Organizing data transmission for reliable multicast over satellite links

Antoine CLERGET, Walid DABBOUS
{aclerget,dabbous}@sophia.inria.fr
I.N.R.I.A. U.R. de Sophia Antipolis,
2004 route des Lucioles - BP 93,
06902 Sophia Antipolis Cedex (France)

1 Introduction

Intrinsically broadcast communication channels, satellite links offer a natural way of multicasting data over a large geographical region. Using such links, one can benefit from an environment where adding thousands of new recipients does not cost anything in terms of network resources. There are a lot of potential applications such as software distribution, database updates, information broadcast (weather forecast, financial data, ...). However, some constraints are associated with satellite links : first, for geosynchronous satellites, transport protocols must be able to cope with very important delays. Important delays lead to a poor system reactivity and combined with link's asymmetry, or even unidirectionality, feedback from receivers may be quite difficult to implement efficiently. Second, satellite links, as all other wireless links, undergo an important Bit Error Rate (BER). To lower that error rate to levels comparable to that of wired-links, one must add an important link-level bit redundancy, reducing the useful throughput of the link. This could be avoided if the transport layer supported corruption losses. The problem is therefore to deal with congestion in an environment where feedback is quite inefficient and losses are not all due to congestion.

Having a very large number of recipients, some receiving the data directly from the satellite link, others receiving it relayed by an antenna through the M-Bone (the best is for each receiver to receive the data from the satellite through the closest antenna available), we will suppose that we have an important heterogeneity of paths leading to them in terms of bandwidth and error rate. Our aim is to ensure reliable data transmission and to minimize for each receiver its transmission time.

To match the constraints mentioned earlier, as well as scalability considerations, we study feedback-free mechanisms, which means that the recipients do not acknowledge the data received. To ensure reliability, we must therefore adopt a Forward Error Correction (FEC) technique, where lost data can be recovered with redundancy packets. Even without feedback, we can bring the failure probability as low as we want by transmitting for a long enough time. We can make sure that introducing so much redundancy has no impact nor on the receivers, since they end the reception as soon as they have enough information, nor on the network, which thanks to pruning, will not relay the surplus of packets.

2 Related Work

The issue of reliable multicast over satellite links has already been studied, giving birth to protocols such as MFTP [1]. In MFTP, the file is transmitted entirely on the first pass, and missing pieces on subsequent passes. However, it considers that all recipients receive the file directly through the satellite link, and therefore does not deal with problems such as rate and congestion control, or receiver's heterogeneity. A number of other reliable multicast protocols using feedback deal with congestion control, by adapting the sender's rate to the worse or to a given proportion of the receivers in reply to "congestion reports".

As outlined in Receiver-driven Layered Multicast (RLM) [2] (which does not raise the issue of reliability) it is interesting to use multiple multicast groups to deal with receiver's heterogeneity. The transmission rate is receiver-driven since it is the receiver that adjusts it to avoid congestion by joining or leaving one of the multicast groups called "layer". In [3], Lorenzo Vicisano, Luigi Rizzo and Jon Crowcroft present a protocol based on FEC and layered multicast that uses little or no feedback for congestion control and error correction. However, our work is focused on a context where we experience corruption losses, as observed on the satellite link, and therefore we do not make the assumption that losses are due to congestion. Moreover, we propose another way of organizing data within layers that does not impose an exponential distribution of rates. To mimic the behavior of TCP, some protocols that deal with congestion adopt a strategy of multiplicative decrease in rate when congestion is detected, and additive increase otherwise. Others estimate the "equivalent" throughput, using the relation between throughput and loss rate for a TCP connection : $Throughput \sim 1/\sqrt{Loss\ rate}$. These protocols are said to be "TCP-friendly", because they should behave like TCP when confronted to a congested network. For such protocols, an exponential distribution of rates leads to a slow and imprecise reaction to congestion.

