N° 5241

Juin 2004

THÈME 1

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5241--FR+ENG

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

François Baccelli*, Augustin Chaintreau †, Zhen Liu‡, Anton Riabov§

Thème 1 — Réseaux et systèmes
Projet TREC

Rapport de recherche n° 5241 — Juin 2004 — 41 pages

Abstract: This work addresses two key issues in reliable multicast overlay networks : end-to-end reliability in the presence of node failure/overflow, and throughput scalability in the presence of random perturbations. A decentralized architecture is proposed for taking care of these two issues. This architecture uses distinct point to point TCP connections between adjacent pairs of end-systems, together with a window based back-pressure control, which links adjacent pairs of TCP connections via an application-layer mechanism. Each end-system maintains forwarding buffers and a back-up buffer storing copies of the forwarded packets. The forwarding buffers are used to enforce the back-pressure mechanism whereas the back-up buffer is dedicated to re-establishing connectivity in case of end-system failure. This architecture, that we propose to 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, which is of independent interest, contrasts with known results about the non-scalability of IP-supported multicast for reliable group communication.

Key-words: Multicast, Application Layer, Network Overlay, TCP tandem.

* INRIA-ENS, 45 rue d'Ulm 75005, Paris, France, francois.baccelli@ens.fr
† INRIA-ENS, 45 rue d'Ulm 75005, Paris, France, augustin.chaintreau@ens.fr
‡ IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, zhenl@us.ibm.com
§ IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, riabov@us.ibm.com

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 —Télécopie : +33 1 39 63 53 30

Le réseau applicatif "TCP d'un vers beaucoup": une architecture extensible pour la diffusion multipoint fiable

Résumé : Ce travail étudie la diffusion multipoint fiable construite sur un réseaux de connexions entre terminaux (également appelé réseau applicatif). Nous analysons deux aspects de ce type de communication : la fiabilité de bout-en-bout, qui concerne la garantie de bonne réception de toutes les données, y compris en cas d'arrêt de certains terminaux ou de pertes dans ces terminaux; l'extensibilité, qui concerne le maintien d'un débit acceptable même pour des groupes de grande taille. Nous proposons d'assurer ces deux propriétés de manière décentralisée en utilisant des connections TCP distinctes entre terminaux adjacents et un mécanisme de "pression-inverse" par fenêtre glissante, qui lie les connexions adjacentes. Chaque terminal maintient des mémoires tampons qu'il utilise pour le réacheminement des données, ainsi qu'une mémoire de sauvegarde, conservant la copie des paquets qui l'ont quitté. Les premiers assurent, avec le mécanisme de pression-inverse, l'adaptation de la vitesse d'écoulement des paquets aux conditions du réseau. La mémoire de sauvegarde intervient à la disparition d'un terminal. Cette architecture, que nous proposons d'appeler "TCP d'un vers beaucoup" est une extension naturelle du contrôle adaptatif de TCP au cas de plusieurs destinations. Elle permet notamment une adaptation globale bien que décentralisée de la diffusion multipoint aux conditions de congestion dans les diverses parties du réseau.

Nous montrons au moyen de modèles mathématiques, de simulations de grands réseaux et d'expérimentations sur l'Internet, que cette architecture garantit la fiabilité de bout en bout, même en cas de pannes multiples, pourvu que les mémoires de sauvegarde soient dimensionnées de manière appropriée. Nous montrons aussi que sous certaines hypothèses statistiques sur les perturbations induites par le traffic transversal décrites dans l'article, et pour toute taille des mémoires tampons, le débit de cette diffusion fiable est supérieur à une constante strictement positive indépendante de la taille du groupe. Ce résultat d'extensibilité, d'intérêt général, tranche avec les résultats négatifs sur le débit des mécanismes de diffusion fiable par routage IP multipoint.

Mots-clés : Communication multipoints, réseau applicatif, connexions TCP en série.

1 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. See for instance Narada [11], Yoid [13], ALMI [22], Host Multicast [29], NICE [5], Delauney graph [19], and [26, 27, 24].

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 [10], Overcast [15] and ROMA [17]. There is for instance an increasing interest in using TCP based multicast overlays for real time applications (see studies on multimedia streaming over TCP [14, 21]). 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, which, in this context, consists in the detection of failures and in the reconstruction of trees. This issue is addressed in [6] for the case of UDP point to point connections, where a resilient multicast architecture based on redundant transmissions and backup links is proposed. 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. 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 show that end-to-end reliability can be achieved in a fully decentralised and hence scalable way in an overlay network where nodes could both fail and overflow due to finite buffers. In the one-to-many TCP overlay architecture proposed in this paper, a different TCP connection is used from each mother node in the multicast tree to each of its daughter nodes as in RMX [10] or ROMA [17]. A specific feature of the one-to-many TCP overlay architecture is the use of a sliding-window back-pressure mechanism between each pair of adjacent TCP connections in order to prevent buffer overflow in any node. This back-pressure mechanism is implemented at the application layer and does not require any

modification of the current TCP stack. In addition, a back-up memory buffer is set in each node in order to store copies of processed packets. This buffer is used for resilience. It allows one to re-establish connection after a node failure or departure. 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. See for example [12, 8, 18] and the references therein. In particular, it was shown in [7, 25, 9] 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 first studied in [28], 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. In [2], an AIMD (Additive Increase Multiplicative Decrease) window congestion mechanism with ECN (Early Congestion Notification) was considered as the node-to-node transfer protocol together with 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.

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 3.6, 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 [28] in order to cope with larger and larger groups is not necessary to guarantee a group throughput.

In this 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 back-pressure analyzed in [2], where nodes influence each other downstream only, and where simple induction arguments based on the acyclic nature of interactions could then be made. This lack of acyclicity creates the main difficulty for the analysis of architectures using back-pressure. As we will see, a consequence of this 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 modeling difficulty comes from the fact that packets need to be delivered to a node and relayed in sequence. Hence packet losses in network links create forwarding interruptions in the destination nodes, that last until the retransmitted packet arrive.

For addressing 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 in spite of the acyclic setting described above, 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 stand-alone 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 §3. Section 4 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 §5.

2 The One-to-many TCP Overlay Architecture

2.1 Multicast Overlays

We consider the problem of reliable group communication where the same content has to be transported in an efficient way from a root node to a set of users. The broadcasting of this content is made efficient via the setting of a multicast tree where each node of the tree duplicates the packets it receives from its mother node and sends them to all its daughter nodes. 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 the models introduced in §3), 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. As illustrated in Figure 1, after receiving data from its mother node, a node will replicate the data forward it to each of its daughter nodes in the overlay tree using a distinct TCP connection for each daughter node.

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

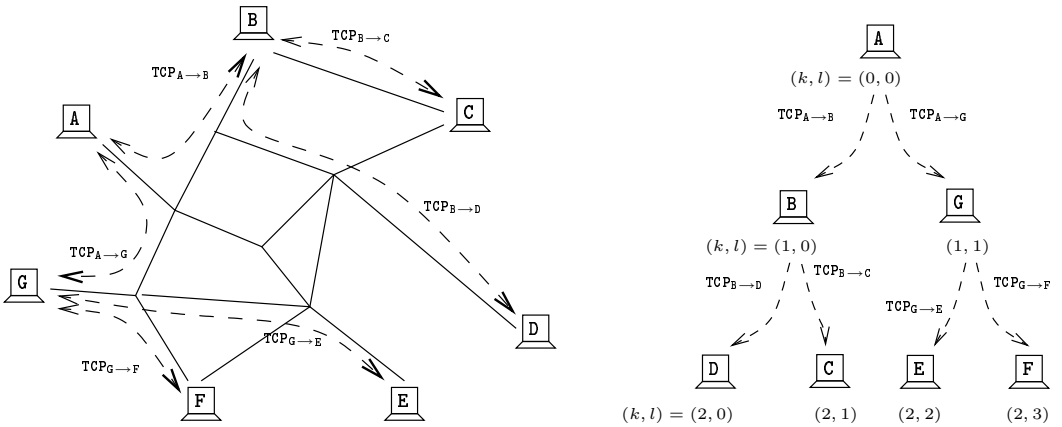

Figure 1: An overlay multicast tree : physical topology (left) and abstract topology (right)

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 2 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).

We shall assume that in all TCP connections, Fast Retransmit Fast Recovery [1] is implemented. We consider both situations with and without Selective Acknowledgment (SACK). In the scalability analysis of Section 3, we also consider the case of ECN as an intermediate step.

Tree topology will have an effect on the performance of the group (see e.g. [5]). If the depth of the tree is large, nodes deep in the tree will receive packets after a long delay. On the other hand, if the (out)degree of the tree is large, the downstream TCP connections will compete for the bandwidth of the shared links, especially that of the "last mile". In this paper, we will 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.

From management perspective, the tree topology information needs to be stored and updated by the end-systems. Each node should at least have a partial view of the tree. Different architectures could be considered and implemented. In this work, we consider a simple, but not necessarily the most efficient, structure which consists in allowing each node to know its ancestor nodes and its entire subtree.

Notation We will consider several tree topologies, for which we introduce the following generic notation: we number end-systems by a pair (k, l) designing their location in the multicast tree. The first index k gives their distance to the root of the tree (or *level*). The

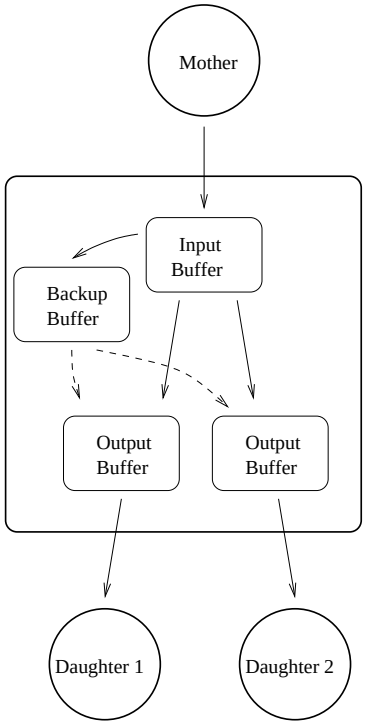

Figure 2: Different buffers in a node

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 (right). The mother node of end-system (k, l) will be denoted $(k-1, \mathtt{m}(k,l))$. The daughter nodes of end-system (k, l) will be labeled $(k+1, l')$ with $l' \in \mathtt{d}(k,l)$. For a complete binary tree, $\mathtt{m}(k,l) = \lfloor \frac{l}{2} \rfloor$ and $\mathtt{d}(k,l)$ is $\{2l, 2l+1\}$.

