TCP-Friendly Many-to-Many End-to-End Congestion Control

Tal Anker
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
anker@cs.huji.ac.il
Radlan Computer Communications, Israel
tala@radlan.com

Danny Dolev
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
dolev@cs.huji.ac.il

Ilya Shnayderman
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
ilia@cs.huji.ac.il

Innocenty Sukhov
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
kesha@cs.huji.ac.il

Abstract

The paper addresses the issue of TCP-friendly congestion control mechanism for many-to-many communication environment. Lack of congestion control inhibits deployment of WAN applications that involve collaboration of groups of processes in the Internet environment.

Recent efforts targeted unicast WAN congestion control (TFRC). We extend that approach to multicast many-to-many applications that operate using a middleware framework. Our congestion control mechanism was implemented within a group communication middleware and tested in a multi-continent environment. The measurements have proved the proposed approach to be robust, efficient and TCP-friendly, as well as to provide fairness among processes that compete for shared resources.

1. Introduction

Congestion control is a fundamental building block that enables WAN deployable applications to function in the Internet environment. Many applications of this kind involve a group of processes that are to collaborate by communicating via multicast. There are several middleware implementations that offer group communication services with a variety of different semantics ([4, 5, 11, 17, 21, 27]). In the current paper we address the issue of TCP-friendly congestion control mechanism for group communication middleware. The importance of having a TCP-friendly mechanism stems from the necessity for an application to co-exist in the Internet with a variety of concurrent TCP sessions generated by other applications.

The specific group communication middleware that we focus on is *Xpand* [7] WAN group communication system (GCS). However, our results are applicable, with some adaptation, to other middleware systems as well. The main objective of the Xpand design is to address the needs of a wide spectrum of collaborative wide area network (WAN) applications without compromising the semantics of traditional GCSs, namely, group abstraction, group membership monitoring and reliable multicast. The services we exploited in order to add congestion control to Xpand were group membership and the ability to detect message losses, which are inherent to any reliable multicast scheme.

Xpand and similar group communication systems typically use UDP for all their communication needs, which allows them to benefit from the native IP multicast infrastructure for message delivery. For this reason, Xpand lacks the TCP built-in congestion and flow control mechanisms, which may lead to unfair bandwidth share, unstable transmission rate or even severe network congestion in WAN environment. Moreover, usage of TCP congestion control mechanism over UDP would neither scale nor provide the optimal results in a GCS.

In order to overcome these drawbacks, a congestion control mechanism was added to Xpand, based on the "TCP-friendly rate control protocol (TFRC)" [15], with modifications that extend it to a many-to-many framework and to general message traffic, without focusing on multi-media applications only. TFRC is an equation-based unicast congestion control mechanism in which the equation derived from a model of TCP long term throughput ([23]) is used to limit the sender's transmission rate. We have extended the TFRC equation-based approach from unicast to many-

to-many message exchange, where every member can be either a sender, or a receiver, or both.

A recent paper ([31]) that addresses a similar problem focuses on directly expanding TFRC to deal with multicast. The main difference between the two mechanisms is in the method used to determine the slowest receiver in a scalable manner. TFMCC uses the original randomized feedback cancellation scheme, while in Xpand we take the advantage of the built-in hierarchy, thus using a more natural approach to achieve scalability in multi-cluster based system. This approach enables building a simpler feedback mechanism without feedback separation, as well as a simpler round-trip time estimation.

In Xpand the feedback is aggregated based on the built-in hierarchical structure, and a sender is explicitly provided with an acceptable transmission rate from each remote receiving LAN.

Xpand was implemented and tested over the WAN environment described in [29]. Xpand not only utilizes the congestion control mechanism, but also contains a built-in flow control mechanism that follows the standard techniques described in the literature ([2, 13, 14]).

The performed measurements validated the implemented congestion control mechanism. The results showed that our extended TFRC model is applicable to WAN, as well as to many-to-many communication patterns. The resulting behavior is TCP-friendly, and fairness is achieved among multiple senders sharing resources.

The paper outline is as follows: Section 2 presents the environment assumptions; Section 3 covers the basic design assumptions; Section 4 presents the algorithm details; Section 5 covers some of the test results, and Section 6 considers various related papers.

2. Environment

The Xpand group communication system is a middleware for distributed many-to-many applications and is fully described in [7, 8]. The current paper presents only the essential features of the system and the relevant assumptions. The processes participating in Xpand are grouped into clusters, so that all members of each cluster belong to the same LAN. The clusters are spread over WAN. Xpand distinguishes between two types of processes: *regular* and *delegate*. An application instance linked to an Xpand library is a regular process. In each LAN, the delegate is a designated daemon that serves as a representative of its LAN to all the other Xpand clusters and may be spawned by the first application instance in the cluster. Among other benefits, this hierarchical design allows for better scalability of the congestion control mechanism. The relationships between Xpand's components and its environment are shown in Fig. 1. The congestion control mechanism is integrated within Xpand's core module.