3 Data Organization

To correct errors without acknowledging sent data, we introduce redundancy within sent packets : k packets of data are encoded into n packets in such a way that receiving any k among these n is sufficient to rebuild the original k packets. Forward Error Correction gets more and more efficient compared to retransmit queries as the number of recipients grows since they can all correct $n-k$ errors, eventhough these errors are different. It is interesting to use large values of k and n, (ideally, k is the number of packets in the file to be transmitted, and $n >> k$). Unfortunately, known FEC techniques for large values of k and n are slow to compute. The file to be transmitted is therefore split into B "blocks" of k packets. Each block is then "fec-encoded" into a block of n packets. The end-user ends the reception when he receives k different packets from each of the B blocks.

In a totally feedback-free environment, and even in general, it is interesting to manage receivers' late arrival. In an application transmitting informations day long, we would like to be able to have users join at any time and receive the data in minimum time. Given a reception window of $B.k$ packets, we should then have k (different) packets of each block (Characteristic C_1). To ensure quick recovery of lost data, we must also have a good block interleaving, which means that given any reception window of k packets, we should have a packet of each block (Characteristic C_2).

We send packets through "channels" at the same rate and group these channels within the different layers, defining their respective rates. The sending rate of a layer is therefore proportional to the number of channels sent on it. The following data organization tries to meet the characteristics defined above for the $B.k$ and k packets reception windows (C_1 and C_2), no matter how many channels the receiver is listening to.

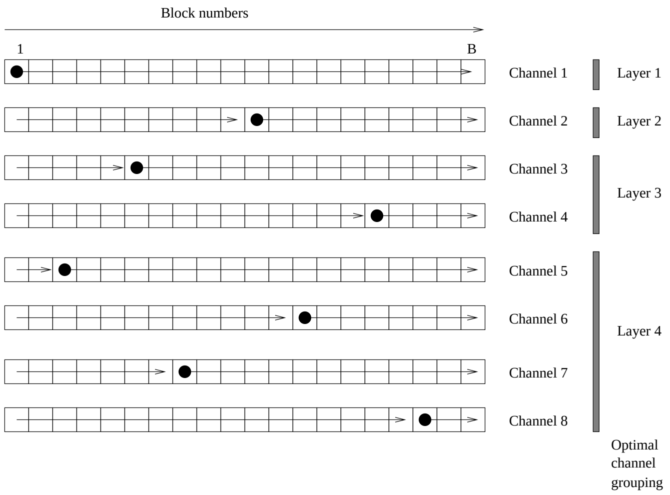

Figure 1: Block numbers of packets transmitted on the channels over time

The packet sent at date t on channel number c is defined by the pair ([block number b_c^t], [packet number p_c^t]) (fig. 1) :

- $b_c^t = \underbrace{\lfloor B.I(c) \rfloor}_{\text{offset}} + t\ [mod\ B]$ and $I : x = \sum_{i=0}^{m} b_i.2^i \mapsto I(x) = \sum_{i=0}^{m} b_i.2^{-i}$
- p_c^t is the index the first non-sent packet in block b_c^t.

The idea behind layered multicast is to reduce congestion by leaving some of the multicast groups ("higher" layers), pruning preventing their packets from being forwarded through the bottleneck. But for pruning to work, receivers behind a same bottleneck must unsubscribe to the same layer. Recipients must then follow this rule : *To subscribe to layer L_i, a recipient must first subscribe to layer L_{i-1}* .

Whatever the data organization, meeting characteristics C_1 and C_2 for all receivers, or, in other words, for all receivers to finish in minimum time when there is no loss (i.e. $File\ size = B.k / \sum Subscribed\ layers'\ rate$), and to recover from losses as quick as possible (without feedback), layer's i rate must be a multiple of the sum of layer's 1 thru $i-1$ rate :

$$\exists p \in \mathbb{N}, r_i = p.\sum_{j=1}^{i-1} r_j$$

This means that the layers' rate distribution must be exponential.