2.2 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); (2) losses due to input buffer overflow; (3) losses due to output buffer overflow. The first type of losses are recovered by the TCP acknowledgment and retransmit mechanisms. The second type of losses will not occur thanks to the back-pressure mechanism of TCP (RENO, New RENO or SACK). 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.

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.

2.3 Re-sequencing Delay due to Packet Losses

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 will retransmit the packet eventually. However, some packets with larger sequence number will arrive before the duplicate (of the lost packet) arrives. These packets will not be copied out to forwarding buffers until the duplicate arrives. Such a delay in packet processing will have negligible impact to the throughput of the TCP connection in question. 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 will have ripple effects in the corresponding subtrees. In turn, due to the back pressure mechanisms, these performance degradations will impact the root sending rate, and therefore the group communication throughput. Thus, it is very important to understand how the scalability is affected by the reliability of transfers.

2.4 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 node failure. The keep-alive messaging system can be established in a similar way. In this paper, we will 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 shall present some in Section 4 and study their impact in Section 5. This reconfiguration is completed by propagating the information of new subtrees up to the root, and by propagating the ancestor chain information down to the newly reconnected subtrees. This tree topology information update process is initiated by the mother node and the daughter nodes of the failed node; whereas the ancestor chain information update process is initiated by the "step-mother" nodes. 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 will first be copied to the output buffer corresponding to 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 will show in Section 4 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. More precisely, Let $B_{\texttt{OUT}}^{\max}$ and $B_{\texttt{IN}}^{\max}$ be the maximum sizes of output and input buffers, respectively. Then the backup buffers should be of size

$$B_{\texttt{BACK}} \geq m \cdot (B_{\texttt{OUT}}^{\max} + B_{\texttt{IN}}^{\max}) + B_{\texttt{OUT}}^{\max} \tag{1}$$

in order to tolerate m simultaneous failures. Under such a condition, the daughter nodes of the failed node can be re-attached to any of the nodes in the subtree rooted at the m-th generation ancestor of the failed node.

It is worthwhile noticing that this backup buffer architecture for the end-to-end reliability is very simple. In particular, it can be implemented at the application level and there is no need of searching for nodes possessing the packets with the right sequence numbers that these daughter nodes need.

2.5 Leave and Join Procedures

The leave procedure can simply be the notification of the departure to the mother node and to the daughter nodes, followed by the disconnect of the corresponding TCP sessions. The tree can then be reconfigured as in the node failure situation. When a node joins the group, it can first contact the root node which will inform the new node who should be its mother node. The new node can then establish a TCP connection with the designated mother node. In order to guarantee the end-to-end reliability in the new tree, the root should notify the new node about the constraints on the buffer sizes such that the input and output buffer

sizes do not exceed $B_{\mathtt{IN}}^{\max}$ and $B_{\mathtt{OUT}}^{\max}$, respectively, and the backup buffer size should satisfy inequality (1). These leave and join procedures are completed by the topology information update process as described previously in the node failure case.

3 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 will 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 [28], 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 ressources of each router-link in each TCP connection. On each such ressource, this competition creates additional queuing delays between the processing of successive packets of the multicast flow. We model routers as single server queue with FIFO discipline and infinite buffer. 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 [4, 9, 2] 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 will be made under a set of stochastic assumptions described below and will cover both the ECN marking and case of packet losses. We will first present the model for packet ECN marking. We address later in the section the more complex loss, retransmission and re-sequencing phenomena described in Section 2.3.

The models are described under several different but related formalisms which will be useful in what follows:

- *Random weighted graphs* and first passage percolation, which seem to be the right formalism to adequately represent all the required mechanisms in a compact way. It is instrumental in the mathematical proof of rate scalability;
- *Petri nets* can be seen as some sort of folding of the above random graphs. They give a more global and visual representation of the network topology and control mechanism; we only include them in this section for the sake of exposition.
- *(max,plus) recursions* are linked to maximal weight paths in the random graphs; this formalism turns out to be the most efficient for simulating the dynamics of this class of networks.

The elaboration of these representations within the context of this non-acyclic and loss–re-sequencing framework are one of the main technical achievements in this work.

3.1 A Petri Net-Like Representation for the ECN Marking Case

Our model, that we describe first for the loss-free ECN marking case, is represented in Figure 3 with a formalism that recall Petri net. This figure goes into detail for a binary tree with height 2, including blocking mechanism associated with window flow control and back-pressure. A detailed description of the notation used on this figure can be found below. For more details on Petri net see [3].

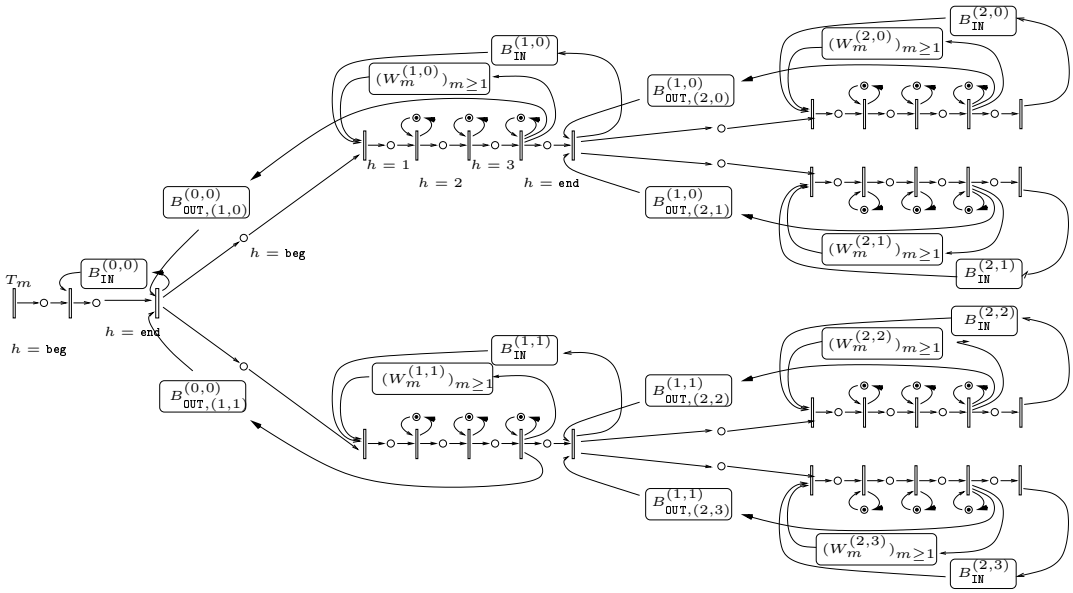

Figure 3: Example of a Binary Tree of Height 2 with Input and Output Blocking

In order to help the reader to identify the mechanisms used in end-systems, and interpret them within this Petri net-like representation, we have added (in Figure 4) a zoom of an end-system and of the information exchanged with its mother and daughter nodes.

Notation used :

- **End-systems** are labeled according to the index (k, l) presented in Section 2. For end-system (k, l) we denote by $B_{\mathtt{IN}}^{(k,l)}$ the size (measured in packets; we will 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').

- The **TCP connections** to end-system (k,l) is labeled with the index (k,l). Each TCP connection has a route that consists of a sequence of $H_{(k,l)}$ routers in series. Routers of TCP connection (k,l) are labeled by index $h = 1,2,\ldots,H_{(k,l)}$. Each router is represented as single server queue. The input buffer of router h of connection (k,l) is represented in the Figure 3 by the place with label (k,l,h) (represented by circles). The place $(k,l,\mathtt{beg})$ of TCP connection (k,l) represents the output buffer of end-system $(k-1,\mathtt{m}(p))$. Similarly, the place $(k,l,\mathtt{end})$ represents the input buffer of end-system (k,l). The notation for index h is indicated on Figure 3 for the root and the TCP connection $(1,0)$.

TCP Window flow control : Let $(W^{(k,l)}_m)_{m\geq 1}$ denote the window size sequence for TCP connection (k,l). More precisely, $W^{(k,l)}_m$ is the window size seen by packet m. This sequence takes its values in the set $\{1,2,\ldots,W_{\max}\}$, where $W_{\max}$ is the maximal window size. We will assume the following random evolution (corresponding to TCP RENO's congestion avoidance AIMD 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 $\{1,2,\ldots,W_{\max}\}$; 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 $(W^{(k,l)}_m)_{m\geq 1}$ is an aperiodic and ergodic Markov chain (see [4]).

Remark : About the window size unit. In our model, we assumed that each packet was acknowledged. In current TCP implementations, an acknowledgment is sent for every second segment. This could be taken into account by saying that a packet transmission in the model represent the transmission of two segments in the TCP connection. One should then take an "abstract packet" size of $2 \times MSS$ in the model. The variable W_m, which is an integer expressed in abstract packets, should then be equal to the integer part of $CWND/(2 \times MSS)$ where CWND is the congestion window given for the TCP protocol; it should then increase of $MSS/(2 \times MSS) = 1/2$ for each window successfully transmitted (i.e. the value of W_m is increased by 1 after the successful transmission of $2W_m$ packets).