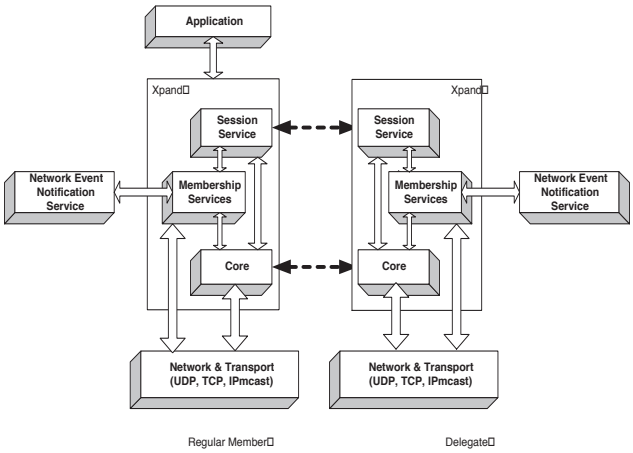


Figure 1. Xpand's layer structure

2.1. Application Layer Multicast

As was noted before, Xpand's reliable multicast service relies on the availability of a many-to-many communication substrate. The network service that supports many-to-many communication in IP networks is IP multicast. There is a limitation on using IP multicast since it is only partially deployed in various networks, which creates isolated regions supporting IP multicast.

In order to overcome this difficulty and to "bridge" regions supporting IP multicast, Xpand introduces a layer called *Application Layer Multicast* (ALM), which is used only as an interim solution. This layer constructs a message distribution tree incorporating delegate nodes, each node being chosen from a region that supports a native multicast. The tree construction is performed automatically and is beyond the scope of the current paper (See [7, 8]).

2.2. RON Testbed Environment

In our WAN experiments we used the Netbed's RON ([29]) wide-area nodes. The tests involved 6 sites, 5 located in North America and one in Jerusalem, Israel. The nodes are Celeron/733 or PentiumIII/1.1G machines running FreeBSD 4.7 / Linux 2.4 operating systems, connected via commodity Internet and Internet2. The links have diverse bandwidths, delays and packet loss rates (see Section 4.4). Since there is no IP multicast among RON machines, Xpand builds an overlay network (Application Layer Multicast (ALM)) over the nodes. With ALM, messages are delivered by unicast among the nodes. The ALM tree created for our specific set of RON nodes is described in Fig. 2.

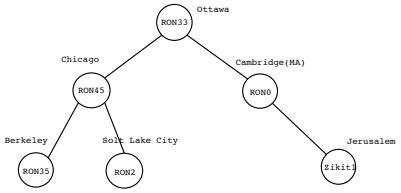


Figure 2. ALM layout

3. Xpand Flow/Congestion Control Mechanism Design

The design of Xpand congestion/flow control mechanisms takes advantage of Xpand's hierarchical structure. In Xpand, both senders and receivers from each LAN are represented by the LAN's delegate. We designed many-to-many single rate mechanism that is TCP-friendly, in which the delegates of the senders cooperate with the delegates of the receivers in order to adjust the message flow to be TCP-friendly.

To introduce the congestion control mechanism into Xpand, some parameters characterizing the network are measured by the delegates and used by the senders to adjust their transmission rate in accordance with the network conditions. In [16], a message flow is defined to be TCP-friendly if under the same conditions its sending rate is within a factor of two of the sending rate of TCP flow. To achieve that, Xpand must be able to estimate the appropriate TCP rate. To reach that goal, we use Eq. 1 from [23] characterizing TCP steady state throughput as a function of packet loss rate, estimated round trip time and TCP retransmission timeout:

$$T_{TCP} \approx \frac{s}{RTT\sqrt{\frac{2bp}{3}} + RTO \min\left(1, \sqrt{\frac{27bp}{8}}\right) p\left(1 + 32p^2\right)}, \quad (1)$$

where T is the desired transmission rate in bits/sec; s is the average packet size in bits; RTT is the estimated round trip time in seconds; $p \in [0, 1]$ is the loss event rate; $RTO = 4RTT$ is the TCP retransmission timeout value in seconds recommended in [16], and $b = 2$ is the number of packets acknowledged by a single TCP acknowledgement.

In order to use this equation, we designed mechanisms for RTT estimation and for loss rate measurement. These parameters are to be continuously calculated and translated into an acceptable rate for each receiving delegate.

While TCP is unicast, our model calls for many sending processes and several sending delegates. The congestion control mechanism is used to allocate an aggregate transmission rate for each LAN and, within it, to estimate each local sender application transmission rate. Afterwards, each sending delegate is to fairly distribute the aggregate rate among its local senders.

3.1. Design Decisions

Xpand design distributes the load of calculating the congestion parameters between senders' delegate and receivers' delegates. Each calculation is performed at the optimum location, as detailed below.

RTT Estimation: performed by the sender's delegate for each <remote delegate, group> pair. The RTT estimation does not require synchronized clocks.