With our data organization, a receiver (possibly arriving late) will finish in minimum time if it listens to 2^m channels, $m >= 0$. Since we do not want to have an exponential rate distribution, we must accept a slight overhead (between 0% and 15%) when listening to n channels, $n \neq 2^m, \forall m$, and will have an optimal receiving time otherwise.

We have therefore proposed a data organization that :

- Is optimal for all receivers, whatever the number of layers they are listening to, if we accept to have an exponential distribution of rates.

- Is optimal for the receivers that listen to a number of layers that, grouped, contain 2^m channels, $m > 0$ and slightly suboptimal (overhead between 0% and 15%) for others if we wish to be able to set arbitrary layer rates.

3.1 Overhead

3.1.1 Due to non-exponentiality

We compare the reception time when listening to N channels to an optimal reception time (without the suboptimality due to a non-exponential rate distribution), i.e. the reception time when listening to 1 channel divided by N.

- N : Number of channels
- B : Number of blocks
- R : Sending rate (in packets per second)
- k : Number of packets per block
- p : Packet loss probability
- Probability of receiving k packets among n :

$$P^a_{k,n} = C^k_n.p^{n-k}.(1-p)^k$$

- Probability of having to send n packets for the receiver to receive k packets

$$P^b_{k,n} = (1-p).P^a_{k,n} = C^{k-1}_{n-1}.p^{n-k}.(1-p)^{k-1}$$

- Probability of having to send n packets for one of the B blocks :

$$Prob^c_{k,n} = \left(\sum_{j=k}^{n} Prob^b_{k,j}\right)^B - \left(\sum_{j=k}^{n-1} Prob^b_{k,j}\right)^B$$

- Let u be defined as[1] :

$$u_{0,n} = 0$$

$$u_{k,n} = u_{k-1,n} + \frac{1}{2^m} + \left\{ \begin{array}{l} \frac{1}{2^m}\ if\ I(2^m - 1 - 2^m * u_{k-1,n}) > n \\ 0\ otherwise \end{array} \right\}$$

and $2^{m-1} < n \leq 2^m$.

To receive k packets from each block, the receiver has to wait for $\lceil D_{k,N} \rceil$ packets :

$$D_{k,N} = B.\left(\lfloor \frac{k}{N} \rfloor + u_{k\%N,N}\right)$$

- Expected receiving time for one receiver :

$$Exp(p,k,N) = \sum_{j=k}^{\infty} Prob^c_{k,j}.\lceil D_{j,N} \rceil / R$$

[1] $I(2^m - 1 - 2^m * x/B)$ represents the channel whose pointer is on the last block after x steps

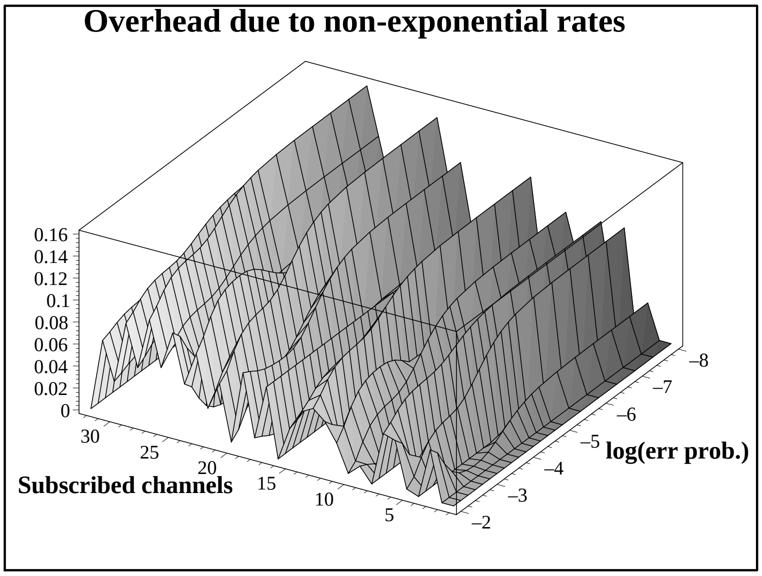

Figure 2: Suboptimality due to non-exponentiality of channel rates distribution

- Optimal receiving time :

$$Opt(p, k, N) = Exp(p, k, 1)/N$$

Graph 2 shows the value of $\frac{Exp(p,k,N)}{Opt(p,k,N)} - 1$ with $k = 8$. What is interesting is to notice that it reaches 15% in the worst case. The reception time is indeed optimal for $N = 2^m$.