The Token Game. Let us describe the processing of the model, represented for instance in Figure 3. Tokens can be stored in places (represented by circles) and transported to other places through transitions (represented by bars). They may represent either packets, or acknowledgments, or more generally control events associated with the routing, transport, or the back-pressure mechanisms. As in a Petri net, the general rule is that any transition takes place as soon as a token is available in each of the places upstream of this transition. One token is then consumed from each place upstream and one token is created in all places

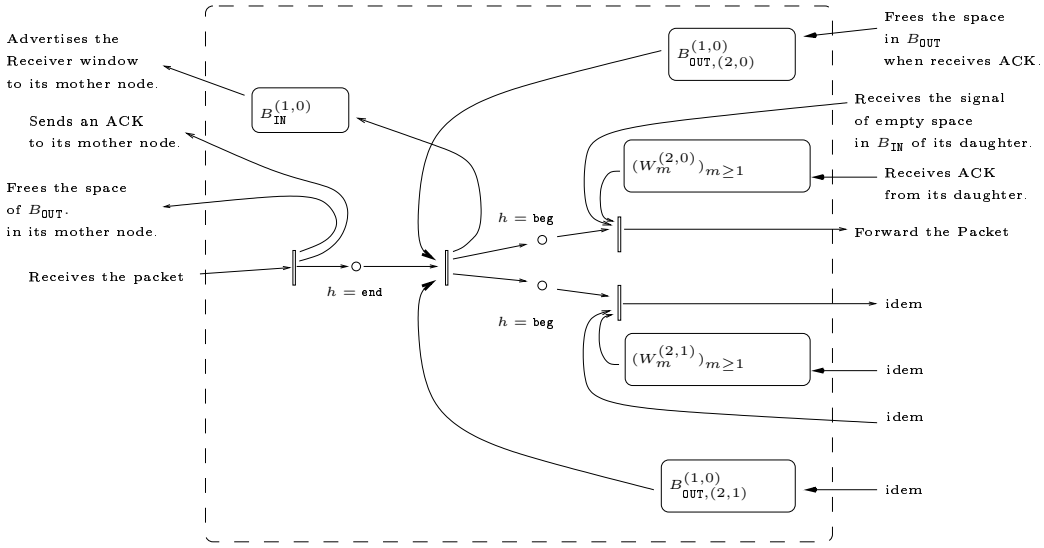

Figure 4: Description of an End-System.

downstream of the transition, after some random processing time which depends on the transition.

For instance, tokens representing data packets of the group communication are created at the root node located on the left part of the figure; they are then processed from place to place, namely from buffer to buffer, with respect to the processing rule of transitions and these transitions' random processing times. These tokens leave a place with index $h = \mathtt{beg}$ and $h = \mathtt{end}$, corresponding to a buffer in an end-system, through a transition with null processing time. The other transitions, originating from places with index $h = 1, \ldots, H_{(k,l)}$ represent the transmission of the packet through a router/link. As routers treat packets in a FIFO way, a local feedback loop with one token is attached to each of these transitions, preventing packet m to be processed before packet $m-1$ has left. The processing times in these transitions are random variables describing the impact of cross traffic, that is an additional waiting time between the processing of packets of the group communication, due to packets coming from other flows. See [2] for more details on the matter. In this framework, the random processing time of packet m through router (k, l, h) will be denoted by $\sigma_m^{(k,l,h)}$ and will be referred to as the *Aggregated Service Time* of packet m.

The initial condition of the system is that all places with an index h, that represent buffer containing data packet of the communication, are empty of tokens. This correspond to the case where initially multicast communication has not started and all memory buffer are empty in every end-system.

The feedback arcs, that go in the direction opposite to the data stream, represent the various flow control and back-pressure mechanisms. The associated places, found in the feedback loops of the system, have been expanded to boxes in Figure 3, and they have been labeled with the number of tokens that they contain (see below).

- The feedback arc with place labeled $B_{\mathtt{IN}}^{(k,l)}$ represents the advertisement of the receiver window size of end-system (k,l) back to its mother node. Since the total number of tokens in the cycle made of the place on this arc and of the places $(k,l,1)$, $(k,l,2)$, ..., $(k,l,\mathtt{end})$ is an invariant and remains equal to $B_{\mathtt{IN}}^{(k,l)}$, one sees that when the total number of in-fly packets of TCP connection (k,l) added to the number of packets in the input buffer of end-system (k,l) is equal to $B_{\mathtt{IN}}^{(k,l)}$, then the place of this feedback arc has no token left, so that the transition downstream of place $(k,l,\mathtt{beg})$ is blocked indeed.

- The feedback arc with place labeled $B_{\mathtt{OUT},(k',l')}^{(k,l)}$ represents the blocking of the stream out of the input buffer of end-system (k,l) because of a lack of space in the output buffer associated to TCP connection (k',l'). This arc stems from the transition upstream of place $(k',l',\mathtt{end})$ rather than from that downstream of place $(k',l',\mathtt{beg})$ because packets are only deleted from this output buffer when acknowledged by the receiver of TCP connection (k',l').

- The feedback arc labeled with the congestion window $(W_m^{(k,l)})_{m\geq 1}$ represents the dynamic window flow control of TCP. Notice that this arc does not behave like in a classical Petri net, as the size of the window used changes with the packet number.

3.2 Evolution Equation

In the previous section we have seen that the control mechanism implemented by TCP connections and end-system back-pressure can be represented with transitions and places that link packet transmission along the paths of the tree. Notice that this system does not behave as a classical Petri net as controls implemented by TCP cannot be modeled by a constant number of tokens in a feedback loop. However, the system that we depicted can still be exactly represented as a (max,plus) linear recursion, that we describe in detail in this section, and that will be instrumental for simulating large networks.

Let T_m, $m \geq 0$, denote the time when packet m is available at the root node. In this work we are only interested in the case of 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 transition (k,l,h) (representing a router or some other device) completes the transmission of packet $m \geq 0$. By convention $x_m^{(k,l,h)} = -\infty$ for $m < 0$.

In particular, $x_m^{(k,l,\mathtt{beg})}$ is the time when packet m departs from the output buffer of the source node of TCP connection (k,l) (that is the buffer with size $B_{\mathtt{OUT},(k,l)}^{(k-1,\mathtt{m}(k-1,l))}$), whereas

$x_m^{(k,l,\mathtt{end})}$ is the time when packet m departs from the input buffer of the receiver node of TCP connection (k,l) (that is from the buffer with size $B_{\mathtt{IN}}^{(k,l)}$).

If the network is initially empty of multicast packets (except for the root), which corresponds to an initial condition for the Petri net as depicted in Figure 3, 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 (where $\vee$ denotes the maximum) :

$$x_m^{(0,0,\mathtt{beg})} = T_m \vee x_{m-B_{\mathtt{IN}}^{(0,0)}}^{(0,0,\mathtt{end})} \tag{2}$$

$$x_m^{(0,0,\mathtt{end})} = x_m^{(0,0,\mathtt{beg})} \vee \left(\bigvee_{l \in \mathtt{d}(0,0)} x_{m-B_{\mathtt{OUT},(1,l)}^{(0,0)}}^{(1,l,H_{(1,l')})} \right) \tag{3}$$

and by the following set of equations for node (k,l), $k \geq 1,\ l \geq 0$:

$$x_m^{(k,l,\mathtt{beg})} = x_m^{(k-1,\mathtt{m}(k,l),\mathtt{end})} \vee x_{m-B_{\mathtt{IN}}^{(k,l)}}^{(k,l,\mathtt{end})} \vee x_{m-W_m^{(k,l)}}^{(k,l,H_{(k,l)})} \tag{4}$$

$$x_m^{(k,l,1)} = \left(x_m^{(k,l,\mathtt{beg})} \vee x_{m-1}^{(k,l,1)} \right) + \sigma_m^{(k,l,1)} \tag{5}$$

$$x_m^{(k,l,H_{(k,l)})} = \left(x_m^{(k,l,H_{(k,l)}-1)} \vee x_{m-1}^{(k,l,H_{(k,l)})} \right) + \sigma_m^{(k,l,H_{(k,l)})} \tag{6}$$

$$x_m^{(k,l,\mathtt{end})} = \left(x_m^{(k,l,H_{(k,l)})} \vee \left(\bigvee_{l' \in \mathtt{d}(k,l)} x_{m-B_{\mathtt{OUT},(k+1,l')}^{(k,l)}}^{(k+1,l',H_{(k+1,l')})} \right) \right). \tag{7}$$

These (max,plus) recursion equations turn out to be very handy to compute in a very efficient way transmission of packets in large network. They are used in the simulations presented in Section 5.1.

3.3 Path of Maximal Weight in a Random Graph

Consider the random graph G where the set of vertices is:

$$\begin{aligned} V &= \{(0,0,\mathtt{beg},m),(0,0,\mathtt{end},m),\ m \in \mathbb{Z}\} \\ &\cup \{(k,l,h,m),\ k \geq 1,\ l \geq 0,\ h \in \{\mathtt{beg},1,2,\ldots,H_{(k,l)},\mathtt{end}\},\ m \in \mathbb{Z}\} \end{aligned}$$

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

$$\begin{aligned}
E_1 &= \{(0,0,\mathtt{end},m) \to (0,0,\mathtt{beg},m),\ \forall m \in \mathbb{Z}\} \\
&\cup \left\{(k,l,1,m) \to (k,l,\mathtt{beg},m),\ (k,l,\mathtt{end},m) \to (k,l,H_{(k,l)},m),\ \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\right\} \\
&\cup \left\{(k,l,h,m) \to (k,l,h-1,m), \quad \text{for } h = 2,\ldots,H_{(k,l)} \text{ and } \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\right\} \\
&\cup \{(k,l,\mathtt{beg},m) \to (k-1,\mathtt{m}(k,l),\mathtt{end},m),\ \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\}\ , \\
E_2 &= \{(k,l,h,m) \to (k,l,h,m-1), \quad \text{for all values of } h \text{ and } \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\} \\
E_3 &= \left\{(k,l,\mathtt{beg},m) \to (k,l,H_{(k,l)},m - W_m^{(k,l)}),\ \forall k \geq 1, l \geq 0, m \in \mathbb{Z}\right\} \\
E_4 &= \left\{(k,l,\mathtt{beg},m) \to (k,l,\mathtt{end},m - B_{\mathtt{IN}}^{(k,l)}),\ \forall k \geq 0, l \geq 0, m \in \mathbb{Z}\right\} \\
E_5 &= \left\{(k,l,\mathtt{end},m) \to (k+1,l',H,m - B_{\mathtt{OUT},(k+1,l')}^{(k,l)}),\ \forall l' \in \mathtt{d}(k,l)\right. \\
&\qquad\qquad \left. \text{and } \forall k \geq 0, l \geq 0, m \in \mathbb{Z}\right\}.
\end{aligned}$$

The most efficient way for depicting G is to concentrate on the case of TCP connections in series rather than on a tree. This is done in Figure 5, for a line of two connections, where the E_1 edges are the horizontal ones and the E_2 edges are the vertical ones. The other edges are marked on the figure.