Loss Rate Estimation: Since in most cases there are many more receivers than senders, it is logical to distribute the loss rate calculations among the receivers. The hierarchical structure requires that the calculation be done at the delegates. It appears reasonable to calculate the loss rate at the receivers' delegates which have more accurate information on message losses. The loss rate is estimated for a <receiver delegate, sender delegate, group> tuple.

Loss rate calculations require RTT estimation, which is done by senders' delegate and passed to receivers' delegates. As a result, a receiver's delegate gets RTT calculated half RTT earlier, which may slow down the responsiveness of the scheme. This inaccuracy is smoothed by using weighed RTT average.

We assume that RTT and the calculated loss rate are uniform for all receivers belonging to the same group within the same LAN. This is a safe assumption since message loss and RTT in LAN are negligible, compared to those in WAN.

Local sender's detection. The senders' delegate needs to identify potential senders in its LAN and their sending rate demand.

Local sender's rate allocation. The senders' delegate needs to distribute the aggregate transmission rate among its local senders. We use the "Max-Min fairness" ([9]) criteria to allocate the rates.

Local sender's traffic shaping. We use a token-bucket-like mechanism in order to enable the sender to conform to the assigned rate.

"Slow start" mechanism. We incorporate a slow start mechanism into the local sender's rate limit mechanism, so that it would be TCP-friendly even prior to a loss event.

Single LAN group. The congestion control mechanism is able to deal with a special case when all group members reside on the same LAN. Due to the lack of space, we do not present this aspect in detail.

Issues with Multicast Delivery Tree: There are several ways to build a multicast distribution tree in the Internet environment, using different multicast routing protocols. As we assume that multicast trees might be different for different groups, we measure RTT and loss rate for each group.

4. The Congestion Control Mechanism Description

In this section, the general algorithm flow is presented, followed by a detailed description of each step. While the mechanism is designed for many-to-many framework, the algorithm is presented from a single source LAN perspective and for a single group[1] (G). The source LAN delegate (SD) communicates with the delegates (RD) of LANs that contain receivers for the given group. Every delegate implements the same mechanism and thus imposes congestion control over all the senders and the receivers. At the same time, Xpand message flows are friendly to TCP flows that use the same links.

The basic steps of the algorithm are as follows:

1. RTTs are computed at the senders' delegate per receiver's delegate and per multicast group and afterwards passed to the receiver's delegate;

2. Loss rate is measured at every receiver's delegate per sending delegate and group <SD, RD, G>. In these measurements, the receiver's delegate uses the RTT estimation obtained from SD;

3. The acceptable receiving rate is calculated by RDs using the measured loss rate and the RTT. RD sends its acceptable rate to the corresponding SD;

4. SD obtains the acceptable (restricting) rates reported by all remote delegates and chooses the minimal value to be the aggregate transmission rate for the senders belonging to the group within its LAN;

5. SD adjusts the local senders' transmission rate according to the aggregate transmission rate.

4.1. RTT Estimation by Senders' Delegate

For a given group G, RTT is measured by SD for every LAN that contains members of G. Intra-LAN delays are negligible in WAN environment. We assume that all the messages flowing from the sending LAN to each receiving LAN (targeted for the same group) traverse the same route. We assume that the control traffic from RD to SD also takes a steady route, which may be different from SD to RD route. As a consequence, all *G* senders within SD LAN have actually the same *RTT* to RD. Thus, for each group we actually measure *RTT*s from SD to RD. The algorithm is straightforward, as can be seen in Fig. 3.

Note: RTT is ***not*** updated using ***retransmitted*** messages, as the original and the retransmitted messages have the same sequence number. This scheme allows RTT variations to change the transmission rate within approximately one round trip time, which is important for the scheme responsiveness.

[1] Congestion control among multiple groups is achieved implicitly.

4.2. Receiver Loss Rate Estimation

Loss rate is measured by a receiver's delegate (RD). Regular receivers do not participate in loss measurements. RD may receive multiple flows from the same sending LAN for a group. All such flows are aggregated and considered to be a single meta-flow. This is done to improve the statistics of message loss rate and to address scalability issues.

A packet loss in our implementation is detected when the sequence number of a received message within a specific flow is out of the order. Since we assume packets to traverse the same multicast (unicast) delivery path, packet reordering is infrequent and has no significant influence on the overall picture.

The model in [23] requires that two loss events be separated by at least RTT seconds to be statistically independent. A loss event starts on a packet loss and continues RTT seconds after it. To determine whether a lost packet should start a new loss event or be considered as a part of a current loss event, we compare the approximated time stamp of a lost packet with the time associated to the beginning of the last lost event[2]. To approximate the time stamp of a lost packet, we use the suggested interpolation to infer its nominal "arrival time" (see [15, 16] for details). In the calculation that follows, we ignore packets that arrive or are lost within RTT seconds of the time associated to the lost packet that started the loss event.

In order to compute the loss rate, we need to count the number of packets contiguously received between loss events. If a loss event is determined to have started at time T_1 and the next loss event started at time T_2, the number of packets between the loss events is the total number of packets in all n flows that have arrived within period $[T_1 + RTT,\, T_2)$.