3.2 Overhead compared to an ideal ARQ

We here compare the data transmission time with our scheme and with an ideal ARQ where acknowledgments are sent immediately and never lost.

- Probability of successfully transmitting the packet to 1 receiver after n retransmissions :

$$P_n^d = 1 - p^n$$

- Probability of successfully transmitting the packet to r receivers after n retransmissions :

$$P^e = (1 - p^n)^r$$

- If T represents the number of transmissions needed to send one packet to r receivers :

$$Exp_{ARQ}(T) \quad = \quad \sum_{n=0}^{\infty} Prob(T = n).n$$

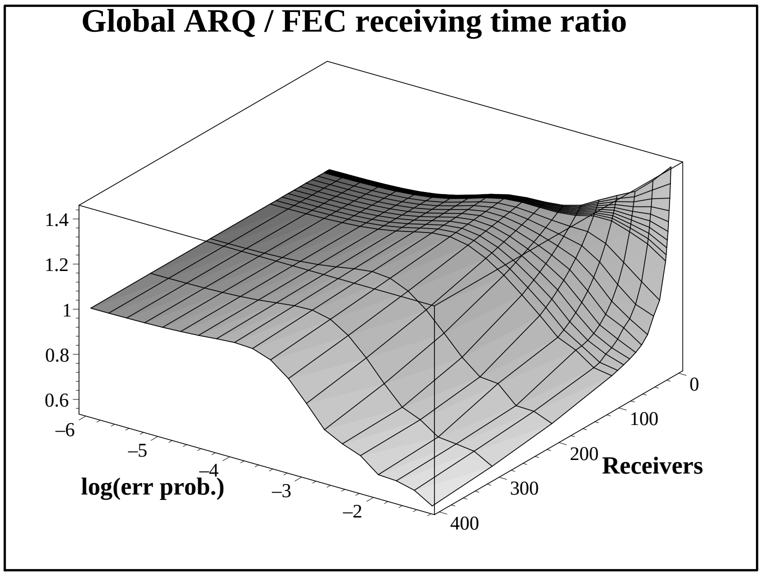

Figure 3: Suboptimality due to non-exponentiality of channel rates distribution

$$
\begin{aligned}
&= \sum_{n=0}^{\infty} (Prob(T \geq n+1) - Prob(T \geq n)).n \\
&= \sum_{n=0}^{\infty} Prob(T \geq n) \\
&= \sum_{n=0}^{\infty} 1 - (1-p^n)^r
\end{aligned}
$$

Graph 3 shows the value of $\frac{Exp(p,k,N)}{Exp_{ARQ}(p,r)}$ with $k = 8$ and $N = 1$ (or $N = 2^m$).

4 Ending the transmission

To ensure reliability without feedback, redundancy must be sent as long as receivers miss information to rebuild the file (i.e. as long as they haven't received k packets per block). With $n >> k$, we should have enough redundancy computed to satisfy all the receivers, but if necessary, additionnal redundancy can be added by resending over and over again these same n packets. The transmission stops when all receivers are satisfied. The sender therefore takes the decision to stop transmitting :

- When all receivers leave the multicast groups.
  With a source based multicast routing protocol, the pruning will then reach the source's local network mrouted. We can either modify the local mrouted to inform the source that all receivers left, or have a spy behind the mrouted report to the sender.