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

Initial condition : we consider the situation where the system starts to transmit from packet $m = 0$, with all buffers in end-systems empty. For representing this initial condition, we will define 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 (2)-(7) that for all k,l,h,m

$$x_m^{(k,l,h)} = \max_{\substack{\pi \text{ a path in } G^{[0]} \\ (k,l,h,m) \rightsquigarrow (0,0,\mathtt{beg},0)}} \{\mathtt{Weight}(\pi)\}. \tag{8}$$

In what follows, we assume that the out-degree (or fan-out) of the tree is bounded by a constant D. Under this assumption, as shown in the Section 3.6,

- any vertex in G has a number of neighbors bounded by a constant;
- the length $\mathtt{Weight}(\pi)$ of a path π, leading from (k,l,h,m) to $(0,0,\mathtt{beg},0)$ in G can be bounded by a constant multiplied by $k+h+m$.

These two constants do not depend on the size and topology of the tree.

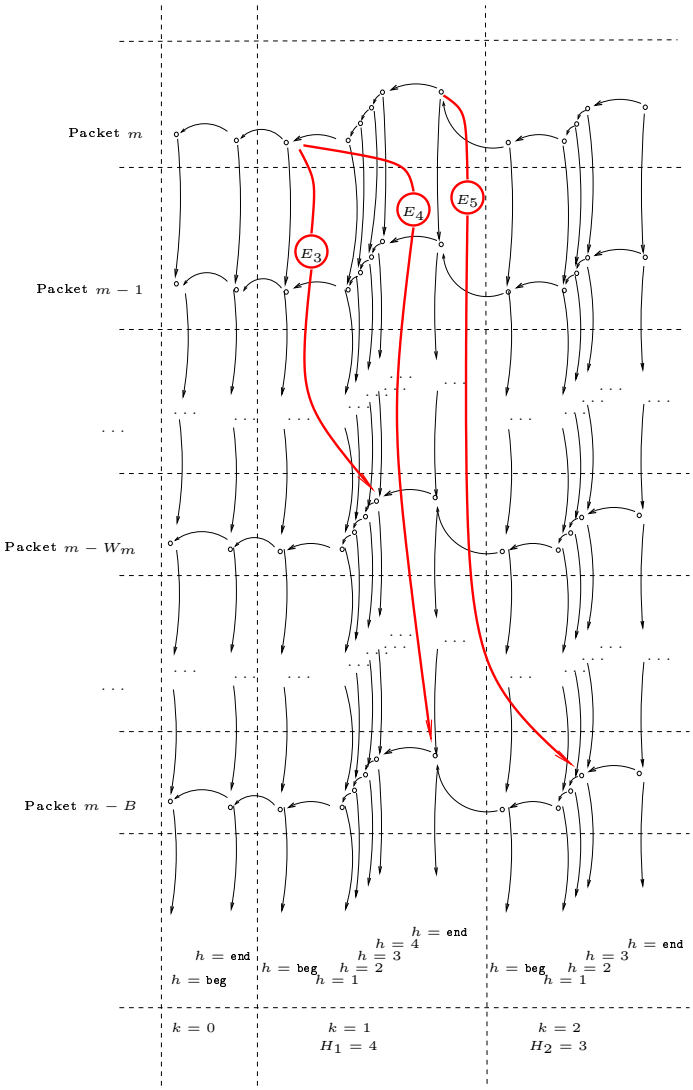

Figure 5: Random Graph Representing two TCP Connections in tandem with Back-Pressure Constraints

3.4 The Loss and Re-sequencing Model

In this section we demonstrate that the framework that we introduced can be used to model additional packet-level aspect of Overlay Multicast using TCP. The self-clocking model of

the TCP mechanism will remain the same here as in the ECN packet marking case. However, when a packet loss happens, retransmission packets are added as new branches of the random graph that have potential effects on later packets. In addition to that, new constraints are added between two consecutive connections for each packet loss to represent the additional delay needed to reorder the packets. The extended model that we obtain can still be written in the same framework, using maximal weighted paths in random graphs.

For the sake of clear exposition, we first study the *nominal case*, where only one connection is considered in isolation and with sufficient buffer to receive all the data (so that it won't be influenced by back-pressure). In this case, packets are sent by the TCP sources as soon as the congestion window allows it (when the packet to be send is at most the highest sequence number acknowledged plus CWND).

3.4.1 Fast Retransmit Fast Recovery in the Nominal Case

Let us start with the case where one packet (with index m) is lost and no other neighboring packets is lost.

The departure of packet m is triggered by the ACK of packet $m - W_m$. After that, the ACKs for packets $m - W_m + 1, \ldots, m - 1$ are received; they trigger the departure of packets $m + 1$, $m + 2$ up to $m + W_m$ (since the window necessarily increases of one unit on this interval and is hence equal to $W_m + 1$ when packet $m + W_m$ is emitted). Packet m is lost but packets $m + 1, m + 2 \ldots$ are received; they trigger duplicate ACKS to be sent to the source. When the third duplicate ACK is received by the source (corresponding to the arrival of packet $m + 3$), it starts the Fast Retransmit Fast Recovery Procedure : Retransmission of packets $m, m + 1, \ldots, m + W_m$ are sent, the current CWND is halved, and inflated by three units. Most of the time (except for the case where $W_m \leq 3$) the emission of packet is then stopped, as the highest sequence number received is $m - 1$ and CWND was reduced to $(W_m + 1)/2 + 3$. New packets, that were already sent, $m + 4, m + 5, \ldots$ are then received; each of them sends a new duplicate ACK back to the source, which inflates CWND of one unit. Hence, packet $m + W_m + k$ is emitted when duplicate ACK corresponding to packet $m + (W_m + 1)/2 + k$ is received, as if the window observed by it is $(W_m - 1)/2$. This phase ends when retransmitted packet m arrived immediately after packet $m + W_m$ (which had triggered packet $m + W_m + (W_m - 1)/2$). The normal increase evolution of the window is then resumed, with CWND equal to $(W_m + 1)/2$ and the highest acknowledged sequence number being $m + W_m$. Packet $m + W_m + (W_m + 1)/2$ is thus immediately sent.

To summarize, for packet $m + 1, \ldots, m + W_m$, the window is evolving naturally with the additive increase. Then it is $\max((W_m - 1)/2, 1)$ for packets $m + W_m + 1, \ldots, m + W_m + (W_m - 1)/2$. Additive increase is then resumed for packet $m + W_m + (W_m + 1)/2$ with window initially set to $(W_m + 1)/2$

3.4.2 A Conservative Simplified Model for the General Case

The representation of the loss of a packet in the non nominal case is more complex as some of the packets $m + 1, m + 2, \ldots, m + W_m$ might not have left the source when the loss of

m is detected: the emission of these packets is allowed by the congestion window, but other constraints (back pressure, packet not available from previous node) might have delayed them.

Our aim below is not to build an exact model for the case with losses and re-sequencing, but rather to describe a simplified and tractable model obtained via a set of conservative transformations. For proving the scalability of the case with losses and re-sequencing (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.

Let m', $m \leq m' \leq m + W_m$, be the index of the last packet emitted after m and before the loss was detected. The window evolution for packets $m, m+1, \ldots, m'$ follows a normal additive increase; it is then kept constant to $(W_{m'} - 1)/2 \geq (W_m - 1)/2$ until the retransmission of packet m is received (by definition just after packet m' arrives). When this happen, the latest packet that could possibly be sent is $m' + \max((W_m - 1)/2, 1) \leq m + W_m + \max((W_m - 1)/2, 1)$.

The following simplified window evolution

- the window is set to $\max((W_m - 1)/2, 1)$ for $m+1, m+2, \ldots, ..., m+W_m+\max((W_m - 1)/2, 1) - 1$;
- 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 will have larger windows at all times and hence better throughput than the considered simplified model.

3.4.3 Retransmission, Re-sequencing

We have chosen to include in general the retransmitted packets at the last possible step of the communication (between $m + W_m$ and $m + W_m + 1$, as in the nominal case). This tends to overload the network at intuitively the worst time (after the self clocking mechanism have resumed with a half window).

In the case where SACK is implemented by the TCP connection, the simplified model described here is still conservative compared to the real window evolution. Only the lost packet is retransmitted, instead of an entire window.

Packets received by the destination end-system under consideration have to be forwarded to its daughter nodes according to the order defined by the sequence number. When packet m is lost, as described above, packets $m + 1, \ldots, m'$ are blocked in the input buffer of the receiving end-system. They are released as soon as retransmission of packet m, sent between packets m' and $m' + 1$, is received. Again, as the exact value of m' is not easy to evaluate, we will make a conservative choice, and assume that when packet m is lost, it has to wait for the arrival of the latest possible packet (i.e. $m + W_m$), plus the retransmission. This in particular implies that packets $m+1, ..., m+W_m-1$ also incur this additional re-sequencing constraint.

3.4.4 Random Graph Model

In the random graph associated with the loss model, we denote the vertex for end-system (k, l), packet m and index h by $\mathtt{v}(k, l, h, m)$. For all $k \geq 1$ l, h and m, we add a vertex $\mathtt{v}'(k, l, h, m)$ on top of $\mathtt{v}(k, l, h, m)$, which represents the potential retransmission of a packet just between packets m and $m + 1$. Consequently, 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 $m + 1$.

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

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

Note that with no further additional edges, these complementary vertices play no role.

