DyRAM : a Reliable Multicast Protocol.

Moufida Maimour, Cong-Duc Pham

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

DyRAM : a Reliable Multicast Protocol

Moufida Maimour
Cong-Duc Pham

January 2003

Research Report N° 2003-05

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

DyRAM : a Reliable Multicast Protocol

Moufida Maimour
Cong-Duc Pham

January 2003

Abstract

Group communications (multicast) are foreseen to be one of the most critical yet challenging technologies to meet the exponentially growing demands for data distribution in a large variety of applications of the Internet (grid computing, web applications, distributed simulations...). When reliability is required, there is no straightforward solutions and meeting the objectives of reliable multicast is not an easy task. Active networks open a new perspective in providing more efficient solutions for the problem of reliability. In this context, routers are able to perform customized computations on the packets flowing through them. In this report, we propose a receiver-based (replier) local recovery multicast protocol with dynamic repliers elected on a per-packet basis. Designed to provide an efficient reliable multicast service without any cache facilities inside the network, our approach, uses low-overhead active services in routers. The current report addresses the design, evaluation and the implementation of an efficient and scalable reliable multicast protocol noted DyRAM standin for Dynamic Replier Active reliable Multicast.

Keywords: Active networks, Reliable multicast, Congestion avoidance, Evaluation, Analysis, Simulation, Implementation.

Résumé

Les applications qui utilisent le multicast fiable sont de plus en plus nombreuses. Un grand nombre de protocoles de multicast fiable ont été proposés ces dernières années. Les réseaux actifs ouvrent de nouvelles perspectives pour de nouvelles approches plus performantes. Dans ce rapport, nous présentons DyRAM, un protocole de multicast fiable basé sur la technologie des réseaux actifs. DyRAM utilise un recouvrement local en structurant les récepteurs dans une hiérarchie construite dynamiquement par paquet perdu, cela avec l'assistance de routeurs actifs dans l'arbre de multicast. Cet aspect dynamique permet un équilibrage des charges sur l'ensemble des récepteurs. DyRAM est conçu pour limiter les besoins de cache au niveau des routeurs tout en proposant des services actifs moins coûteux en temps de traitement. Dans ce rapport est présentée notre expérience dans la conception, l'évaluation et l'implémentation de DyRAM.

Mots-clés: Réseaux actifs, Multicast fiable, Contrôle de congestion, Evaluation, Analyse, Simulation, Implémentation.

DyRAM : a Reliable Multicast Protocol *

Moufida Maimour and Cong-Duc Pham.

1 Introduction

Motivations behind multicast are to handle one-to-many communications in a wide-area network with the lowest network and end-system overheads, mainly by avoiding redundant traffic on physical links with packet duplications as close as possible to the final destination. Multicast is foreseen to be one of the most critical yet challenging technologies to meet the exponentially growing demands for data distribution in a large variety of applications of the Internet (grid computing, video-conferencing, web applications, distributed simulations...). At the network level IP multicast provides an efficient one-to-many IP packets delivery but without any reliability guarantees. However data dissemination applications such as distributed computing or interactive simulations usually require a reliable transfer that requires that all packets are safely delivered to the destinations.

1.1 Adding reliability to IP multicast

Meeting the objectives of reliable multicast is not an easy task and, following the "IP vision", IP multicast (RFC 1122 and [3]) has no mechanisms for reliability. Therefore reliability has to be added at a higher layer. The problem of reliability in multicast protocols has been quite widely covered during the last 10 years. Early reliable multicast protocols use an end-to-end solution to perform the loss recovery. Most of them fall into one of the following classes : sender-initiated, receiver-initiated and receiver-initiated with local recovery protocols. In sender-initiated protocols, the sender is responsible for both the loss detection and the recovery. These protocols usually do not scale well to a large number of receivers due to the ACK implosion problem. Receiver-initiated protocols move the loss detection responsibility to the receivers. They use NACKs instead of ACKs. However they still suffer from the NACK implosion problem when a large number of receivers have subscribed to the multicast session. In receiver-initiated protocols with local recovery, the retransmission of a lost packet can be performed by some other nodes in the multicast tree [5, 14, 20, 13, 15, 7]. Most solutions now propose local recoveries to both avoid the implosion of control messages at the source and to reduce the recovery latency. There are basically 2 possibilities for enabling local recoveries : (*i*) use some receivers or dedicated servers as repliers (in some cases with a router-assisted solution), or (*ii*) use network elements such as routers for caching. In the first category, the replier could be any receiver in the neighborhood (SRM [5]), a designated receiver (RMTP [14], TMTP [20], LMS [13], PGM [15]) or a logging server (LBRM [7]) in a hierarchical structure.

1.2 Towards active reliable multicast

Local recoveries appear to be the most promising choice for achieving reliability on an open Internet network. Using a replier has the main advantage of requiring a memory usage as low as possible within network elements and is potentially more scalable. The choice of the replier can be done in several ways. Protocols that use some kind of router assistance are for instance LMS [13] and PGM [15]. LMS consists in adding more topology information with little help from routers. With some specific forwarding functions within routers, the protocol elects a designated replier (leader) for each subtree to respond to the requests

*Authors may be reached via e-mail at Moufida.Maimour@ENS-Lyon.Fr, and Congduc.Pham@ENS-Lyon.Fr This text is also available as a research report of the Institut National de Recherche en Informatique et en Automatique `http ://www.inria.fr`.

made by the end hosts. PGM also uses some router assistance (not yet active router but rather PGM router) to discover and elect repairers that must be on the path towards the source.

Going a step further, and gaining in generality and flexibility, the use of active network concepts [17] where routers themselves could contribute to enhance the network services by customized functionalities have been proposed in the multicast research community. Recently, the use of these active network concepts has been proposed in many research areas including multicast protocols [9, 8, 2], distributed interactive simulations [4] and network management [16]. There are many difficulties, however, in deploying in a large scale an active networking infrastructure. Security and performance are amongst these difficulties. However, active networking has the ability to provide a very general and flexible framework for customizing network functionalities in order to gracefully handle heterogeneity and dynamics. Contributing mainly on feedback implosion problems, retransmission scoping and cache of data, active reliable multicast offer a general and flexible framework for customized functionalities in network protocols. ARM [9], AER [8] and RMANP [2] are protocols that use cache of data packets to permit local recoveries and advanced NACK suppression strategies. ARM (Active Reliable Multicast) and AER (Active Error Recovery) are two protocols that were recently proposed in the research community and that use active services within routers. They differ in the strategy adopted for solving the NACK implosion problem. In ARM, a receiver experiencing a packet loss sends immediately a NACK to the source. Active services in routers then consist in the aggregation of the multiple NACKs. In contrast, AER uses a strategy similar to the one used by SRM and based on local timers at the receivers : prior to send a NACK packet, a receiver initiates a timer and waits for a random amount of time. In addition, an active router in ARM would send the repair packet only to the set of receivers that have sent a NACK packet (subcast). In AER, the active router simply multicasts the repair packet to all its associated receivers. In both protocols, the cache of data packets is performed within the network (at the routers in ARM and in servers co-located with the routers in AER) to permit local recoveries.

The approach we propose use local recoveries performed by repliers elected amongst a sub-set of receivers. Instead of an approximate solution based on timers as in SRM, or a complex DLR discovery as in PGM, the elected replier in our case is dynamically determined at each lost packet (without much overhead as it will be described later on), and not determined at the beginning of the multicast session, thus justifying the "dynamic replier" property of our active reliable multicast protocol (noted DyRAM). Since a specific replier can be chosen for each lost packet, it is therefore possible to have several logical subtrees at the same time for the recovery process. In addition, this very dynamic choice of the replier provide load balance features that reduces the receiver overhead. As opposed to LMS and PGM, DyRAM uses intensively the concept of active networking with active services within routers for implementing several advanced mechanisms including the replier election. This choice of active networking is motivated by the fact that the deployment on-the-fly of customized services appears to be more general and flexible than a solution based on a dedicated, static router (PGM routers for example). However, this choice leads to some design constraints where the active services must incur as low as possible execution/memory overhead within routers since the service will mainly be executed in software (dedicated routers can take the ASIC solution for achieving performance).

Therefore we can summarize the motivations behind DyRAM with the following design goals : (i) to minimize active routers load in order to make them supporting more sessions (mainly in unloading them from the function of data caching performed in ARM), (ii) to perform a more efficient replier link election procedure with active services, (iii) to dynamically distribute the recovery task among the downstream receivers to not overload one replier and (iv) to incur as low as possible execution/memory overheads.

The rest of the report is organized as follows. Section 2 gives an overview of the DyRAM protocol and its active services are described. A performance evaluation of DyRAM using simulation is the object of section 3. Section 4 shows how DyRAM is implemented in an active execution environment and preliminary results will be presented in measuring the raw processing time required for enabling low-latency multicast communications on the Internet. A brief overview on the congestion avoidance algorithm used in DyRAM is presented in section 5. Further details on DyRAM including memory management and flow control will be found in section 6 before concluding with some future directions.