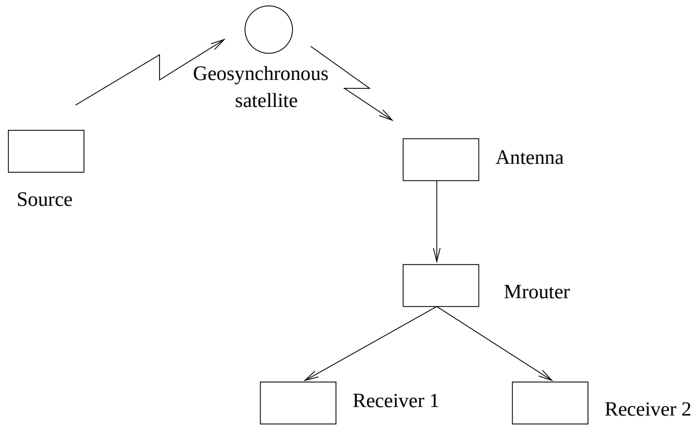

Figure 4: Experimental framework

- After a long enough duration.
  The longer the tranmission, the higher the probability of success. We can therefore choose to be as reliable as we want. Limiting in time the transmission is in some way equivalent to setting an inactivity time in TCP. When all receivers leave, the only useless resource used is the sender's CPU and the local network bandwidth.

4.1 Detecting and avoiding congestion

Detecting and avoiding congestion is necessary if we want to adopt a socially correct behaviour. However, it is not the only reason for a receiver to reduce his receiving rate. Congestion losses will lead to a global inefficiency due to the reception of useless packets. Using a Eutelsat satellite link at INRIA, we have set up an experimental framework (fig. 4).as shown on figure 4. Our experiment shows that setting a receiving rate equal to the bottleneck bandwidth (300 Kb/s) is more efficient than setting a receiving rate greater than that value (400 Kb/s). At the beginning of the transmission, useful packets will be received at the same rate, but the transmission will last longer in the second case since more useless packets are received (fig. 5).

In standard transport protocols used on the Internet, detecting congestion consists in detecting lost packets. On a satellite link, we have bursty losses, due to external factors such as bad weather conditions. When such errors occur, there is no need to reduce the transmission rate. This is why it is useful to be able to distinguish corruption losses from congestion losses. Moreover, we would like this mechanism to work without modifying the routers on the M-Bone, which means that it should be an end to end mechanism.

Packet loss is therefore not a good congestion signal. If we consider that the traffic generated by the source is negligible in regard with the total traffic induced by all the other users, variations of the Round Trip Time (RTT) is not a good congestion signal either [4]. Using an approach similar to that of packet pair rate control [5], we propose to send the data as a series of bursts and try to detect the "flattening of the burst" as a sign of congestion. But we work here in open loop and there are no acknowledgments for the sender to measure delays. It's up to the receiver to measure congestion. Moreover, with packet pairs, no measurement is possible when one of the two packets gets lost. Since the loss is not in itself a congestion signal, we lose information and therefore reactivity when corruption losses are frequent. This is why we do not send pairs of packets but bursts of $b = 7$ packets. This strategy however works only with FIFO routers, and when paths on