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

- Edge E_6: $\mathtt{v}(k, l, \mathtt{end}, m) \rightarrow \mathtt{v}'(k, l, H_{k,l}, m + W_m - 1)$ in order to represent the resequencing of packets $m, m + 1, \ldots, m + W_m - 1$.
- Edges E_7: $\mathtt{v}(k, l, h, m''+1) \rightarrow \mathtt{v}'(k, l, h, m'')$ for all $h = 1, \ldots, H_{k,l}$ and $m'' = m, \ldots, m+ W_m$ 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 $E_1, \ldots, E_7$) is presented in Figure 6 in the case of two TCP connections in tandem. 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; we have also represented the case $B_{\mathtt{IN}} = B_{\mathtt{OUT}} = B$.

3.5 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 some 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 [2],

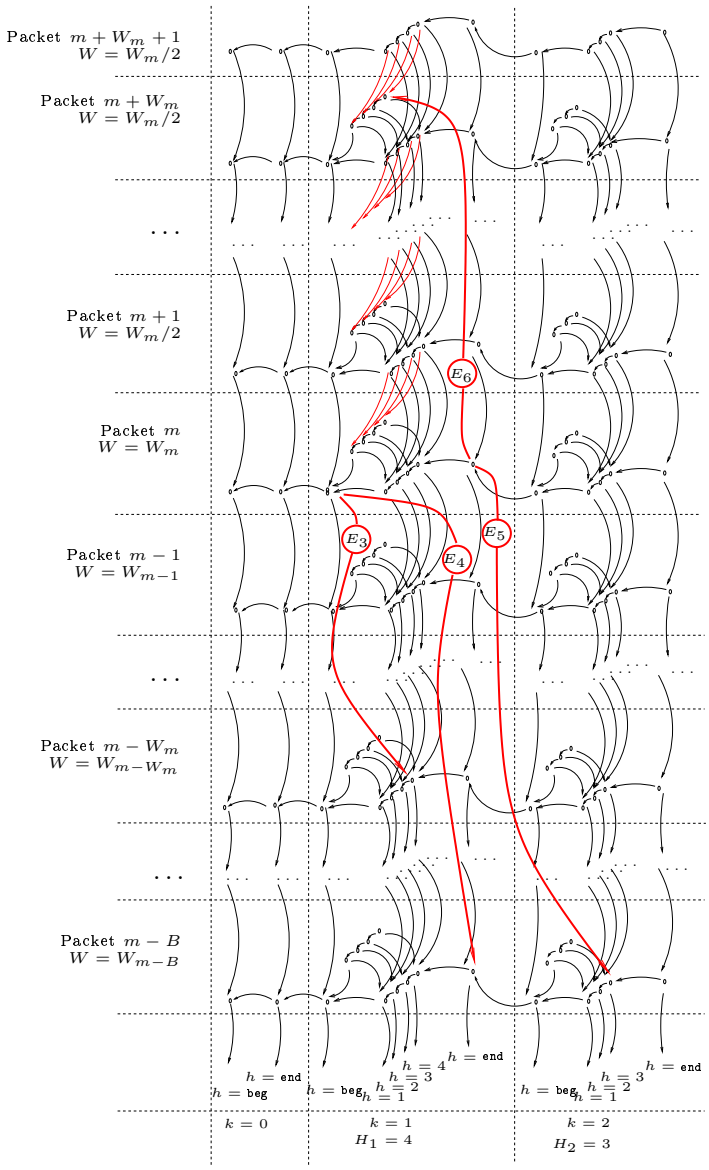

Figure 6: Random Graph Representing two TCP Connections in tandem with Back-Pressure and Retransmission Re-sequencing Constraints

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.

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.

3.6 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 will directly consider possibly infinite trees.

We call *homogeneous* model the case where:

- The tree has a fixed degree D;
- All TCP connections are structurally and statistically equivalent: the number of hops is the same in all connections; the packet ECN marking or loss process is independent and identically distributed in all connections, with packet ECN marking or loss probability p; aggregated service times are independent and identically distributed in all routers, with law σ with finite mean;
- All back-pressure buffers have the same size everywhere in the tree.

We call *non-homogeneous* model that where:

- The fan out degree in the multicast tree (described by indexes (k, l)) is bounded from above by a constant D;
- The numbers of hops of all routes are bounded from above by a constant, that is $H_{k,l} \leq H$ for all (k, l).
- The packet loss probability in TCP connection (k, l) is bounded from above by a constant p;
- The parameters $B_{\mathtt{IN}}^{(k,l)}$ and $B_{\mathtt{OUT},(k',l')}^{(k,l)}$ are respectively bounded from below by constants $B_{\mathtt{IN}}$ and $B_{\mathtt{OUT}}$ that do not depend on (k, l) and (k', l');
- The aggregated service times are independent and upper bounded (in the strong ordering sense) from above by a random variable σ with finite mean.

3.6.1 Definition of Group Throughput

In the homogeneous case, there exists a constant $\gamma \in \mathbb{R} \cup \{+\infty\}$ such that the a.s. limit

$$\lim_{m \to \infty} \frac{m}{x_m^{(k,l,h)}} = \gamma \tag{9}$$

holds for any (k, l, h). We call γ the *group communication throughput.*

Proof: Let G be the graph defined in the earlier sections: the weight of vertex (k,l,h,m) is defined for each $m \in \mathbb{Z}$ as well as the window process $\{W_m^{(k,l)}\}_m$. The aggregated service times, and the losses which create some of the edges in G are all i.i.d. in the index of packet m. We further assume that for each TCP connection, the window process $\{W_m^{(k,l)}\}_m$ is in its stationary state. Each of these processes is an ergodic (irreducible, aperiodic and positive recurrent) Markov chain, which is mixing. The product of these independent and mixing Markov chains is mixing too. We define θ to be the shift of index m in the random graph G. For instance $\sigma_m^{(k,l,h)}(\theta(\omega)) = \sigma_{m+1}^{(k,l,h)}(\omega)$, $W_m^{(k,l)}(\theta(\omega)) = W_{m+1}^{(k,l)}(\omega)$ and similarly for the other variables associated with edges. The law of G is invariant by this shift, and this shift is ergodic (it is even mixing).

For all $n \in \mathbb{Z}$, let $G^{[n]}$ be the graph obtained from G when replacing the weights of all vertices (k,l,h,m) with $m < n$ by $-\infty$. Note that this definition extends that of $G^{[0]}$ in Section 3.3.

We now introduce the sequence :

$$\xi_{m,m'} = \max_{\substack{\pi \text{ a path in } G^{[m]} \\ (0,0,\mathtt{beg},m') \rightsquigarrow (0,0,\mathtt{beg},m)}} \{\mathtt{Weight}(\pi)\}, \quad m < m'.$$

For all $m < n < m'$, $\xi_{m,n} + \xi_{n,m'}$ can be rewritten as

$$\max_{\substack{\pi \text{ a path in } G^{[m]} \\ (0,0,\mathtt{beg},n) \rightsquigarrow (0,0,\mathtt{beg},m)}} \{\mathtt{Weight}(\pi)\} + \max_{\substack{\pi \text{ a path in } G^{[n]} \\ (0,0,\mathtt{beg},m') \rightsquigarrow (0,0,\mathtt{beg},n)}} \{\mathtt{Weight}(\pi)\} .$$

As $n > m$, any vertex in $G^{[n]}$ has a smaller weight than the corresponding vertex in $G^{[m]}$, so that the above sum is smaller than

$$\max_{\substack{\pi \text{ a path in } G^{[m]} \\ (0,0,\mathtt{beg},n) \rightsquigarrow (0,0,\mathtt{beg},m)}} \{\mathtt{Weight}(\pi)\} + \max_{\substack{\pi \text{ a path in } G^{[m]} \\ (0,0,\mathtt{beg},m') \rightsquigarrow (0,0,\mathtt{beg},n)}} \{\mathtt{Weight}(\pi)\} ,$$

where $G^{[n]}$ is replaced by $G^{[m]}$ in the second term. This last sum is also the maximal weight over all paths in $G^{[m]}$ leading from $(0,0,\mathtt{beg},m')$ to $(0,0,\mathtt{beg},m)$ and that go through $(0,0,\mathtt{beg},n)$ (this is true because $\sigma_m^{(0,0,\mathtt{beg})} = 0$ for all m) and hence it is smaller than $\xi_{m,m'}$. Hence,

$$\xi_{m,m'} \geq \xi_{m,n} + \xi_{n,m'}.$$

Since in addition, $\xi_{m,m+k} = \xi_{0,k}(\theta^m)$, it follows from Kingman subadditive (actually super-additive) ergodic theorem (see [16]) that there is an almost sure limit

$$\lim_{m\to\infty} \frac{1}{m}\xi_{0,m} = \xi .$$

Since θ is ergodic, ξ is constant. We define $\gamma = \frac{1}{\xi}$ and thus we have :

$$\lim_{m\to\infty} \frac{m}{x^{(0,0,\mathtt{beg})}} = \gamma \quad \text{a.s..}$$

One deduces from the following comparison :

$$x_m^{(0,0,\mathtt{beg})} \leq x_m^{(k,l,h)} \leq x_{m+B_{\mathtt{IN}}+k\times(B_{\mathtt{IN}}+B_{\mathtt{OUT}})}^{(0,0,\mathtt{beg})}.$$

that the same limit holds for any (k,l,h). We can show the first inequality as any path in G from $(0,0,\mathtt{beg},m)$ to $(0,0,\mathtt{beg},0)$ can be easily completed into a path from (k,l,h,m) to $(0,0,\mathtt{beg},0)$. The second inequality comes by completing a path from (k,l,h,m) to $(0,0,\mathtt{beg},0)$ with the arc

$$(0,0,\mathtt{beg},m+B_{\mathtt{IN}}+k\times(B_{\mathtt{IN}}+B_{\mathtt{OUT}})) \rightsquigarrow (0,0,\mathtt{end},m+k\times(B_{\mathtt{IN}}+B_{\mathtt{OUT}}))$$

and then using that for any (k',l') and daughter node $(k'+1,l'')$ there exists a path :

$$(k',l',0,\mathtt{end},n) \rightsquigarrow (k'+1,l'',\mathtt{end},n-(B_{\mathtt{IN}}+B_{\mathtt{OUT}})) \ .$$

□

The number γ depends on the size and the topology of the tree, on the number of routers in each overlays, on the evolution of the window, and on the law of the aggregated service times modeling cross traffic, on the loss process in the TCP connections, and on the parameters of the back-pressure mechanisms. It will be called the *asymptotic group throughput* of the multicast overlay.