2 DyRAM General Features

2.1 Description

DyRAM is a reliable multicast protocol with a recovery strategy based on a tree structure constructed on a per-packet basis with the assistance of routers. It is an incremental step from the existing propositions but it has been designed to provide low-overhead active functionalities with a dynamic replier election on a per-packet basis. The protocol uses a NACK-based scheme with receiver-based local recoveries where receivers are responsible for both the loss detection and the retransmission of repair packets when it is possible. In order to perform flow and congestion control as well as memory management in addition to a more efficient replier election, ACKs are piggybacked in the NACKs. In the absence of NACKs, ACKS are also periodically piggybacked by special messages called "Congestion Reports"(CRs). These CRs contain the necessary information for the congestion and flow control which are beyond the scope of this report. Nevertheless sections 6.3 and 5 summarizes how they are performed. For more details about flow control and congestion avoidance in DyRAM refer to [11]

Routers play an active role in DyRAM which consists (in addition to the services related to congestion avoidance and flow control) in the following active services :

- the NACK suppression of duplicate NACKs in order to limit the NACK implosion problem.
- the CRs aggregation thus avoiding the CRs implosion at intermediate nodes and at the source.
- the subcast of the repair packets only to the relevant set of receivers that have experienced a loss. This helps to limit the scope of the repairs to the affected subtree.
- the dynamic replier election which consists in choosing a link as a replier to perform a local recovery from a downstream receiver instead of caching data packets at the routers. Dynamic election provides robustness to host and link failures.
- the early loss detection of packet losses and the emission of the corresponding NACKs. This is very helpful in providing a low recovery delay.

In DyRAM, the source sends data packets to the multicast address subscribed to by all the receivers. The core functionalities of IP multicast are assumed to deliver the packets to the receivers in a best effort manner. A receiver detects a loss by sequence gaps and upon the detection of a loss, a receiver immediately sends a NACK toward the source and sets a timer. Since NACKs and repairs may also be lost, a receiver will re-send a similar NACK when the requested repair has not been received within the timeout interval. Practically, the timeout must be set to at least the estimated round trip time (RTT) to the source. In order to detect the loss of the last packets (no gap in sequence number in this case), a receiver will request the next packet which is supposed to be received if no data has been received within a predefined receiving time window. This window width depends on the transmission rate of the source. The last data packet is easily identified as the EOF (end-of-file) packet which will be sent by the source as the last one. Upon the reception of a NACK packet, the source sends the repair packet to the multicast address.

In order to limit the processing overheads of duplicate NACKs and to avoid the corresponding retransmissions, the source, the active routers and the receivers (remember that DyRAM uses repliers among receivers) ignore similar NACKs for a certain period of time. According to this rule, a NACK will be said "valid" if it is received for the first time or no similar NACKs are received during this discard time window. How this discard window and the other timeouts are adjusted and how the different round trip times (RTTs) are estimated is addressed in section 2.5.

2.2 Packets Structure

All the used packet types contain in their header the multicast address, the source address and the active service identifier (`SVC`) to be performed on it (see figure 1). A data packet is labelled uniquely by a sequence number and contains a dedicated field (isR) to distinguish an original transmission from a repair. For the purpose of flow control and memory management at the receivers, a data packet contains a field with the maximum sequence number of the data packet from which previous ones can be removed from the receivers memory (see section 6.3). A data packet contains also the current emission rate which is mainly useful to be known by the receivers in order to correctly update timeouts.

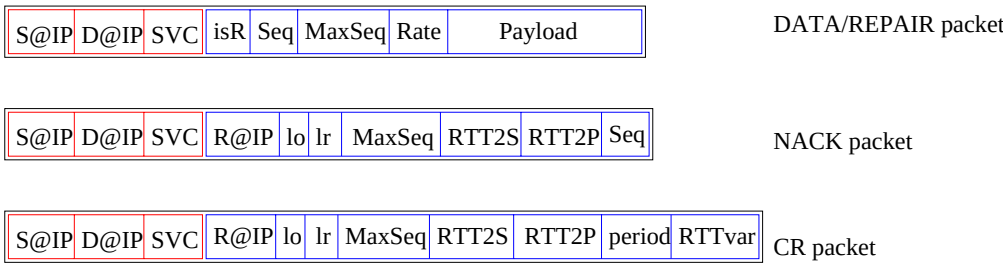


FIG. 1 – Packets structure.

A NACK or a CR packet include in addition to their originator address, their last ordered (lo) and last received (lr) data packet. Moreover, a receiver will report its RTT to the source and to its active router using the $RTT2S$ and $RTT2P$ fields contained in NACK and CR packets. Whereas a NACK contains the sequence number of the requested data packet, A CR contains two other fields which are the CRs current period and an aggregation of the RTT variation experienced by the subtree from which the CR is received. These two fields are used by the DyRAM congestion avoidance algorithm.

2.3 DyRAM Active Services

The active services related to reliability are described in this section. Congestion and flow control services are summarized in sections 6.3 and 5. More details about the congestion avoidance algorithm in DyRAM can be found in [11].

2.3.1 Router's Soft States

Within active routers, the different active services can be implemented simply by maintaining information about the data packets, NACKs and CRs received. This set of information is uniquely identified by the session source and the multicast address.

For each received NACK, the router creates or simply updates an existing NACK state (NS) structure which has a limited life time. A NS is removed on the reception of the corresponding repair. Hence its life time can be estimated as the maximum expected time required by the corresponding repair to be received by the router. This structure maintains for every nacked packet, a subcast list ($subList$) that contains the numbers of links from which a NACK has been received.

DyRAM also enables the routers to keep track of the received data packets in order to quickly detect packet losses occurring from upstream and affecting all its subtree. This can be done simply by maintaining a track list (TL) structure for each multicast session handled by the router. A TL has three components :

- $lastOrdered$ is the sequence number of the last data packet received in the order.
- $lastReceived$ is the sequence number of the last received data packet.
- $lostList$ contains the list of data packets not received by the router with sequence number greater than $lastOrdered$ and less than $lastReceived$. This list is empty when ($lastReceived < lastOrdered + 1$) and contains at least one element otherwise.

In order to perform quicker, accurate and optimized replier elections (as will be seen in section 2.3.5), we keep within a router a link state structure (LS) which contains for every downstream link three fields which are :

- $lastOrdered$ is the sequence number of the last data packet received in the order by this link (by all the receivers behind this link).
- $lastReceived$ is the sequence number of the last received data packet by this link.
- $weight$ is an assigned weight to this link which is proportional to its capacity to be a replier.

These information are retrieved from the CRs and NACKs received during the multicast session as will be explained in the next sections.

2.3.2 NACK suppression and CR aggregation

On receipt of a valid NACK for a data packet, a router creates the corresponding NS structure and initializes a timer noted NST (NACK Suppression Timer). During this timer, all subsequent NACK packets received are used to appropriately update the LS and NS structures and are dropped afterward. According to our definition of a valid NACK in section 2, a NACK is considered as valid if when it is received there is no active NST.

The LS structures are also updated on the reception of the CRs. All the CRs received from the downstream links are aggregated in one CR to be sent on the upstream link. The aggregated CR contains the sequence number of the minimum last ordered and the maximum last received data packets among those reported by the CRs received from downstream. More details on the CR aggregation and especially for the RTT variation field, can be found in [11].

2.3.3 Subcast Functionality

The subcast list in the NS structure is an important component for the subcast functionality. This list contains the set of links (downstream or upstream) from which a NACK has been received. When a data packet arrives at an active router it will simply be forwarded on all the downstream links if it is an original transmission. If the data packet is a repair packet the router looks for a corresponding NS structure and would send the repair on all the links that appear in the subcast list.

2.3.4 Replier election

The "replier election" is a key functionality which is specific to DyRAM. It allows local recoveries with support from the active routers instead of loading them by heavy caching functions.

On the receipt of a NACK for a data packet from a given link, a router would update the corresponding LS structure when necessary. If the received NACK is valid, a NST is initialized and a NS structure is created. Additionally, the router verifies (using the LS structure) if the requested data packet has already been acked by a downstream link. If so the NACK will be immediately forwarded downstream on one of the links that appear to receive this data packet. Otherwise a replier election phase is initialized by setting a timer noted DTD (Delay To Decide) in order to collect during this time window as much information as possible about the links affected by a loss (updates of the subcast list and the LS structure).

If the received NACK is not valid, the router updates the corresponding NS structure. If no DTD is active for the requested data packet then the NACK is simply dropped. Otherwise if all the downstream links appear to be in the subcast list then the NACK is immediately forwarded up. The LS structure could be useful to perform quicker and more accurate replier election. When a received CR (or a NACK) piggybacks an ACK for the lost data packet, the corresponding link is elected immediately as a replier and the DTD timer is canceled. If no event has triggered the replier election, this latter is performed when the DTD timer expires. A router would choose one of the downstream links which is not in the subcast list.