The Xpand approach matches the model presented in [23], whereas the original specification ([15, 16]) also counts packets that arrive within the first RTT seconds of a loss event, thus underestimating the actual loss rate.

To calculate the loss rate p, we first calculate I_{mean}, i.e. the average number of packets received between loss events. This is done by calculating the moving average over the past average I_{past} and the most recent estimation (with a quotient 0.5).

$$I_{\text{mean}} = \frac{1}{2} I_{past} + \frac{1}{2} I_{\text{sample}} \ , \qquad (2)$$

[2] In TFRC ([15, 16]) the term "inter-loss interval" is used, defined as the interval from the beginning of a loss event until the beginning of the next loss event.

The RTT Calculation:

A sender multicasts packet P to the entire group G;

Upon receiving P, SD records P's arrival time T_{out};
/* T_{out} is an approximation of the sending time */

Upon receiving P, RD records P's arrival time T_{arr};

RD sends ACK for P to SD at time T_{ack} containing $\Delta = T_{ack} - T_{arr}$;

Upon receiving the ACK for P at time T_{now}, SD calculates $RTT = (T_{now} - T_{out}) - \Delta$;

Figure 3. RTT estimation at the sender's delegate

where I_{sample} is the number of packets that have arrived since the end of the last loss event. At the beginning of a new loss event I_{mean} becomes I_{past}. Equation 2 presents the average number of packets once a new event takes place. Immediately after a loss event, the value of I_{sample} is smaller than I_{past}, which results in an inaccurate value of I_{mean}. To deal with this undesirable effect, we do not update I_{mean} until either a new loss event arrives, or $I_{\text{sample}} \geq I_{past}$.

The quotient is chosen to provide responsiveness, on the one hand, and to filter out samples that are too far from the average value due to inaccuracy of measurements, on the other hand. One of the original goals of [15, 16] was to achieve congestion control that is TCP-friendly, while allowing traffic rate that is smoother than that of TCP. This property is important for various applications, e.g. multimedia. Since for GCS the smoothness is not an important property, we used a simpler equation (Eq. 2) that produces a TCP-friendly rate change response, though not that smooth.

RD computes the loss rate ($p = 1/I_{mean}$). Using this value and the RTT, RD calculates the acceptable receiving rate (Eq. 1) and sends it to SD.

SD computes the updated aggregate rate by taking the minimum over all the acceptable receiving rates obtained from the relevant RDs. It then divides this rate among its local senders and notifies them about their new sending rate[3] (See Section 4.3). Each local sender uses a token-bucket-like mechanism in order to shape its traffic to comply with its sending rate.

4.3. Multiple Senders in LAN

As was noted above, we determine the transmission rate per group, as we assume the possibility of different multicast distribution trees for different groups. Given an aggregate transmission rate for a particular group, there is a problem of dividing the aggregate group transmission rate among multiple senders residing on the same LAN. This is the task of the senders' delegate to distribute the transmission rate among local regular senders. The delegate uses a "Max-Min fairness" ([9]) criteria to ensure a fair rate allocation.

4.3.1 Determining local senders' rate demands

We describe here a scheme that enables SD to learn its local senders' demands for transmission rates to be used as an input to the fair rate allocation scheme.

The demand evaluation scheme is to be responsive enough to avoid under-utilization of the resource (transmission rate), e.g. in case when one sender receives a rate higher than it actually needs, whereas the other one receives a rate lower than it needs.

Application transmission rate may change more frequently than the aggregate rate, since the latter is computed infrequently and is smoothed, as opposed to the bursty traffic pattern of an individual sending application.

Thus, SD must sample[4] its local senders' demands much more frequently than it receives aggregate rate updates. Each time a new vector of senders' demands is collected, the delegate recomputes the sending rate allocation.

Each local sender determines its application demand by measuring the rate at which Xpand's sending window fills up and gets drained. To avoid unnecessary fluctuations when evaluating a transmission demand, the sender uses weighed "history" of its window's occupancy. To provide fast responsiveness, the weight of the "history" must be lower than that of more recent values.

When the window is full, the local sender cannot measure its application's transmission demand, as the application is blocked. In this case, a special approach is taken to estimate transmission rate demand. There is no point requesting a larger rate from SD if the last requested rate has

[3] The two-way communication between SD and its local sender regarding sending rate allocation is conducted via a reliable channel and is beyond the scope of the current paper.

[4] The sender can explicitly ask for transmission demand update from its delegate if its local application demand changes by at least some δ amount.

not been complied with. Since SD uses the Max-Min fairness criteria, increasing the requested rate will not increase the allocated rate. If the delegate complies with the last rate request and the window is still full, the sender doubles its requested rate demand. This process continues until the window opens up.

When a new sender starts transmitting, it is always permitted to start sending at some predefined initial rate. The delegate recognizes the presence of the new sender and takes it into account by recalculating the sending rate allocation.