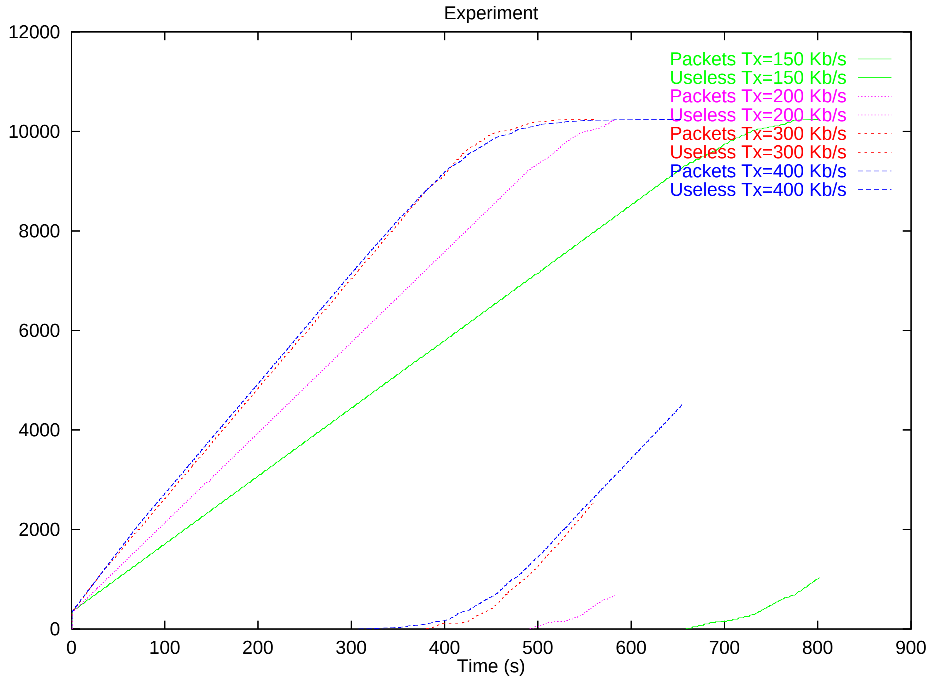

Figure 5: Useful and useless packets received over time for different receving rates

the MBONE are not too long.

The position of the packet within the burst is included in the packet's header. Let $D(p)$ be the inter-packet delay preceding packet p, $HB(t_a, t_b)$ be the group of packets that are at the head of a burst between dates t_a and t_b (i.e. position within the burst $= 0$).

- $Q_t = 1 - \frac{\sum_{p \in HB(t, t-\Delta_1)} D(p)}{\Delta_1} . \frac{B_s}{B_s - 1}$

  is the mean burst ratio observed by the receiver between dates $t - \Delta_1$ and t (i.e. percentage of time during which it is receiving the data). B_s is the number of packets in a burst. (Here we have $B_s = 7$). Δ_1 is the measurement time.

- $\hat{Q}_t = Inf_{u \in [t-\Delta_2, t]} Q_t$
  represents the shortest burst ratio observed since $t - \Delta_2$. We consider that we have observed this in ideal conditions (i.e. we were using all the bottleneck's bandwidth at that time).

- $R = \bar{Q}.n/N$ where

  - $\bar{Q}$ is the burst ratio chosen by the sender. It must be chosen large enough to have the bursts interfere with external traffic, but small enough to make a significant difference at the receiver's end between the congested state (no bursts) and the non-congested state (bursts). We chose $\bar{Q} = 0.33$
  - n is the number of subscribed channels,
  - N is the total number of channels.

  R represents the "equivalent" burst ratio chosen from the sender, considering that only the subscribed channels are sent through the bottleneck.