Once the election is completed, the router will forward the NACK downstream on the elected link. If this link ends to a receiver, then this latter will be the elected replier. Otherwise it ends to an active router which in turn will perform a replier election and so on until the NACK reaches a receiver. The elected replier would then unicast the repair packet to its upstream router which would in turn subcast the repair packet on all the links in the subcast list. To avoid the processing of the NACKs for a data packet for which a repair has already been sent, an active router ignores requests for this data packet during a given period of time.

2.3.5 Optimizing the replier election

The elected replier in our case is dynamically determined at each lost packet, and not determined at the beginning of the multicast session as in [13]. The replier link in [13] is only updated when the group membership changes or a receiver is no longer willing to act as a replier. Thus justifying the "dynamic replier" property of our protocol. To avoid overloading a particular downstream link, the router always try to choose a different link from the previously elected one (if available) by respecting a ring order among them, thus realizing when possible a load balance (in addition to provide robustness to replier and link failure).

Using the *weight* field of the LS structure, a more sophisticated replier election could be performed. In fact by mean of the *weight* field, a "replying capacity" is assigned to each downstream link. When the choice between multiple downstream links is present, we try always to choose the link that appears to be the most "capable" one in a way so the number of times a link is elected will be proportional to its assigned weight. To estimate the weight to be associated to say the ith link, we use the following normalized metric :

$$W_i = \frac{\min(RTTvar_{down} \times RTT_{down})}{RTTvar_i \times RTT_i \times (lr_i - lo_i + 1)}$$

The minimum in the numerator is taken among all the downstream links. $RTTvar_i$ varies inversely with the available bandwidth on the ith link. The RTT gives an idea on the receivers distance to the active router. As for $(lr_i - lo_i + 1)$, it can be seen as an estimation of the loss rate experienced by the subtree of the link i. Information about $RTTvar_i$, RTT_i, lr_i and lo_i are retrieved from the CRs (eventually from NACKs) received on the ith link. RTT_i is the RTT recently experienced by the closest receiver downstream the ith link.

2.3.6 Early loss detection

Earlier reliable multicast protocols put the burden of both the loss detection and the recovery at the sender side. These protocols are TCP-like and use ACKs combined with timers to detect losses. These ACK-based schemes violate the IP-Multicast Model [3] where the source is not aware of the identity of the receivers. The use of NACKs instead of ACKs moves the loss detection responsibility to the receivers. A receiver detects a loss on receipt of an out-of-order data packet or on a timeout expiration. When a loss is detected, the receiver would send a NACK to the source requesting the lost data packet. In DyRAM, like the receivers, active routers are also responsible for the loss detection. In fact active routers are capable to detect packet losses and consequently generate corresponding NACKs to be sent to the source.

An active router in a similar way to a receiver, would detect a loss when a gap occurs in the data packet sequence or on timeout expiration. The data service, in the absence of losses, simply consists in updating the track list. When a loss occurs, the router immediately generates a NACK packet towards the source. If the router has already sent a similar NACK for a lost packet then it would not send a NACK for a given amount of time (based on a timer). During this period, all NACK packets received for this data packet from the downstream links are ignored.

2.4 DyRAM Main Algorithms

This section presents in a synthetic manner the algorithms performed by the active routers in the multicast tree.

2.4.1 The Data Packet (DP) Service.

When an active router receives a data packet, it will execute the data packet service described below :

---

update the TL $trackList$ of the session
if DP is a repair **then**
  look for the corresponding NACK state structure NS
  send DP through links in $NS.subList$
  free NS
else
  forward DP on all the downstream links
  for each DP_{lost} with $((DP_{lost}.seq > TL.lastOrdered + 1)$ and $(DP_{lost}.seq \in TL.lostList))$ **do**
    send a NACK($DP_{lost}.seq$) to the source
    create or update the corresponding NS structure NS_{lost} :
    $NS_{lost}.seq \leftarrow DP_{lost}.seq$
    $NS_{lost}.subList \leftarrow$ all the links downstream
  end for
end if

2.4.2 The NACK Packet Service.

Provided that $B \geq 2$ where B is the number of the router's downstream links (if $B = 1$ then the NACK is simply forwarded upstream), an active router receiving a NACK packet from the link number i will perform the following actions :

---

update the corresponding LS structure if necessary :
$LS_i.lastOrdered \leftarrow NACK.lo$
$LS_i.lastReceived \leftarrow NACK.lr$
if NST is active **then**
  update the corresponding NS structure :
  $NS.seq \leftarrow NACK.seq$
  $NS.subList \leftarrow NS.subList \cup \{i\}$
  if DTD timer is active **then**
    if all the downstream links are in $NS.subList$ **then**
      forward the NACK upstream toward the source
      cancel the corresponding DTD timer
    else if $\exists j \neq i, LS_j.lastOrdered \geq NACK.seq$ **then**
      cancel the corresponding DTD timer
      forward the NACK on link j downstream
    end if
  end if
  drop this NACK
else
  initialize NST
  create the corresponding NS :
  $NS.seq \leftarrow NACK.seq$
  $NS.subList \leftarrow NS.subList \cup \{i\}$
  if $(\exists j \neq i, LS_j.lastOrdered \geq NACK.seq)$ and (DTD is active) **then**
    cancel the corresponding DTD timer
    forward the NACK on link j downstream
  else
    initialize a DTD timer
  end if
end if

2.5 Estimation of the RTTs and Timeouts Setting

In our active reliable multicast protocol DyRAM, many of its functionalities depend on appropriately setting the timers. These timers values depend on the different RTTs between the tree nodes. In this section, we first report our method to estimate the RTTs and secondly present how the different timeouts are set.

2.5.1 RTT estimation

A straightforward solution to estimate the RTTs between each receiver and the source consists in sending ping messages periodically from the receivers to the source. This latter will be overwhelmed in the presence of a large number of receivers. In DyRAM instead of estimating directly the RTTs between the source and each receiver, we begin by estimating the RTTs between every node and its parent in the multicast tree. To illustrate our method we show in figure 2, one path of a multicast tree that connects the source (labeled by 0) to one of the group receivers (with the label n). Intermediate nodes which are labeled by numbers from 1 to $n-1$ are the routers. The generalization of our method to the other receivers and intermediate nodes is straightforward.

Heartbeat messages (HB and HB_RESP) are periodically exchanged between each node i and its parent $(i-1)$ in the same way as the ping/pong messages, instead of sending the HB messages until the source. The first HB messages are started to be sent by the receivers. The reception of a HB by a router triggers the emission of a HB message to its parent in the multicast tree in addition to responding with a HB_RESP message.

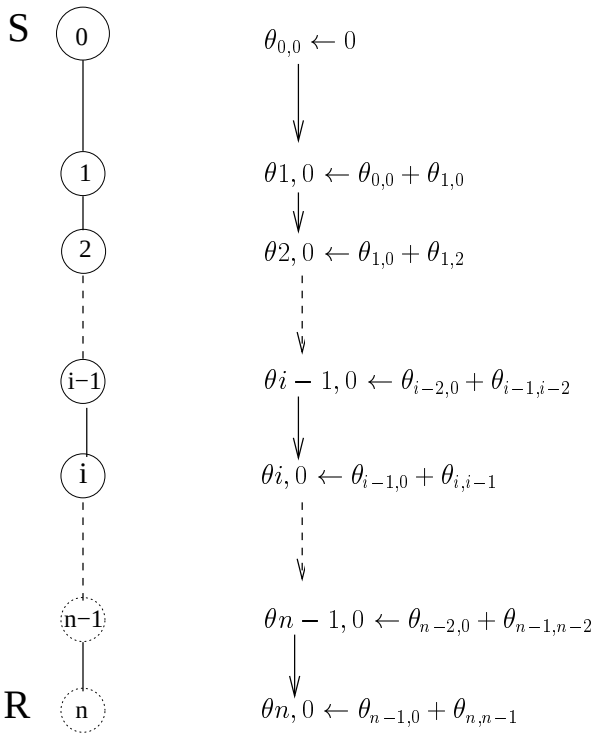


FIG. 2 – RTT estimation

For the following, we note $\theta_{i,j}$ the RTT between the ith and the jth node. Recall that $\forall i,j : \theta_{i,j} = \theta_{j,i}$. Consequently, $\theta_{i-1,i}$ is the RTT between the node i and its parent and $\theta_{0,i}$ is the RTT between a node i and the source.

A node i, in order to compute its RTT to the source, first computes its RTT to its parent node by sending a HB message timestamped with the emission time. The parent node, on the reception of the HB message, responds with a HB_RESP that contains the parent's ith node RTT to the source already computed. The node i, on receipt of the HB_RESP message is able to compute its RTT to the source as the sum of the RTT of its parent to the source and its own RTT to its parent as follows :

$$\forall i > 0 : \theta_{0,i} = \theta_{0,i-1} + \theta_{i-1,i} \tag{1}$$

Initially and for a first estimation, the source multicasts a special message HB_RESP_S which contains its own RTT to itself (of course we have $\theta_{0,0} = 0$). The node number 1 will update its $\theta_{0,1}$ using (1) and forwards it downstream. The node number 2 will in turn update its $\theta_{0,2}$ using the same equation and forwards it downstream and so on (figure 2). Afterwards the use of the HB and HB_RESP messages will be sufficient to estimate the different RTTs.