4.4. Implementation Details

Xpand has a complementary built-in flow control mechanism, which is used as long as no congestion is detected over the WAN. Once congestion is detected, the above described TCP-friendly congestion control mechanism takes over. If no remote receiver exists, i.e. all the members reside within a single LAN, the TCP-like window congestion avoidance mechanism is used.

We also use a slow-start mechanism for some specific cases, e.g. if no remote members exist and a new sender starts its transmission. Another case is when no remote receiver has detected any loss event, thus the acceptable receiving rate that is sent to the SD is infinite. Obviously, when a loss is detected, the sending rate is immediately updated.

5. Performance Results

The congestion control was tested in multi-continent (WAN) setting of RON testbed Network. The tests included six sites: Zikit1 (Jerusalem), RON0 (Boston), RON2 (Salt Lake City), RON33 (Ottawa), RON35 (Berkeley), RON45 (Chicago). All measurements were carried out within the same time-window every day. The time-window was chosen to be early morning time in the North America, which is mid-day in Israel. As there is no IP multicast connectivity among these sites, a propriety application layer multicast (ALM) mechanism was used ([7]). The obtained message distribution tree is presented in Fig. 2. RON33 was selected to be the tree root by the algorithm used in the dynamic ALM tree construction.

In tables 1, 2 and 3, the columns represent senders and the rows represent receivers. These numbers were collected during the first test (see below).

Table 1 presents the median round trip time (RTT) among the sites as it was measured by the senders. The messages from the sender to the receiver were sent along the ALM tree, and the ACKs were sent via unicast back to the sender.

Table 1. Round trip time (RTT) (ms)

Site	Zikit1	RON0	RON2	RON33	RON35	RON45
Zikit1		170.92	282.15	205.25	277.99	213.99
RON0	170.62		120.18	51.67	122.20	62.82
RON2	260.60	91.95		57.65	77.15	57.87
RON33	236.30	50.00	85.70		85.59	26.68
RON35	299.13	121.32	96.58	76.23		59.80
RON45	242.47	62.29	58.75	25.27	59.64	

Table 2. Loss rate (%)

Site	Zikit1	RON0	RON2	RON33	RON35	RON45
Zikit1	0.00	0.00	2.90	0.00	7.63	2.33
RON0	0.33	0.00	3.42	0.00	7.40	2.47
RON2	4.20	2.87	0.00	3.39	6.35	1.96
RON33	0.33	0.00	3.31	0.00	7.84	2.67
RON35	4.80	3.26	3.34	4.36	0.00	1.50
RON45	1.15	0.82	0.71	0.63	4.85	0.00

Table 2 presents the median loss ratio as calculated by the receivers. One can notice that loss ratio increases along the ALM paths, as the number of ALM overlay links increases.

Table 3 presents the theoretical rate calculated by Eq. 1. There are some blank cells in the table, since no message was lost on the corresponding links due to the fact that the rate was limited by another receiver.

In the following subsections we present performance measurements of the congestion control mechanism within Xpand system. The measurements validate the congestion control mechanism, its TCP-friendliness and its applicability to multicast in WAN.

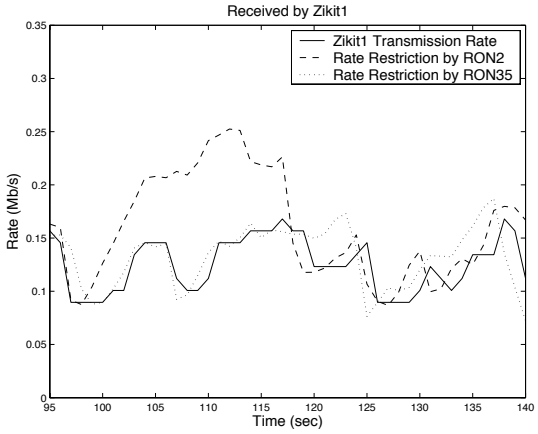
5.1. Rate Restriction

In the first test, six members join the same group G. Each member in turn sends messages to G for 60 seconds, at the rate allowed by the congestion control mechanism. Zikit1 measures data arrival rate for each sender (including itself). Figure 4 presents the received rate measured by Zikit1 for each sender. In order to evaluate the congestion control mechanism, we examined the performance of Zikit1. The variation in its rate is caused by the changes in rate forced by the slowest receiver.