For the non-homogeneous case,

$$\liminf_{m\to\infty} \frac{m}{x_m^{(k,l,h)}} \geq \gamma, \tag{10}$$

where again is deterministic and is independent of (k,l,h). In this case, γ is the asymptotic group throughput of some lower bound system which should be clear.

3.6.2 Scalability under Light Tailed Assumptions

The assumptions of this section are the non-homogeneous ones of the last section. We use path enumeration for studying throughput in the case when the random variable σ is light tailed, i.e. such that there exists a real number $\tau > 0$ with

$$\mathbb{E}[e^{t\sigma}] \leq A(t) < +\infty, \tag{11}$$

for all $0 \leq t \leq \tau$.

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*

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

uniformly in (k, l), both for the ECN and the loss & re-sequencing 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.

Proof: The random variable $x_m^{(k,l,\texttt{end})}$ is the weight of the maximum weight path from $(k, l, \texttt{end}, m)$ to $(0, 0, \texttt{beg}, 0)$ in the random graph previously introduced (for ECN packet marking in Section 3.3, for packet losses in Section 3.4).

Let us first treat the case of ECN packet marking. We introduce the function ϕ :

$$\begin{array}{rcl} \phi(k,l,h,m) & = & (H+2)k + 2(H+2)m + v(h) \\ \text{with } \ v(h) & = & \left\{ \begin{array}{ll} 0 & \text{for } h = \texttt{beg} \\ h & \text{for any } 1 \leq h \leq H \\ H+1 & \text{for } h = \texttt{end} \end{array} \right. \end{array} .$$

It takes integer values and strictly decreases along any possible path in the random graph.

As a consequence, a path in this graph from (k, l, h, m) to $(0, 0, \texttt{beg}, 0)$ cannot come back to the same vertex, and also the set of vertices contained in such a path cannot exceed $(H + 2)k + 2(H + 2)m + H + 1$.

As the maximal number of neighbors of a node in the graph is $\max(3, D+1)$, the number of possible paths from (k, l, h, m) to $(0, 0, \texttt{beg}, 0)$ is bounded by :

$$(\max(3, D + 1))^{(H+2)k+2(H+2)m+H+1}.$$

Hence, using Markov's inequality, we get that for all π as described :

$$\begin{array}{rcl} P\left(\texttt{Weight}(\pi) \geq xm\right) & \leq & e^{-txm}\mathbb{E}[e^{t\texttt{Weight}(\pi)}] \\ & \leq & e^{-txm}A(t)^{(H+2)k+2(H+2)m+H+1} \end{array}$$

And using that the probability of some union of events is upper bounded by the sum of the probabilities, we see that this implies that for $m \geq k \geq 1$ and for $D \geq 2$,

$$P\left(x_m^{(k,l,\texttt{end})} \geq xm\right) \leq (D + 1)^{5(H+2)m} e^{-txm} A(t)^{5(H+2)m}.$$

If x is chosen large enough (in fact such that $e^{tX} \geq (A(t)(2D + 1))^{5(H+2)}$), the series

$$\sum_m P\left(x_m^{(k,l,\texttt{end})} \geq xm\right)$$

converges, so that from the Borel-Cantelli lemma, $P(\limsup_{m\to\infty} x^{(k,l,\texttt{end})}/m \leq x) = 1$, proving the result.

It is not difficult to see that the same argument holds for packet losses, in the random graph representing retransmissions and consecutive reordering. The function ϕ is now given

by :
$$\begin{aligned} \phi(k,l,h,m) &= W_{\max}(H+2)\,k + (H+2)m + v(h), \\ \text{with } v(h) &= \begin{cases} 0 & \text{for } h = \texttt{beg} \\ h & \text{for any } 1 \leq h \leq H \\ H+1+(H+2)(W_{\max}-1) & \text{for } h = \texttt{end} \end{cases}, \end{aligned}$$

where $W_{\max}$ is the maximum window size (note that TCP imposes that $W_{\max} \leq B_{\texttt{IN}}$).

Again, this function takes integer values and it decreases strictly along any path in the random graph and the ideas described above can then easily be extended to this case. The presence of one or several additional vertices $v'(k,l,h,m)$ in the paths is hardly an issue, as their number between two regular vertices is bounded. □

3.6.3 Scalability under Heavy Tailed Assumptions

Proving the same scalability result under heavy tailed assumptions requires to go deeper in the analysis of the properties of the random graph G.

The maximal weight of a path has been studied previously for regular graphs (lattice with any finite dimension). It was shown in particular that the maximal weight of a path in such a lattice is increasing at most linearly in function of its size provided the weights of the nodes are random variables with finite moments of order $2+\epsilon$ for some $\epsilon > 0$.

More precisely, Lemma 5.4 in [20] can be used almost directly to extend Theorem 1 to the case of a chain topology (i.e. TCP connections in series) and a constant buffer size, under the assumption $\mathbb{E}[\sigma^{2+\epsilon}] < +\infty$ for $\epsilon > 0$. For this, it is enough to embed a path of G into a greedy lattice animal with a size that is bounded by a linear function of the size of the path.

Whether this property remains true for a general tree topology it still an open question at this stage that we will study in future work.

4 End-to-End Reliability

In this section we investigate into the end-to-end reliability issue. As we mentioned in Section 3, 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. Throughout this paper we shall assume that the root node never fails.

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 2 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.

4.1 Guarantee of the End-to-End Reliability

Definition of End-to-End Reliability. We define overlay multicast system to be *end-to-end reliable with tolerance to m failures*, if after removing *simultaneously* m 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 m 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 m failures, it is not guaranteed to recover correctly from any additional failures. However if l, for some $1 \leq l \leq m$, failures occur, the system will be able to recover from additional $(m-l)$ 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 $B^{\max}_{\mathtt{OUT}}$ and $B^{\max}_{\mathtt{IN}}$ be the maximum sizes of output and input buffers in the system, respectively. A *backup buffer of order r* has size $(r \cdot (B^{\max}_{\mathtt{OUT}} + B^{\max}_{\mathtt{IN}}) + B^{\max}_{\mathtt{OUT}})$.

Failure Recovery Algorithm. We use the following simple algorithm to recover from failures. 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') retransmits all packets contained in its backup buffer. Then it continues the transmission, reading from input buffer and writing to output buffer. 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.

Theorem 2 *An overlay multicast system with backup buffer of size*

$$(m \cdot (B^{\max}_{OUT} + B^{\max}_{IN}) + B^{\max}_{OUT})$$

is end-to-end reliable with tolerance to m failures.

Proof: To estimate required backup buffer size, we first consider a chain of nodes $(k_1, l_1) \rightarrow (k_2, l_2) \rightarrow (k_3, l_3)$. Let $W^{(k_{i+1}, l_{i+1})}$ be the size of the receiver window on the TCP Connection (k_{i+1}, l_{i+1}), for i=1, 2. Suppose that node (k_2, l_2) fails. When failure is detected, node (k_3, l_3) will connect to node (k_1, l_1) and request it to re-send packets starting from packet number t+1, where t is the number of the last packet that node (k_3, l_3) received. The number of packets stored in input and output buffers at node (k_2, l_2), plus the number of packets 'in-fly' to and from node (k_2, l_2), is at most $(B_{\mathtt{OUT}}^{\max} + B_{\mathtt{IN}}^{\max})$. This bound is guaranteed by TCP's choice of receiver window size: at most $W^{(k_2, l_2)}$ packets will be 'in-fly' to node (k_2, l_2), and $W^{(k_2, l_2)}$ does not exceed the amount of free memory in the input buffer of node (k_2, l_2). Similarly, at most $W^{(k_3, l_3)}$ packets will be 'in-fly' to node (k_3, l_3), but they are not removed from the output buffer of node (k_2, l_2), until (k_3, l_3) acknowledges that it has received the packets. Therefore the difference between the smallest packet number at node (k_1, l_1) and the highest packet number at node (k_3, l_3) does not exceed the sum of buffer sizes at node (k_2, l_2). During re-transmission the application at node (k_1, l_1) does not have access to the output socket buffer, and may need to re-transmit the contents of this buffer as well. Hence the total number of packets that need to be re-transmitted is bounded by $B_{\mathtt{OUT}}^{\max} + (B_{\mathtt{OUT}}^{\max} + B_{\mathtt{IN}}^{\max})$, which is the size of an order 1 backup buffer.

If (k_2, l_2) has more than one daughter node, each of the daughter nodes will require at most $B_{\mathtt{OUT}}^{\max} + (B_{\mathtt{OUT}}^{\max} + B_{\mathtt{IN}}^{\max})$ packets to be re-transmitted, and the same backup buffer of order 1 will provide all necessary packets.

If more than one failure occurs, and there is more than one failing node on the path from disconnected node (k, l) to it's surviving ancestor node (k', l'), the surviving ancestor node will need to re-transmit the contents of input and output buffers at all failing nodes on the path, plus the contents of output buffer at (k', l'). Since the number of failing nodes is bounded by m, we have proven the theorem. □

Note that in our definition of tolerance of failures we used standard notion in the fault tolerance literature. The proof of Theorem 2 actually proves a much stronger result, which we state it as a corollary here:

Corollary 1 *An overlay multicast system with backup buffer of size*

$$(m \cdot (B_{\mathit{OUT}}^{\max} + B_{\mathit{IN}}^{\max}) + B_{\mathit{OUT}}^{\max})$$

is end-to-end reliable with tolerance to m simultaneous and consecutive failures in a chain of the tree.