2.5.2 Timeouts Setting

If all routers at the edge of the core network are active, our protocol performs well with respect to the different defined metrics as will be seen in the section 3. The feedback implosion is avoided by performing NACK suppression at the routers. Duplicate repairs disappear as a consequence of the NACK suppression service. In fact, if all routers are active and have sufficient memory to create a NS structure for each received NACK without replacing anyone of them unless the corresponding repair is received then the source or an elected replier will receive at most one NACK per loss. Otherwise, the sender or a receiver respond to the first NACK. Then subsequent NACKs for this packet are ignored for a fixed amount of time. This discard time window must be well adjusted. A very small time window will produce duplicate repairs and a very

large one will increase the loss recovery delay due to the fact that lost repairs are not quickly detected. This time window depends on the farthest receiver in the group. For our simulations this time window is set to $(\max(RTT_{down}) - \min(RTT_{down}))/2$ where $\max(RTT_{down})$ and $\min(RTT_{down})$ are the maximum (respectively minimum) measured RTT between the source and any other receiver in the multicast tree. We divide by 2 to have a mean value (it is worth saying that an EWMA filter can be used). This timeout as has been discussed is not really required because of the NACK suppression hence it can be set to a minimum value.

Since NACKs and repairs may also be lost, a receiver will re-send a similar NACK when the requested repair has not been received within the timeout interval. This timer used by a receiver to retransmit a new NACK when a repair is not received is at least set to its RTT to the source. To detect the loss of the last data packets, a receiver uses a timer which is set to at least $1/rate_{current}$ where $rate_{current}$ is the current rate of the emission by the source. The current rate is known by each receiver thanks to a dedicated field in a data packet. This field is also used by the congestion avoidance algorithm associated to DyRAM (see section 5 for a summary and the technical report [11] for more details).

As described in section 2.3.4, active routers perform the election of a replier link on a per-packet basis. In order to take the best decision about which links may be the replier link, the DTD timer has to be well chosen. Neither a very small nor a very large one is recommended. With a very small one, a router will not be able to elect the appropriate link as a replier. A very large DTD will increase without justification the recovery latency. A good value for the DTD timer depends on how soon the router expects to receive all the NACK from the downstream link. The DTD timer is set using :

$$DTD = \delta \; DTD_{min} + (1 - \delta) \; DTD_{max}$$

where

$$DTD_{min} = (\max(RTT_{down}) - \min(RTT_{down})/2$$

$$DTD_{max} = DTD_{min} + period$$

where DTD_{min} (DTD_{max}) is the minimum (maximum) estimation of the required time to receive the necessary information to be able to decide which link will be the elected one. *period* is the current period for sending CRs (see section 5). $\max(RTT_{down})$ $(\min(RTT_{down}))$is the maximum (minimum) RTT between this active router and any receiver in its subtree.

An active router A ignores requests for a data packet if a repair has recently been forwarded for approximately $RTT(A, R_x)/2$ where R_x is the farthest receiver in the subtree of A. In addition, if the router is loss detection-capable, then a NACK could be resent if a requested repair is not received yet. When a NACK is sent, a router sets a timer which the expiration triggers the emission of a new NACK. This timer is set (as for the receivers) to at least its RTT to the source. During this time, all NACK packets for this data packet from the downstream links are ignored.

The different timeout values are smoothed using an EWMA filter as follows :

$$V = g \; M + (1 - g) \; V$$

where M is the newly obtained timeout value from the newly measured RTTs and V is the current estimated one. g is a constant to be chosen from the interval $[0.1, 0.2]$.

3 Simulation Results

We implemented a simulation model of DyRAM (in the PARSEC language developed at UCLA [1]) and evaluated its performances on a network model. The network model considers one source that multicasts data packets to R receivers through a packet network composed of a fast core network and several slower edge access networks. We will call *source link* the set of point-to-point links and traditional routers that connects the source to the core network. Similarly, a *tail link* is composed of point-to-point links and routers connecting a receiver to the core network (see Figure 3). We only consider active routers at the edge of the core network. This is due to the fact that the core network is reliable and a very high-speed network.

Adding complex processing functions inside the core network will certainly slow down the packet forwarding functions.

At the beginning, simulations are performed for a basic DyRAM protocol without the loss detection service which is afterwards integrated to perform an other set of simulations. Both of the two sets of simulations, the replier election is performed with respect to a ring order without any optimization.

3.1 Metrics

To evaluate our protocol we have defined a set of metrics. Firstly, M_S is defined as the number of retransmissions from the source per packet and gives an idea on the load at the source.

$$M_S = \frac{Number\ of\ retransmissions}{Number\ of\ sent\ packets}$$

To quantify the load at the network level, we use BW which is the average bandwidth consumed per link during the multicast session :

$$BW = \frac{Bw_{NACK} + BW_{Data}}{N.B + l_b.(N+1)}$$

where BW_{NACK} and BW_{Data} are respectively the bandwidth consumed by the NACK packets and the data packets during the multicast session :

$$BW_{NACK} = \sum_{i=1}^{NackNb} l_{NACK_i}$$

$$BW_{Data} = \sum_{i=1}^{DataNb} l_{DP_i}$$

where l_{NACK_i} and l_{DP_i} are respectively the number of links crossed by the ith NACK packet and the ith data packet.

The third metric is the completion time per packet which is the required time to successfully receive the packet by all the receivers (can be seen as the latency).

For all the simulations, we set $l_b = 55$. A NACK and a data packet are considered to be of 32 and 1024 bytes respectively. All simulation model values are normalized to the NACK transmission time (e.g. the time required to send or receive a NACK is set to 1, for a data packet this time is set to 32). For the processing overheads at the routers, we assume that both NACKs and data packets are processed in 32 time units. These values derived from measures in [9]. In our simulation models, we have taken into consideration the fact that the repairs may also be lost.

We verified that the three metrics defined above behave in the same manner and are dependent on each other. First we will show the benefit of performing the local recoveries from the receivers. Then a comparison between our approach (no cache in routers) and ARM (with cache in routers, no repliers) will be presented. Finally, we will consider the case when DyRAM also benefits from some cache amount in routers in the multicast tree.

3.2 Simulation results without the Loss Detection Service

The network model used in this first set of simulations considers active only at the receivers side. Each active router A_i is responsible of B receivers noted $R_{i1}, \cdots, R_{iB}$ forming a local group. We assume that there is l_b backbone links between the source and every active router A_i.

In our loss model, we consider both the spatial and the temporal correlation of data packet losses. The spatial correlation is introduced by considering a per-link loss rate. The core network is considered reliable as mentioned previously. For the other links (the source link or the tail links), the loss probability is noted p_l. Therefore, the end-to-end probability of a packet loss perceived by a receiver is $p = 1 - (1 - p_l)^2$. The losses at the tail links are assumed to be mutually independent. To take into consideration the temporal correlation of losses, we use the Markov chain model proposed in [18]. Let M be the transition matrix $M(i, j) = P[j/i]$

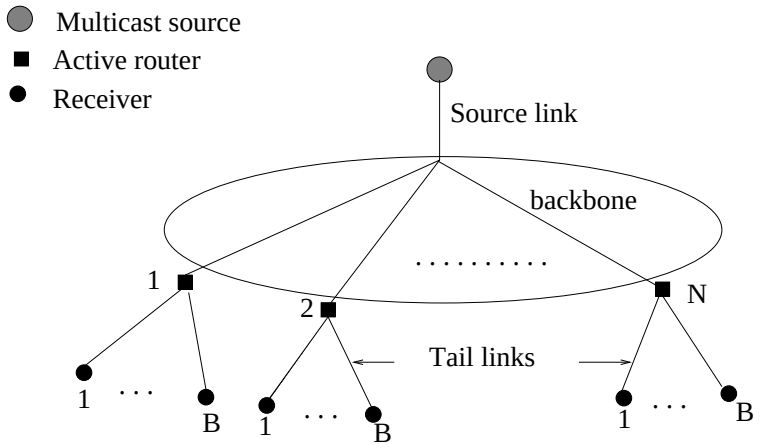
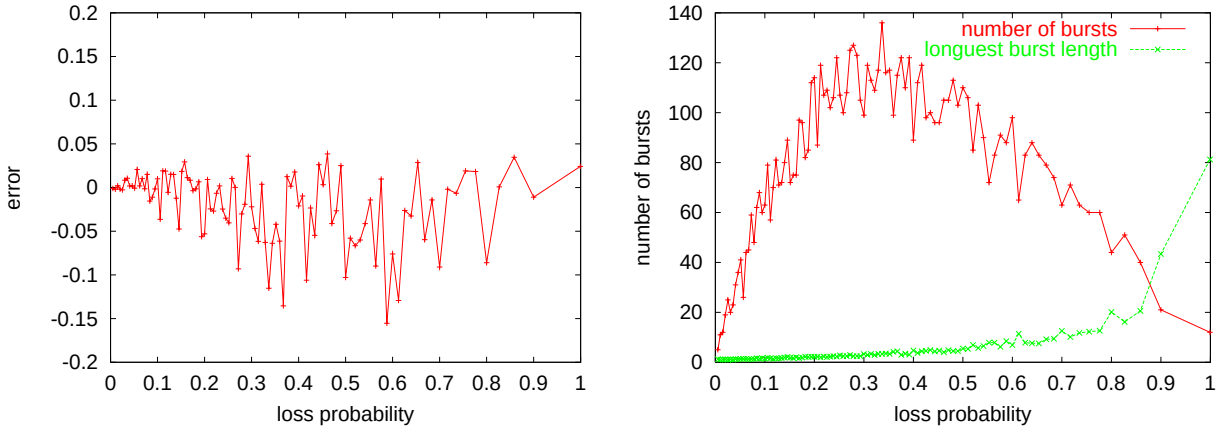


FIG. 3 – Network Model.