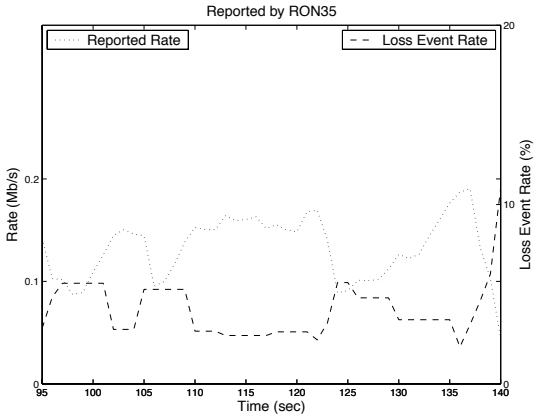
Figure 5(a) shows the slowest receiver limiting Zikit1

Table 3. TCP-friendly rate (Mb/s)

Site	Zikit1	RON0	RON2	RON33	RON35	RON45
Zikit1			0.16		0.07	0.24
RON0	0.96		0.33		0.16	0.80
RON2	0.13	0.49		0.69	0.30	1.02
RON33	0.69		0.48		0.22	1.79
RON35	0.10	0.34	0.42	0.43		1.17
RON45	0.34	1.60	1.84	4.58	0.50	



(a) Zikit1 is limited by RON2 and RON35



(b) Reported rate vs. loss rate

Figure 5. Rate limitation factors

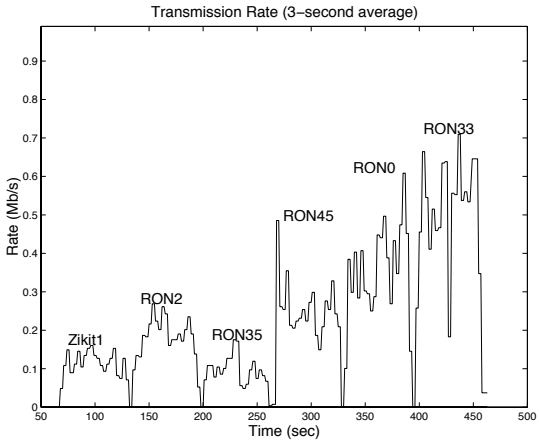


Figure 4. Sequential sending

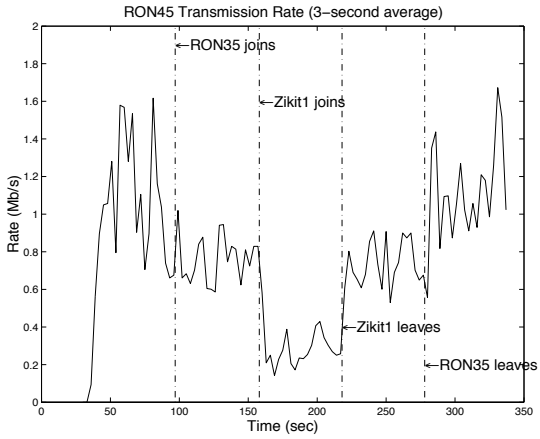


Figure 6. Responsiveness to membership changes

sending rate. It is clear that the sending rate is limited by the rate reported by the slowest receiver. We present only the two slowest receivers that affect the sending rate, since other receivers have reported a much higher receiving rate.

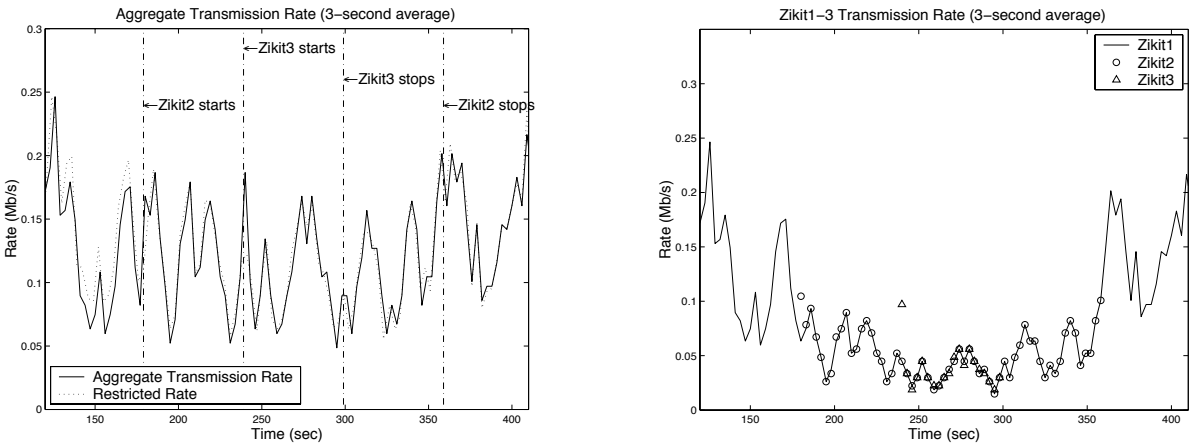
Figure 5(b) presents the loss rate and the reported rate calculated by RON35 for Zikit1 messages over the same time period. One can see that the dominating factor in the rate reported by the receiver is the loss rate, since the RTT remains relatively stable during this period.

5.2. Rate Adaptation to Membership Changes

In this test, all RON machines except RON35 and Zikit1 were members of group G. The application running at RON45 sent as much traffic as it was allowed by other delegates. In Fig. 6 we see that when RON35 joins G, the sending rate of RON45 drops. Afterwards, when Zikit1 joins G, the rate again drops significantly. 60 seconds later, when Zikit1 leaves G, the rate increases and then increases again when RON35 leaves G.