4.2 Handling Leave and Join and Restoring Connectivity

In practice a multicast system should allow nodes to leave and join the group during transmission. Leaving nodes can be handled by the failure recovery scheme. Several different strategies can be used for joining the group. A node joining the transmission may want to connect to a distant leaf node, which is processing packets of the smallest sequence numbers, so that the newly joined node can capture the most of transmitted data. However, if delay

is an important factor, a joining node will try to connect to a node as close to the root as possible. In practice, the maximum number of down-links for each node is limited, due in particular to the last-mile effect, and not every node in the multicast group can accept new connections. Therefore the up-link node for new connection is chosen among "active" nodes, which have not yet exhausted their capacity.

The procedure for restoring connectivity after a failure is similar to the join procedure, but in this case the choice of nodes is further limited to the subtree of the surviving ancestor. In applications where communication delay is to be minimized, the goal is to maintain a tree as balanced as possible, subject to the degree constraint. We propose to use a greedy heuristic to restore connectivity. Our heuristic tries to minimize overall tree depth, subject to the degree constraint, by reconnecting longest subtrees to nodes that are as close to the root as possible. The algorithm description below is given for the case of one node failure, but the case of several failures can be handled as a sequence of single failures.

Algorithm GREEDY_RECONNECT

1. Suppose node (k, l) fails. Let $\mathcal{S}$ be the set of orphaned subtrees, rooted at daughters of (k, l). Let $\mathcal{A}$ be the set of active nodes in subtree of $(k-1, \mathtt{m}(k, l))$, but not in the subtree of (k, l).
2. Choose a node $(k+1, l') \in \mathcal{S}$ that has subtree of largest depth.
3. Choose a node $(p, q) \in \mathcal{A}$ that is closest to the root.
4. Connect $(k+1, l')$ to (p, q).
5. Update $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(k-1, l')\}$ and add active nodes from subtree of $(k+1, l')$ to A.
6. If $\mathcal{S}$ is not empty, go to Step 2.

Depending on the objective function, other approaches can be considered. For example, if throughput is to be maximized, and last-mile links have limited bandwidth, then lower fanout allows to achieve higher throughput, and the optimal topology could be a chain. On the other hand, if delay is to be minimized, the optimal configuration would likely be the star where all the nodes have direct connections with the root node. Finally, if no specific goals are set, a random choice of uplink node (still subject to fanout constraints) is feasible.

5 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. More precisely, we have developed an equation-based simulator which is particularly efficient for the simulation of large trees. We also prototyped our reliable multicast architecture and carried out experiments in the Planet-Lab environment [23]. Finally, we developed a discrete-event simulator to simulate the dynamics of the tree when there are leave/join and failures. In this section, we report and comment on these empirical results.

5.1 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 developed based on the evolution equations of §3.2. The main advantage of this equation-based simulator compared to the traditional discrete-event simulators is that it allows one to handle much larger trees, which is a key issue in the scalability analysis.

The simulation setting and assumptions are summarized as follows:

- **Packet Size and Simulation Length:** All performance results are given in packets, that is the equivalent of two MSS. For reference and compatibility with our PlanetLab experiment we will assume that MSS=100B, so that a packet is 200B (see the discussion at the end of §3.1). In each simulation run, we simulate the transmissions of 10M packets (equivalent to 2GB of data).

- **Tree Topology:** We report results only on the case of balanced binary tree. The end-systems as well as the network connections are homogeneous (see below).

- **TCP Connections:** We consider the homogeneous case, where each connection goes through 10 routers in series. All the packets transmitted on this connection have an independent probability p to get a negative feedback (loss or ECN marking). The default option is $p = 0.01$. We do not include timeouts occurring due to large delay variations in a TCP connection.

- **Network Load:** As is described in § 3, the cross traffic is characterized by the Aggregated Service Times in each router. In these simulations we have considered both exponential and Pareto random values with mean equal to 10ms for each router/link (this incorporates propagation delays as well as queuing delays due to cross traffic). The default option is exponential.

- **Buffer Sizes:** The same experiments were repeated for different values of the buffer sizes. In this paper we only report results for the cases where $B_{\mathtt{IN}}$ is set to 50 packets (i.e. 10KB), and $B_{\mathtt{OUT}}$ varies as 50,100,1000 and 10 000 packets (resp. 10KB, 20KB, 200KB, 2MB). Note that in this scalability analysis, the size of the backup buffer does not have any effect. Thus, for these experiments we have $W_{\max} = \min(B_{\mathtt{IN}}, B_{\mathtt{OUT}}) = 50$ packets.

Throughput Scalability. We have simulated complete binary trees of sizes upto 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 7 is 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 ECN has slightly better throughput with about 2% improvement over TCP SACK.

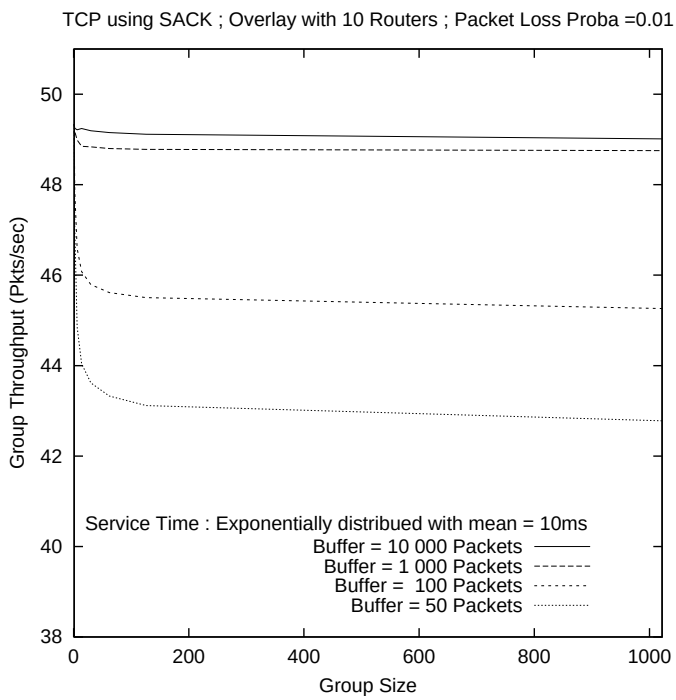

Figure 7: Group throughput as a function of group size, with TCP SACK, exponential cross traffic, and different output buffer sizes.

Comparison between Asymptotic Throughput and Single Connection Throughput. In [2] 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 Table 1 we report the ratio of these two quantities. 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%).

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

Table 1: Ratio of Asymptotic Throughput / One-Connection Throughput

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 Traffic. In our work, cross traffic at the routers is modeled through the aggregated service times. We have shown the scalability of the group throughput under the assumption of light tail of the aggregated service times. The validity of Theorem 1 under heavy tail distribution assumption remains an open question. We now use simulation to see what is the impact of this distribution, in particular, when it is heavy tailed. In Figure 8, we show the 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 the distribution, the smaller the throughput is.

We also observe that even for heavy tail distributions like Pareto, when the second moment exists (which is the case when the parameter is 2.1), the throughput curve has a shape similar to that of the exponential distribution. However, when the parameter is 1.9, the second moment no longer exists, the throughput curve tends to decay faster. This suggests that the light tail distribution assumption could be relaxed and replaced by some moment conditions. Indeed, in the case of chain (a special case of tree), it can be shown, by applying a result of [20], when the aggregated service times have a moment that is strictly higher than the second moment, then the group throughput is lower bounded by a strictly positive constant.

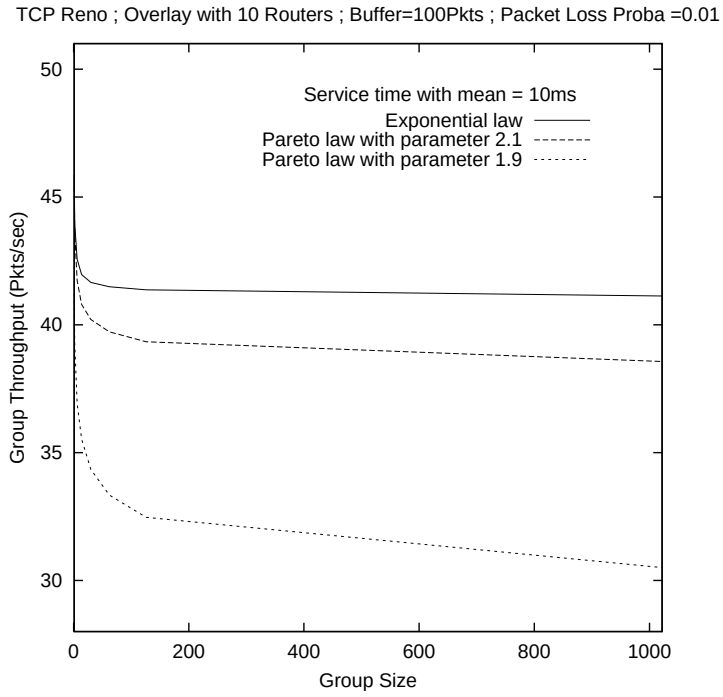

Figure 8: Group throughput for several laws of cross traffic, TCP Reno.

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 9 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.

5.2 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, 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.