(a) (b)

FIG. 4 – The temporal correlation.

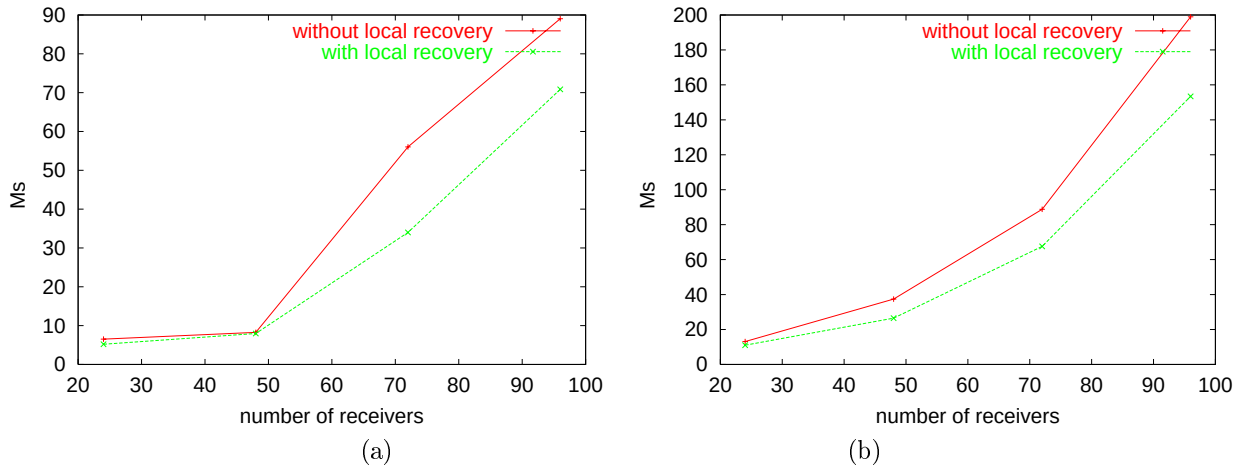


FIG. 5 – Load at the source in DyRAM with and without local recovery (a) $p = 0.05$, (b) $p = 0.25$.

where $P[j/i]$ is the probability to be in state j knowing that the current state is i. Our simulations showed that using the M matrix proposed in [18], p_l is of about 0.9154. In order to be able to perform simulations with different values of p_l, we introduced the M_λ matrix defined by :

$$M_\lambda = \frac{1}{\lambda} \cdot M$$

Having p_l as an input, λ can be computed using :

$$\lambda = \frac{b}{p_l} + a$$

where $a = 0.8068$ and $b = 0.1768$. Figure 4(a) shows that with this loss model, we achieve in most cases the targeted loss rate. Moreover the losses occur in bursts whose length increases with the loss probability as shown in figure 4(b).

3.2.1 Local Recovery from the Receivers.

Figure 5 plots for DyRAM the number of retransmissions (M_S) from the source as a function of number the receivers, with and without local recovery, for 5% and 25% loss rates. These results have been obtained from simulations of a multicast session with 48 receivers distributed among 12 local groups. The curves show that the local recovery introduces less load at the source as the number of receivers increases, especially for high loss rates. Even not shown, the consumed bandwidth and the completion time are reduced in about the same proportions.

Putting the recovery process in the receivers requires at least 2 receivers per active router otherwise local recoveries can not be realized. Therefore the local group size (B) is an important parameter. In order to study the impact of B simulations are performed with 48 receivers distributed among groups of different sizes. Figure 6 compares the per-link consumed bandwidth ratio of DyRAM to an other approach without any local recovery facilities as the group size is varied. As the receivers number per group increases, the consumed bandwidth is reduced with larger ratios for large loss rates. In fact when the group size is larger, there are more chances that the members of the group can recover from each other. For instance, figure 6 shows that with only 6 receivers per group and local recovery we achieve a gain of 80%.

3.2.2 DyRAM vs. ARM.

In this section, we compare our approach to ARM. Figure 7 shows for different loss rates, the ratio of the consumed bandwidth (BW), the load at the source in term of number of retransmissions and the completion

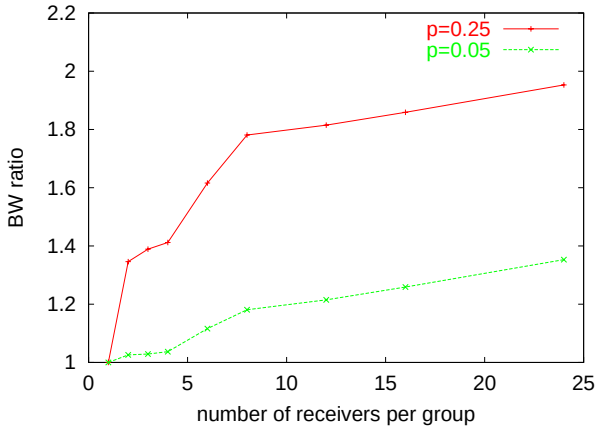


FIG. 6 – Consumed bandwidth ratio in DyRAM with and without local recovery as a function of the group size.

time for ARM and DyRAM. We took into consideration the percentage of the available cache at the ARM routers. Without any cache at the routers, DyRAM always performs better than ARM. When ARM benefits from caching at the routers, it performs better only for very low loss rates or large cache size at the routers. However for large loss rates (e.g. 25%), ARM requires more than 50% of cache at the routers to perform better than DyRAM. Our approach is more promising because even without any caching overhead at the routers DyRAM still has better performances than ARM when the latter can use more than 50% of cache.

3.2.3 DyRAM combined with cache at the routers.

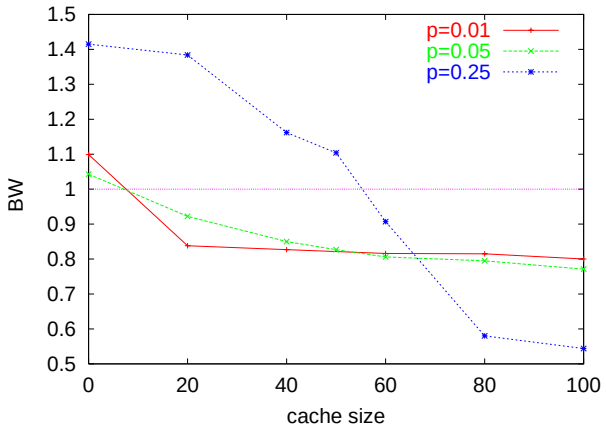
Designed to provide an efficient reliable multicast service without any cache within active routers, the results presented so far show that DyRAM does not provide better performances than an ARM-like protocol when the loss rate is small. However DyRAM performs much better for large loss rates without requiring higher overheads.

The results we present now assume that some cache memory are available in routers in addition to the DyRAM native local recovery strategy. One interesting question is : of how much DyRAM's performances can be enhanced ? To answer this question, figure 8 plots for different loss rates the gain in consumed bandwidth achieved by ARM and DyRAM when the cache size is varied over a simple ARM without caching in routers. The main results are that with cache facilities at the routers for both protocols, ARM never performs better than DyRAM even for low loss rates. When no cache is available at the routers, DyRAM outperforms ARM and this for any loss rate. As p increases, DyRAM performs better and better. In fact for 25% loss rate, ARM needs more than 50% of cache to achieve the same level of performances than DyRAM without any caching at the routers. To obtain a gain of 20%, ARM must benefit from 40% of cache. Instead, DyRAM without any cache already performs 40% better and with 40% of cache, it performs 60% better. One different way to look at the figures is that if DyRAM has only 40% of cache available then it already performs better than ARM with more than 60% of cache (for 1% and 5% loss rate). With 80% of cache, DyRAM has the same level of performances than ARM with 100% of cache available. Stated differently, about 80% of cache at the routers gives DyRAM the same features than a protocol with an infinite storage capacity.