5.3. Fairness Among Senders in a LAN

In Fig. 7(a) we see aggregate rates generated by Zikit1, Zikit2 and Zikit3 while all RON computers receive the messages (Zikit[1-3] all residing in the same LAN at the Hebrew University of Jerusalem). Zikit1 sends messages through the entire duration of the test, while Zikit2 and Zikit3 send messages for a shorter period (180 seconds and 60 seconds respectively). The graph also presents the rate



(a) Aggregate Rate (b) Particular Rate

Figure 7. Aggregation of multiple senders and fairness

by which Zikit's LAN was limited by the other delegates. One can see that the influence of the reported rate, which in turn depends on the loss rate, is much higher than the impact of the number of senders.

In Fig. 7(b) we present the achieved sending rate of three senders sharing the same aggregate transmission rate during the test. The results clearly show that Max-Min fairness is achieved. The measured results were averaged over 3-second intervals. For presentability we chose a relatively large average interval to be shown in Fig. 7(b), while the results were similar on smaller scales.

5.4. TCP-friendliness

Figure 8 presents the TCP-friendliness test results. In this test, we ran six cycles by the same scenario: every cycle started with a 60-second Xpand session, followed by a 60-second TCP and Xpand joint session.

In this test, only Zikit1 and RON35 were the group members. Both Xpand and TCP messages were sent by Zikit1. The achieved rate was averaged over a 3-second interval. The average rate over the periods, when both flows shared the network resources, are also presented in the figure.

In all the cycles, the performance of Zikit1 showed that the bandwidth limited by our congestion control mechanism is comparable to that of TCP and, therefore, the message flow is TCP-friendly.

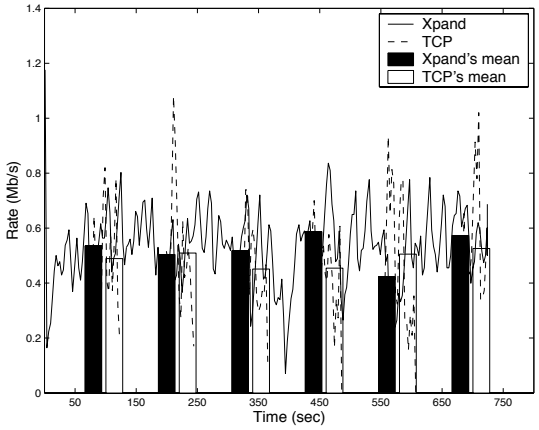


Figure 8. TCP-friendliness

6 Related Works

Congestion and flow control have attracted a lot of research in computer networking. There are numerous approaches to introducing congestion control mechanisms for multicast applications. A comprehensive survey on current unicast and multicast congestion control approaches can be found in [30].

In the current work we focus on a single rate multicast congestion control in which the slowest receiver limits the transmission rate. The major alternative approach uses layered multicast scheme ([10, 19, 28]). Our choice of a single rate is accounted for by the fact that group communication framework requires that all receivers be synchronized. The proposed approach could be extended to multi-layer multicast by using the proposed technique on the rate of each layer.

Several papers propose congestion control schemes for systems with a large number of receivers. In such environment, it is impossible to handle feedback from each sender due to the ACK explosion problem. It is necessary to esti-

mate the receiving capability of the slowest receiver. There are two main approaches used to avoid the ACK explosion problem, the first identifying the slowest receiver and referring to it only, and the second enforcing a hierarchy on the receivers.

PGMCC and TFMCC use the first approach, whose main disadvantage comes from the fact that the 'worst' receiver may change rapidly, while performing a switch-over among different slow receivers might be slow and extremely difficult ([12, 18]).

The second approach is more suitable for GCS, as group membership is maintained and available for group members. Xpand uses an external group membership service to receive group membership notification ([6]). MTCP ([25]), being a one-to-many scheme, creates a multi-level tree hierarchy on receivers. Obviously, a multi-level hierarchy is natural to one-to-many approach. Xpand, which is designed for many-to-many communication, uses a two-level hierarchy, which is a more natural approach for multi-cluster systems. Other protocols like RMTP ([24]) and TMTP ([32]) are one-to-many, taking advantage of the hierarchy and are not specifically designed to be TCP-friendly. RMTP deploys end-to-end congestion control, whereas TMTP implements flow control only.

Multicast communication typically consumes more resources from the network than unicast, since multicast traffic usually traverses through more links, thus a protocol designed for bulk data transfer over multicast in WAN must exhibit TCP-friendliness. TEAR ([26]) and MTCP ([25]) use a window-based technique to provide TCP-friendly congestion control mechanism. The authors report promising results, however, the model is not easily extensible to many-to-many multicast applications due to scalability problems.