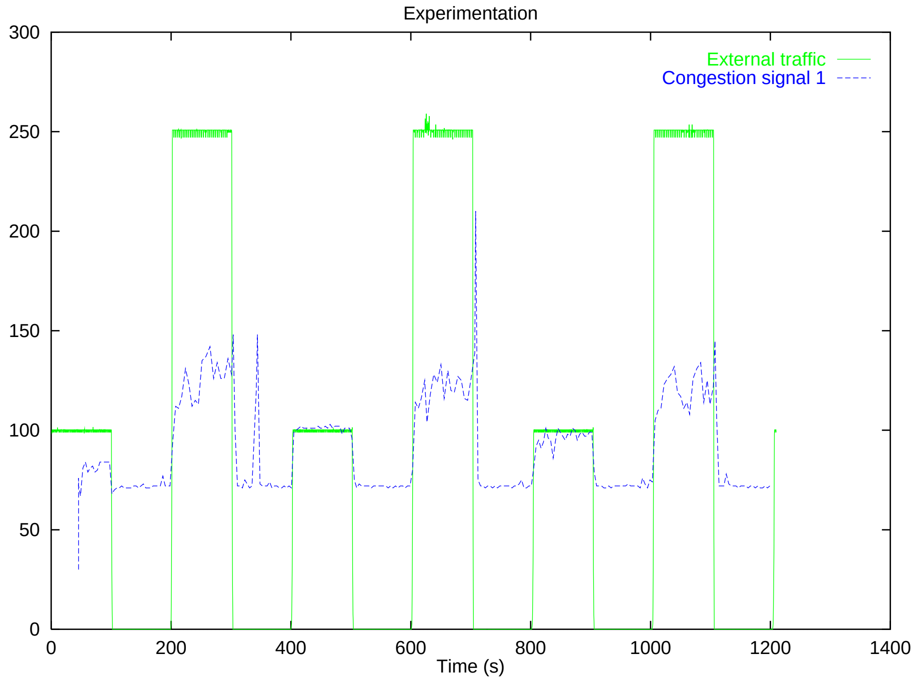

Figure 6: Behaviour of congestion signal S_t (in %)

- If X_t represents the external traffic's rate, I our traffic's mean rate, we want to approximate the portion of the bottleneck bandwidth B used (i.e. $S_t = \frac{X_t+I}{B}$).

$$\frac{X_t.R+I}{Q_t} = \frac{I}{\hat{Q}_t} = B \Rightarrow \boxed{S_t = \frac{Q_t+(R-1).\hat{Q}_t}{R}}$$

We then use

- $S_t > 1.05$ as a congestion signal
- $S_t < 1.00$ and $1 - \hat{Q}_t > 1.5/B_s$ as a no congestion and available bandwidth signal.

We then have a multiplicative decrease - additive increase scheme to join and leave layers. When the receiver has subscribed to layers 1 thru i, he tests for a congestion signal

- after $\frac{N}{log(r_{i-1}/r_i)}$ packets received and leaves layer i if congestion was detected.
- after $\frac{N}{r_{i+1}-r_i}$ packets received and joins layer $i+1$ if no congestion was detected.

N is chosen large enough to take into account join and leave delays, and not too large to have enough reactivity.

Figure 6 shows how our congestion signal evolves with varying external traffic and a constant bottleneck bandwidth. The bottleneck bandwidth is 300 Kb/s and the receiver subscribed to 200 Kb/s. The external traffic is sent between the same source and destination at a rate of 0, 100, and 250 Kb/s.

5 Conclusion

When trying to reliably send data to a large number of receivers using a satellite link and the M-Bone, we face a certain number of difficulties, due to the long delay, asymmetry, and high bit error rate of the satellite link, and the heterogeneity of the M-Bone. To solve the problem of asymmetry, we chose an a feedback-free approach, replacing retransmission requests with Forward Error Correction, and proposed a mechanism that enables receivers to detect congestion and then adapt their own receiving rate. In a heterogeneous environment, the data organization proposed here enables them to receive the data in an almost optimal time, taking into account late arrivals, and is still adapted to a good reactivity to congestion signals. Future work should be done on congestion detection and avoidance : increasing the robustness of the congestion signal, synchronizing join and leaves, enforcing correct behaviors, and testing how this behaves with other flows such as TCP connections.

References

[1] K. Miller, K. Robertson, A. Tweedly, and M. White, "Starburst multicast file transfer protocol (mftp) specification," tech. rep., January 1997. Internet Draft draft-miller-mftp-spec-02.txt.

[2] S. McCanne and V. Jacobson, "Receiver-driven layered multicast," in *Proceedings of ACM Sigcomm '96*, (Stanfowrd, CA), August 1996.

[3] L. Vicisano, L. Rizzo, and J. Crowcroft, "Tcp-like congestion control for layered multicast data transfer," in *Proceedings of IEEE Infocom '98*, March 1998.

[4] S. Biaz and N. H. Vaidya, "Distinguishing congestion and corruption losses : A negative result," tech. rep., August 1997.

[5] S. Keshav, "Packet-pair flow control," *IEEE/Transactions on Networking*. To appear someday. http://www.cs.cornell.edu/skeshav/papers.html.