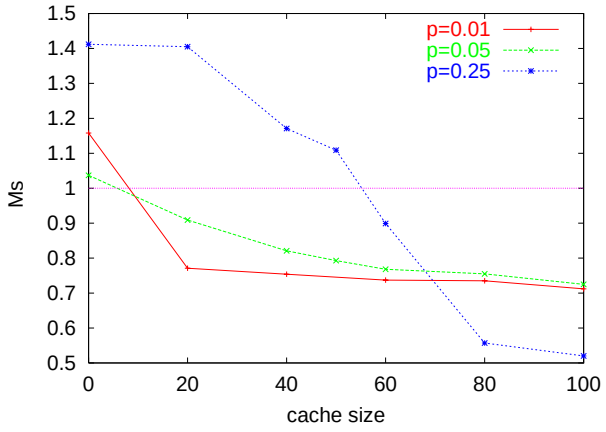
3.3 Adding the Loss Detection Service to DyRAM

Here we will consider the loss detection service in our simulations. Although simple, moving the loss detection into routers arises many questions such as "where to place such a service so it will be more efficient ?" and "what is the overhead of such an additional service ?". A previous analytical study [12] of this active service has shown that one must be careful when doing so in order to get any real benefit.

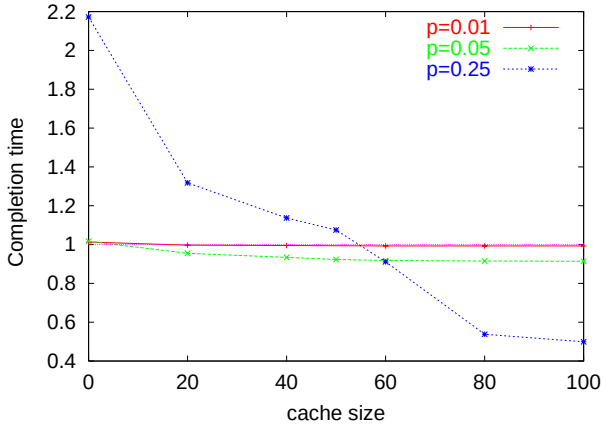
The analytical study was performed using a simple network model with a two level multicast tree and consists in a delay analysis. One source multicasts data packets to one group composed of B receivers $R_1 \ldots R_B$



(a)

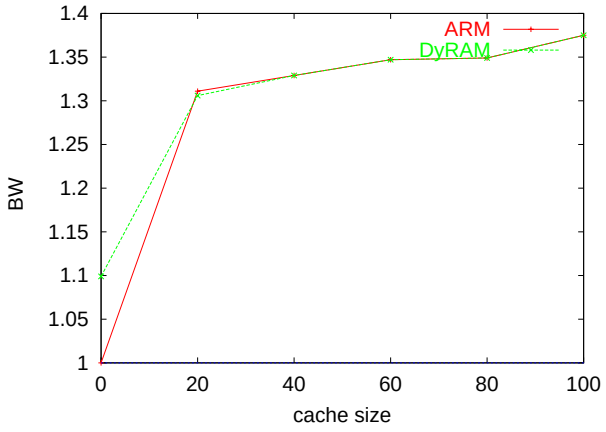


(b)

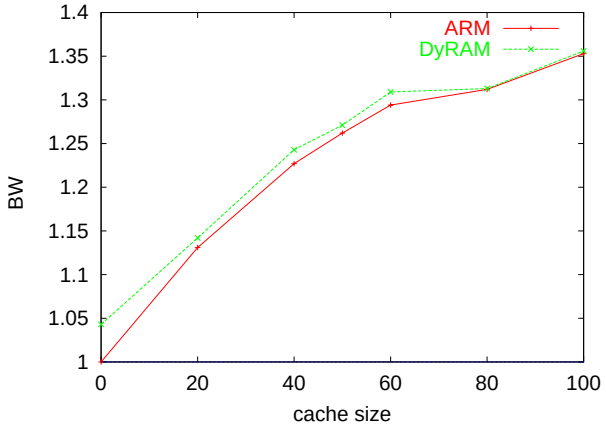


(c)

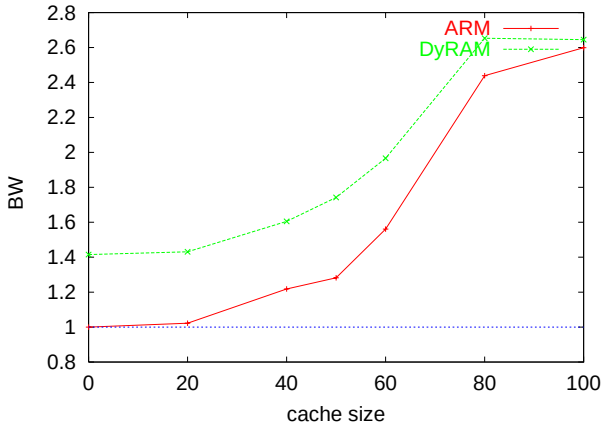
FIG. 7 – DyRAM vs. ARM (a) Consumed bandwidth, (b) Load at the source, (c) Completion time.



(a)



(b)



(c)

FIG. 8 – Gain obtained by DyRAM and ARM when both benefit from cache in the routers (a) $p = 0.01$ (b) $p = 0.05$, (c)$p = 0.25$.

connected to the source via an active router A. The computational framework adopted in our analysis is similar to the one provided in [19]. Each node is modeled by a M/G/1 queue (Poisson arrivals and arbitrary service time distribution). The delay analysis is largely based on the mean waiting time of the system. In order to estimate this mean waiting time for each node under the evaluated protocols, we proceed as follows. First the different flow rates $\lambda_1, \lambda_2, \ldots, \lambda_n$ of the node with their respective service requirement $X_1, X_2, \ldots, X_n$ are determined. Provided that each of these random variables have means and second moments, then the load ρ at this node can be computed using :

$$\rho = \sum_{i=1}^{n} \lambda_i E[X_i]$$

Finally, the mean waiting time $E[W]$ can be computed using the Pollaczek-Khinchine mean value formula as follows :

$$E[W] = \frac{\sum_i \lambda_i E[X_i^2]}{2(1-\rho)}$$

In order to evaluate the impact of adding a loss detection service, we consider two protocols noted A and D. Both of them benefit from the NACK suppression and the subcast services but D, in addition, benefits from a loss detection service at the active routers which are able to detect a gap in a data packet sequence. In this case, the router would immediately generate a NACK packet toward the source and would initialize a timer. On expiration of this timer, without having received the data packet, the router will send another NACK. All NACK packets received for this data packet from the downstream links are ignored until the expiration of the corresponding timer.

In order to consider the detection-capable active router's position, we introduce $\alpha \in]0,1[$ which is the ratio of T_{ar} to T_{sr}. These two parameters are expressed as a function of the number of links crossed by a packet. The link propagation time is set to 1 time unit. Therefore we can use T_{ar} (respectively T_{sr}) to represent the number of links between the router (respectively the source) and any receiver. We will also consider two particular values of α : $\alpha_c = (T_{sr}-1)/T_{sr}$ where the router is one link far from the source ($T_{sr}-1$ links between the router and a receiver) and $\alpha_f = 1/T_{sr}$ where the router is one link far from a receiver.

Figure 9 shows the performances of D as a function of the router position. It plots the ratio of the achieved delay by D when we change the router position (parameter α) to the case where $\alpha = \alpha_f$. We can see that the position of the router with respect to the source is deciding. The gain is increased when the router is the closest to the source. The gain in delay observed is more important for high loss rates. In fact, for a loss rate of 50% the delay is improved by 25% up to 35% if we move the router near the source instead of putting it near the receivers.

As a conclusion one must be careful about where to place a loss detection capable-router. This latter must neither be too far from nor too close to the source. When the router is closer to the receivers, the load introduced by the loss detection service is unjustified because of the long distance to be crossed by the generated NACKs by the router. When the router is put sufficiently far from the source, we maximize the number of losses that can be detected. When the position of the active router is well chosen, we showed that D performs better than A especially for high loss rates.

A set of simulations are performed to show how a loss detection service could decrease the delay of recovery of the DyRAM framework. Let $DyRAM^+$ be the enhancement of $DyRAM$ by adding the loss detection capability. To do so, the four protocols A, D, $DyRAM$ and $DyRAM^+$ are simulated on a network model in figure 10 similar to the adopted one in figure 3 except that a router close to the source (the source active router A_S) is considered. All the active routers are assumed to perform at least two active services, the NACK suppression service and the subcast functionality. Based on our analytical results, the source router is the best candidate to perform the loss detection service in addition to the two previous services. The other active routers should only perform the replier election service as seen in the previous section.