An alternative approach to the window-based mechanism implemented in TEAR and in MTCP is the equation-based one. This approach uses a stochastic TCP model ([22]) which represents a throughput of a TCP sender as a function of packet loss rate and round trip time. TFRC (RFC 3448 ([16]), see also the paper [15]) has been recently recognized and standardized by IETF as a sound approach to TCP-friendliness for unicast traffic. The congestion control mechanism deployed in Xpand was based on this approach.

Only few group communications systems have an end-to-end congestion control mechanism. As to flow control mechanisms for group communication, they have been widely employed (a comprehensive analysis of flow control mechanisms for LAN GCSs can be found in [20])

A system that is comparable to our system is Spread ([4]). Spread uses an overlay network, in which each overlay link behaves in a TCP-friendly manner. In addition, Spread implements an advanced end-to-end flow control mechanism based on a cost-benefit approach([3]).

7 Conclusions and Future Work

Our study has proved that it is possible to create a many-to-many congestion control mechanism that is TCP-friendly and provides a built-in fairness among senders that share resources.

Future research needs to focus on improving the results, in particular, the responsiveness of the mechanism to message losses and group membership changes. In our current implementation, retransmissions were not considered to be a part of the sending rate, though a conservative approach was taken. We intend to find an appropriate way to incorporate retransmissions in the congestion control mechanism while allowing for heterogeneous (unicast and multicast) retransmission schemes.

References

[1] ACM. *Communications of the ACM 39(4), special issue on Group Communications Systems*, April 1996.

[2] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, April 1999. Internet Engineering Task Force, Network Working Group.

[3] Y. Amir, B. Awerbuch, C. Danilov, and J. Stanton. Global Flow Control for Wide Area Overlay Networks: A Cost-Benefit Approach. In *OPENARCH-2002*, pages 155–166, June 2002.

[4] Y. Amir, C. Danilov, and J. Stanton. A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication. In *Proceedings of ICDSN'2000*, 2000.

[5] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: A Communication Sub-System for High Availability. In *22nd Annual International Symposium on Fault-Tolerant Computing*, pages 76–84, July 1992.

[6] T. Anker, D. Breitgand, D. Dolev, and Z. Levy. CONGRESS: Connection-oriented group-address resolution service. In *Proceedings of SPIE on Broadband Networking Technologies*, November 2-3 1997.

[7] T. Anker, G. Chockler, I. Shnaiderman, and D. Dolev. The Design and Performance of Xpand: A Group Communication System for Wide Area Networks. Technical Report 2001-56, Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel, August 2001. URL: http://leibniz.cs.huji.ac.il/research/, See also the previous version TR2000-31.

[8] T. Anker, D. Dolev, and I. Shnaiderman. Ad Hoc Membership for Scalable Applications. In *16th Intl. Conference on Distributed Computing Systems*, Oct. 2002.

[9] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice-Hall, 1992.

[10] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. FLID-DL: Congestion Control for Layered Multicast. In *Second Int'l Workshop on Networked Group Communication (NGC 2000)*, November 2000.

[11] G. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Computing Surveys*, 33(4):427–469, 2001.

[12] D. DeLucia and K. Obraczka. Multicast feedback suppression using representatives. In *INFOCOM (2)*, pages 463–470, 1997.

[13] S. Floyd. Congestion Control Principles. RFC 2914, September 2000. Internet Engineering Task Force, Network Working Group.

[14] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[15] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[16] M. Handley. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448, January 2003. Internet Engineering Task Force, Network Working Group.

[17] M. Hayden. *The Ensemble System*. Phd thesis, Cornell University, Computer Science, 1998.

[18] S. K. Kasera, G. Hjálmtýsson, D. F. Towsley, and J. F. Kurose. Scalable reliable multicast using multiple multicast channels. *IEEE/ACM Transactions on Networking*, 8(3):294–310, 2000.

[19] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM*, volume 26,4, pages 117–130, New York, Aug. 1996. ACM Press.

[20] S. Mishra and L. Wu. An evaluation of flow control in group communication. *IEEE/ACM Transactions on Networking (TON)*, 6(5):571–587, 1998.

[21] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos. Totem: A Fault-Tolerant Multicast Group Communication System. *Communications of the ACM*, 39(4), April 1996.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, Vancouver, CA, 1998.

[23] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, April 2000.

[24] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, April 1997.

[25] I. Rhee, N. Ballaguru, and G. N. Rouskas. MTCP: Scalable TCP-like congestion control for reliable multicast. In *INFOCOM*. IEEE, Mar. 1999.

[26] I. Rhee, V. Ozdemir, and Y. Yi. Tear: Tcp emulation at receivers – flow control for multimedia streaming, 2000.

[27] R. van Renesse, K. P. Birman, and S. Maffeis. Horus: A Flexible Group Communication System. *Communications of the ACM*, 39(4), April 1996.

[28] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *INFOCOM (3)*, pages 996–1003, 1998.

[29] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association. url: www.emulab.net.

[30] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network*, 15(3):28–37, 2001.

[31] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 275–285. ACM Press, 2001.

[32] R. Yavatkar, J. Griffoen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, pages 333–344, 1995.