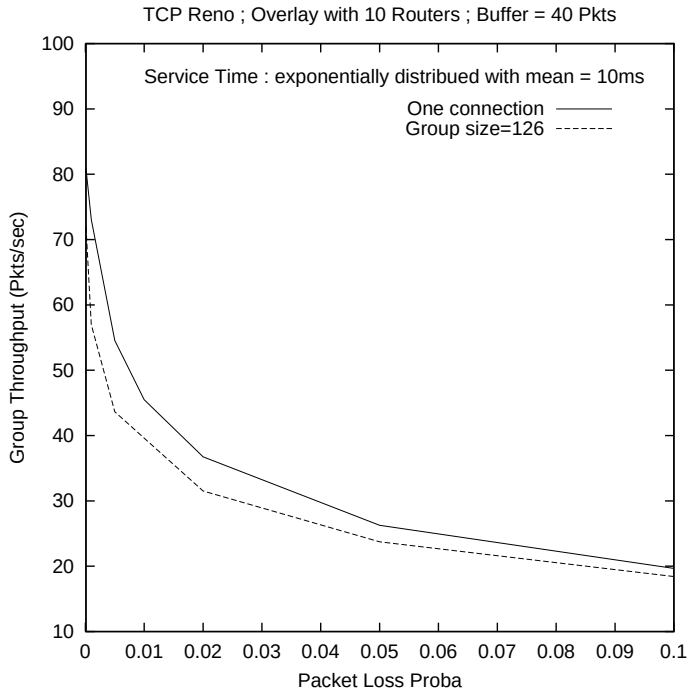

Figure 9: Group throughput w.r.t. packet loss probability, TCP Reno, small buffer.

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 (see Figure 10) 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 Table 2 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

equal to 50 packets, and size of the output buffer is variable. Output buffer size is given in packets.

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

Table 2: Scalability Experiments in Planet-Lab: Throughput in Pkts/sec

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 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 Table 3. 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.

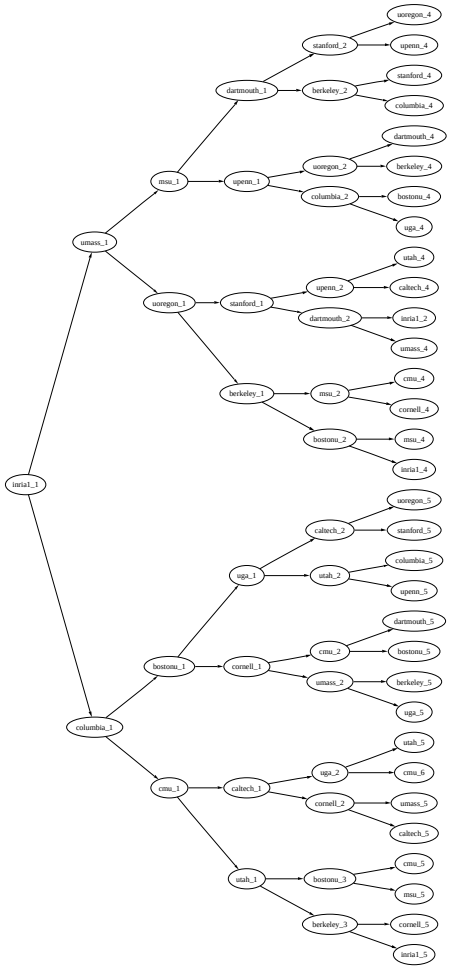

Figure 10: Multicast tree consisting of 64 PlanetLab nodes.

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). Table 4

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

Table 3: End-to-End Reliability Experiments in Planet-Lab

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.

	buf=50	buf=200	buf=500	buf=1000
2 failures	25.6	26.8	45.2	31.5
4 failures	29.2	28.8	36.4	27.2
6 failures	30.9	28.8	30.8	24.0

Table 4: Scalability vs. End-to-End Reliability. Throughput in KB/s

5.3 Simulation of Tree Evolution

To complement the simulations and experiments presented above, we further developed a discrete-event simulator to simulate the evolution of tree topology with failures and recovery under different algorithms. In particular, we evaluate the heuristics presented in Section 4 for the tree reconstruction.

Starting with a balanced binary tree of 1023 nodes, we uniformly choose a failing node, apply random or greedy heuristic to restore connectivity, and add the node back using best-join. The tree must remain binary, joins are only allowed at nodes with out-degree less than 2. We measure the length of longest path and average degree of non-leaf nodes. The two methods used for restoring connectivity are GREEDY_RECONNECT and a randomized procedure, that reconnects orphaned subtrees to randomly chosen nodes with out-degree less than 2.

The results are presented in Figure 11. The plots show average tree depth and inner node fanout over 500 runs. We observe that GREEDY_RECONNECT helps to maintain

significantly lower tree depth, and higher inner node degree, compared to the trivial approach that chooses active nodes randomly.

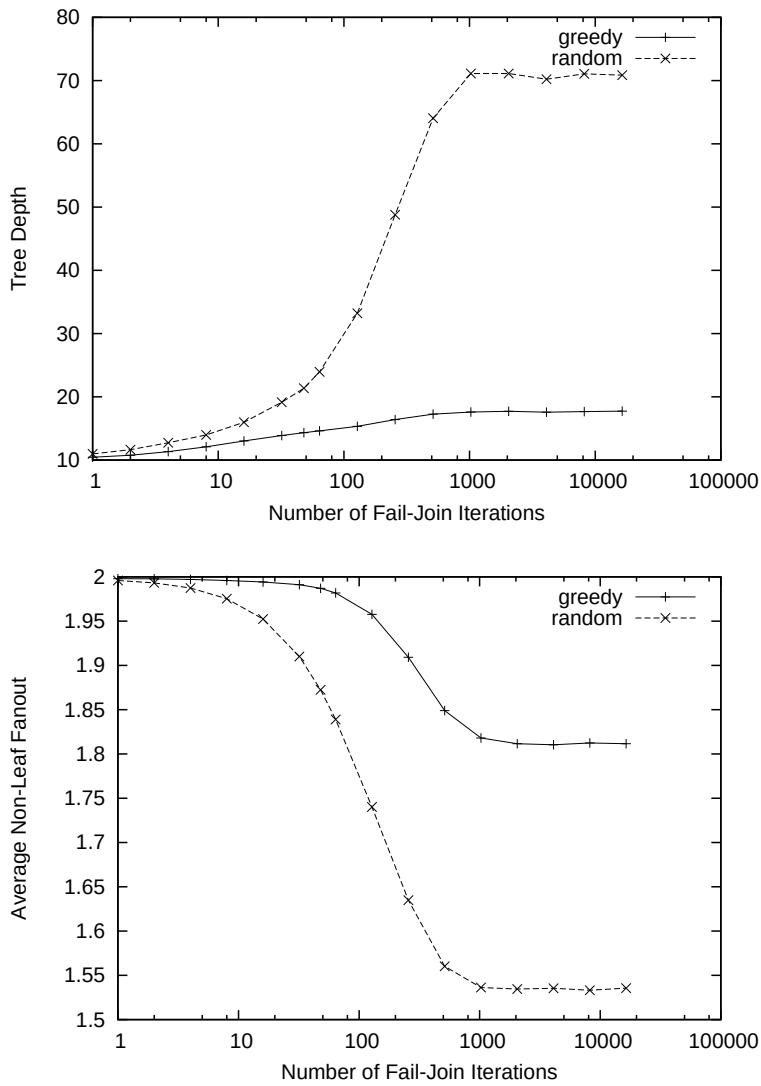

Figure 11: Evolution of tree depth (top), and evolution of average degree of non-leaf nodes (bottom).

6 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.

The next steps will consist in a more complete specification of the overlay architecture parameters. Another interesting question is that of the extension of the models and the analysis techniques to more elaborate statistical assumptions and more complete descriptions of the features of TCP. The general approach that consists in establishing a correspondence between such a problem and longest paths in some complex random graphs, and hence first passage percolation problems on such graphs, seems quite promising.

References

[1] M. Allman and V. Paxson. RFC 2581 - TCP congestion control, 1999.

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

[3] F. Baccelli, G. Cohen, G. Olsder, and J.P. Quadrat. *Synchronization and Linearity.* Wiley, 1992. (Out of print), (An electronic version is available at `http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html`).

[4] F. Baccelli and D. Hong. TCP is max-plus linear and what it tells us on its throughput. *ACM Sigcomm*, pages 219–230, 2000.

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

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

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

[8] C.Bormann, H.-C.Gehrcke J.Ott, T.Kerschat, and N.Seifert. MTP-2: Towards achieving the S.E.R.O. properties for multicast transport. *ICCCN*, 1994.

[9] A. Chaintreau, F. Baccelli, and C. Diot. Impact of TCP-like congestion control on the throughput of multicast group. *IEEE/ACM Transactions on Networking*, 10:500–512, 8 2002.

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

[11] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. *ACM Sigmetrics*, 6 2000.

[12] 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 Transactions on Networking*, 5(6):784–803, 12 1997.

[13] P. Francis. Yoid: Extending the internet multicast architecture, 4 2000. available at `http://www.icir.org/yoid/docs/yoidArch.ps.gz`.

[14] P. H. Hsiao, H. T. Kung, and K. S. Tan. Active delay control for TCP. *Globecom*, 11 2001.

[15] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. *4th Symposium on Operating Systems Design and Implementation*, 10 2000.

[16] J.F.C. Kingman. Subadditive ergodic theory. *Annals of Probability*, 1(6):883–909, 1973.

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

[18] B.N. Levine and J.J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *ACM Multimedia Systems*, 8 1998.

[19] J. Liebeherr and M. Nahas. Application-layer multicast with delaunay triangulations. *IEEE Globecom*, 11 2001.

[20] J. Martin. Large tandem queueing networks with blocking. *Queueing Systems, Theory and Applications*, 41:45–72, 2002.

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

[22] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. Almi: An application level multicast infrastructure. *3rd Usenix Symposium on Internet Technologies and systems (USITS)*, 3 2001.

[23] Planetlab, an open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org`.

[24] A. Riabov, Z. Liu, and L. Zhang. Overlay multicast trees with minimal delay. *Proceedings of ICDCS*, 2004.

[25] E. M. Schooler. *Why Multicast Protocols (Don't) Scale: An Analysis of Multipoint Algorithms for Scalable Group Communication.* PhD thesis, CS Department, California Institute of Technology, 2000.

[26] S. Shi and J. Turner. Placing servers in overlay net-works. *Technical Report WUCS-02-05, Washington University*, 2002.

[27] S. Shi and J. S. Turner. Multicast routing and bandwidth dimensioning in overlay networks. *IEEE JSAC*, 2002.

[28] G. Urvoy-Keller and E. W. Biersack. A multicast congestion control model for overlay networks and its performance. *NGC*, 10 2002.

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

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399