Figure 11 plots the recovery delay (normalized to the RTT) for the four protocols as a function of the number of the receivers for different loss rates. First of all, it is noticeable that protocol $DyRAM$ and $DyRAM^+$ (both with local recovery from the receivers) always perform better. For instance, we can see in figure 11(a) that $DyRAM$ is 10 times faster than A for a loss rate of 5%. Now, when the loss detection service

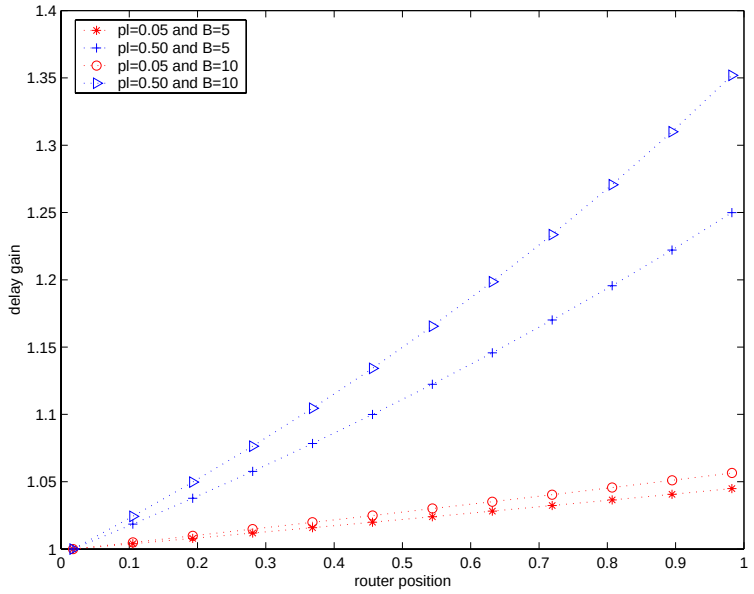


FIG. 9 – The delay gain in D as a function of the router position.

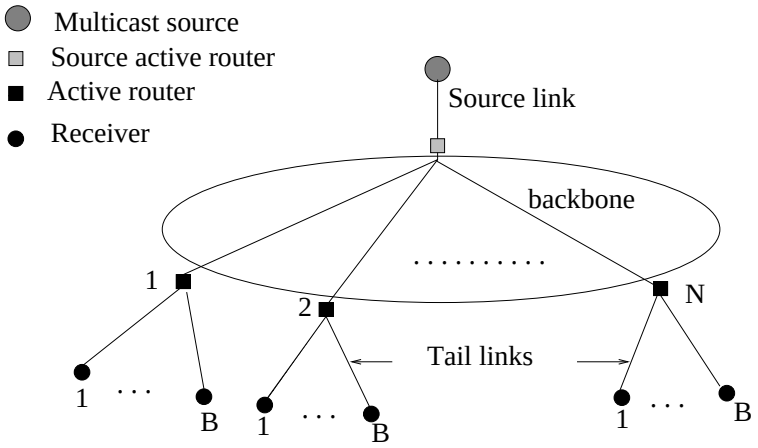
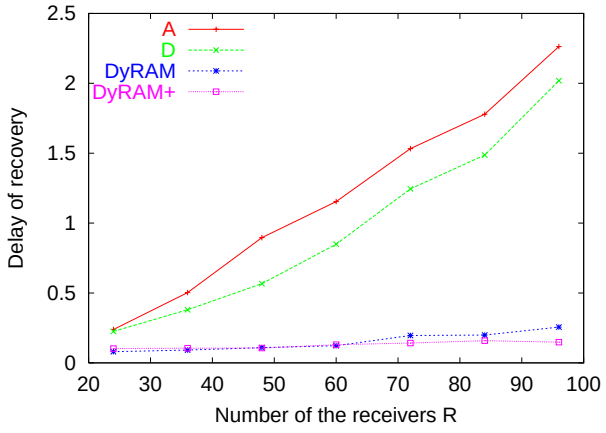
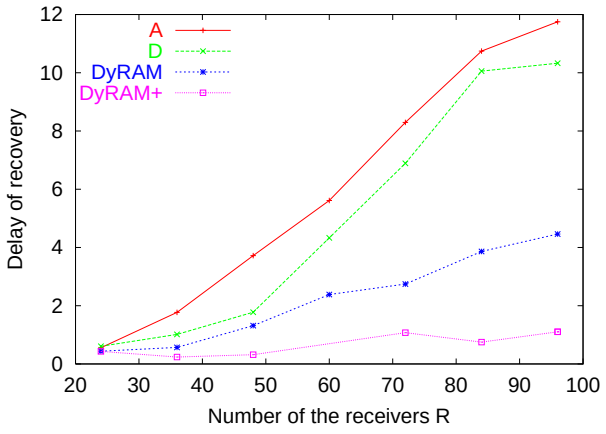


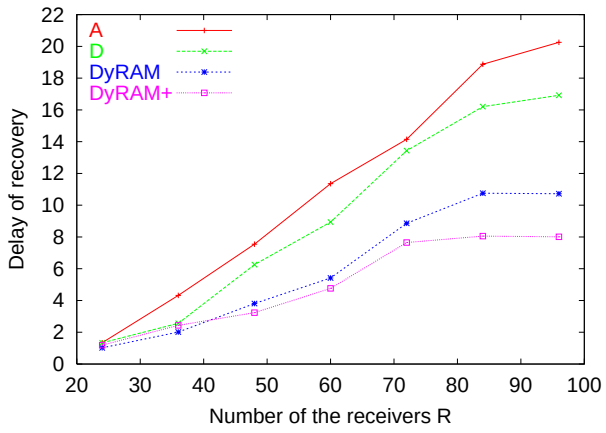
FIG. 10 – Network model.



(a)



(b)



(c)

FIG. 11 – The recovery delay with (a) $p = 0.05$, (b) $p = 0.25$ and (c) $p = 0.50$.

is applied to A (giving protocol D) the recovery delay can be reduced. In fact as we can see for the different loss rates, D always performs better than A thanks to the loss detection service. When applying the loss detection service to $DyRAM$, the delay of recovery decreased mainly for high loss rates and a large number of receivers. For instance, the loss detection service allows DyRAM to go 4 times faster for 96 receivers and a loss rate of 25%. We can also notice in figures 11(a)(c) that $DyRAM$ performs slightly better than $DyRAM^+$ when the number of receivers is small. Therefore it is unjustified to perform the loss detection service for a small number of receivers since the local recovery is sufficient to reduce the recovery delay. This does not appear to be a limitation of the loss detection service since a multicast session has generally to support a large number of receivers.

4 Implementation and Experimental Measurements

The performances of DyRAM have been extensively studied in the previous section by simulation. Results show that local recoveries associated to early loss detection succeed in providing low latencies. However, given the strong reliance of DyRAM on active services, it is crucial to determine how costly these services are. In this section we report our experience in prototyping DyRAM on top of the TAMANOIR execution environment [6]. At this point, we must mention that the purpose of this work is to quantify the raw processing time required for each elementary service and that neither timer optimizations nor end-to-end recovery latencies are investigated.

4.1 The TAMANOIR Execution Environment

TAMANOIR is a new active networking framework that deals with high performance active networks based on compiled JAVA with multi-threading approach to benefit both from performances and portability of services.

A TAMANOIR Active Node (TAN) is a persistent daemon acting like a dynamic programmable router. Once deployed on a node, it is linked to its neighbors in the active architecture. A TAN processes incoming packets with the appropriate service (launched via a *service manager* in a dedicated thread) before forwarding it. In TAMANOIR, a service is a JAVA class containing a minimal number of formatted methods. Each service is inherited from a generic class called simply *Class Service*. This is done in order to simplify the design of future services and especially to allow the TAN to download dynamically new classes.

In each packet we find a label (or a tag) as an identifier for the last TAN crossed by the packet. Therefore, if a TAN does not own the appropriate service, a downloading operation could be performed. This can be done by downloading the service either from the transmitting application or an other TAN in the neighborhood. Downloading services from a service broker is also possible.

4.2 An IP address-based Replier Election

DyRAM is initially intended to rely on a network multicast facility such as IP multicast. However, as an alternative to IP multicast, it is possible to have active services realizing level 3 multicast functionalities. Once a group has been formed relying on IGMP for instance, DyRAM runs its own simple session protocol to gather additional topological information at the DyRAM level to enhance the group anonymity imposed by IP multicast. A router mainly gathers information on the number of receivers or active routers directly connected to it with their IP addresses.

For our first DyRAM prototype, we adopted an implementation based on the use of IP addresses instead of link information, We need to gather some information at the beginning of the multicast session. Consequently an initialization phase is required. At the beginning of the multicast session, the source sends to the the hole group an initialization message (INIT). Upon the reception of this message, a receiver will respond with an INIT_RESP message that contains its IP address.

The first router that receives an INIT message will mark it as already received by at least one active router of the multicast tree. This first crossed router will behave as a source router with the loss detection facility as proposed in section **??**. Moreover such a router will forward immediately a NACK to the source instead of wasting time in a replier election procedure.

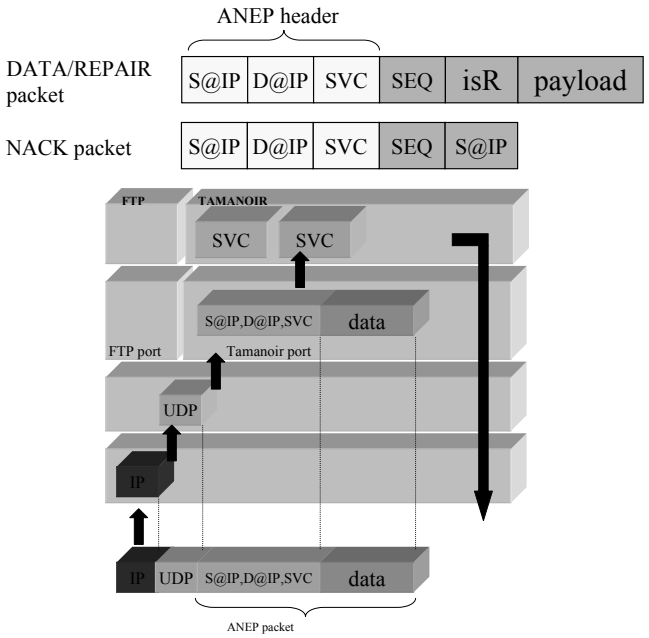


FIG. 12 – Packet format and TAMANOIR architecture.

Each router of the first level (just upstream a receiver) retrieves the IP address of the downstream receivers from their INIT_RESP messages. Afterwards, a first level router will forward one INIT_RESP message while the others will be dropped. An ith level router (with $i \geq 2$) retrieves one IP address per downstream link from the INIT_RESP messages it receives.

In such a manner, the source would receive just one INIT_RESP message which will be dropped. A timer could be initialized which the expiration triggers the emission of the HB_RESP_S message (described in section 2.5). Motivations behind waiting for a given amount of time is to give time to the other nodes to have a first estimation of their RTTs to their parents.

INIT and INIT_RESP messages may be lost. This is why the source, when it sends an INIT message, sets a timer. The expiration of this timer without the reception of the INIT_RESP message, triggers the emission of a new INIT message. CR messages are also used to update periodically the IP addresses kept in a router for the purpose of the replier election. This is done to quickly react to eventual topology changes.

4.3 Packet Format and Data Structures

DyRAM uses the ANEP format for the packets which is the one handled by TAMANOIR. In the ANEP header appears the multicast and the source addresses in addition to the service identifier. The other fields are inserted in the payload. DyRAM is implemented on top of the UDP protocol since the reliability is insured by DyRAM itself. We can see in figure 12 that incoming DyRAM packets are given to the TAMANOIR execution environment which selects the appropriate active service based on the `SVC` (service identifier) field. in this figure we do not show all the fields, many of them are omitted.

In this first DyRAM version, the TL structure is implemented as a double-linked list of bits (based on an array implementation) that allows fast insertion/suppression of bits at the beginning/end of the list. The NS structure is handled with a hash table with the sequence number of the lost data packet as the hash key. For this first prototype, we chose to use IP addresses of the receivers instead of the links information. Each entry refers to a double-linked list (*subList*) of IP addresses . In the current implementation, timers are handled by threads and a hash table of timer threads is used to maintain the list of timers. We will see later on that it may be not the best design choice.

4.4 Test-bed and scenario

Experimentations were performed with a first prototype using a test-bed that consists in a set of receivers and 2 PC-based routers (see table 1 for the characteristics of the used machines : Pentium II 400MHz, 512KByte cache, 128MByte RAM) running a Linux 2.4 kernel and Java version 1.3.1. In order to accurately measure the time spent in each portion of the active services, we use the UNIX `gettimeofday` system call to achieve a microsecond resolution.

Machine	IP address	CPU	RAM	cache
stan	140.77.15.21	AMD Athlon 1GHz	1GB	256KB
ike	140.77.15.12	AMD Athlon 1GHz	128MB	256KB
resamo	140.77.15.20	Pentium II 400MHz	128MB	256KB
resamd	140.77.15.18	Pentium II 400MHz	128MB	256KB
resamc	140.77.15.19	Pentium II 400MHz	128MB	256KB
cluster11	140.77.15.111	Pentium III bi-pro 600MHz X2	256MB	256KB
Bioinfo5	140.77.15.31	AMD Athlon 700MHz	256MB	512KB
resama	140.77.15.99	PENTIUM II bi-pro 350MHz X2	128MB	512KB

TAB. 1 – Machines used for the tests

The purpose is the measurement of the overhead incurred by the active services, a set of experiments were performed in two different configurations. In the first configuration (see figure 13), the source `ike` multicasts packets to `stan` and `resamd` through 2 active routers (`resama` and `resamo`). In order to measure the data service overheads in this topology, one data packet is lost every 25 packets between the source and the first active router `resamo`. Figure 13 shows the steps of the recovery process where the source is the replier : DP and DR refer respectively to Data Packet and Data Repair.

In the second configuration (Figure 14), the source `ike` multicasts packets to a set of receivers through the active router `resamo`. The cost of the replier election is mainly the processing time for comparing the *subList* list against the receiver list in order to determine a replier for the lost data packet.

4.5 Cost of active services

The cost of the data packet service represents the processing time required to forward the packet to the destinations using the underlying IP multicast functionalities. Therefore, this cost is mainly includes the update of the track list when there are no sequence gaps. In case of a gap sequence indicating a packet loss, the data service includes additionally the spent time in setting a timer for ignoring subsequent similar NACKs. The cost of the NACK service contains the processing time to update or create the NS structure. The cost of the data repair service represents the processing time of scanning the *subList* and performing the subcast.

Preliminary results on our test-bed are illustrated in figures 15 and 16. Figure 15 shows the processing time in μs of a data packet, a repair packet and a NACK packet at the `resamo` active router. The x-axis shows the packet sequence number while the y-axis shows the processing time. The data packet size is initially set to 4KByte.

We can see that, in the absence of packet losses, the processing time of a data packet is about $20\mu s$. A gap in the x-axis represents a packet loss. For instance, in our test scenario packet 49 is first lost thus making the processing of packet 50 longer because of the loss detection and associated data structure updates (track list, NS structures...). Figure 15 shows that each packet loss incurs an additional cost of about 12ms to 17ms for the next packet that corresponds to the processing time for the data packet and the time for setting up a timer thread for NACK discarding. It is worth mentioning that the first overhead is only about $250\mu s$! Several optimizations are possible for the timer management (reuse of old threads, only 1 thread with a hash table for timers,...) and we expect much lower overheads in next implementations. The NACK packet and the repair packet that follow the lost packet are processed in approximately $135\mu s$ and $123\mu s$ respectively

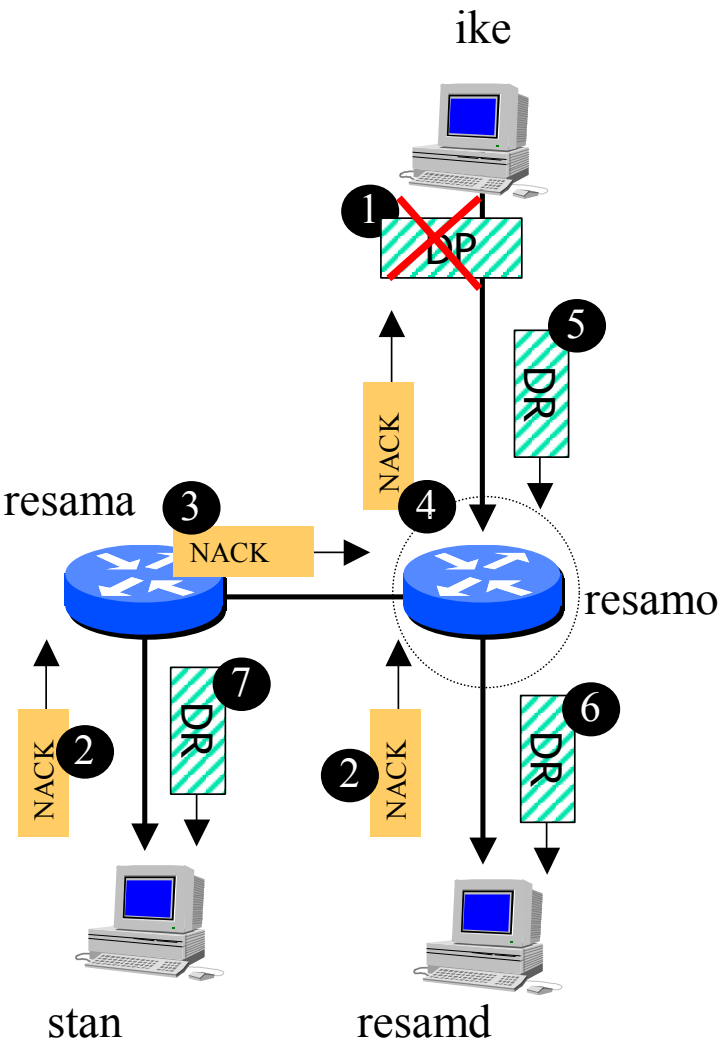


FIG. 13 – Topology 1 : DP, NACK and DR service.

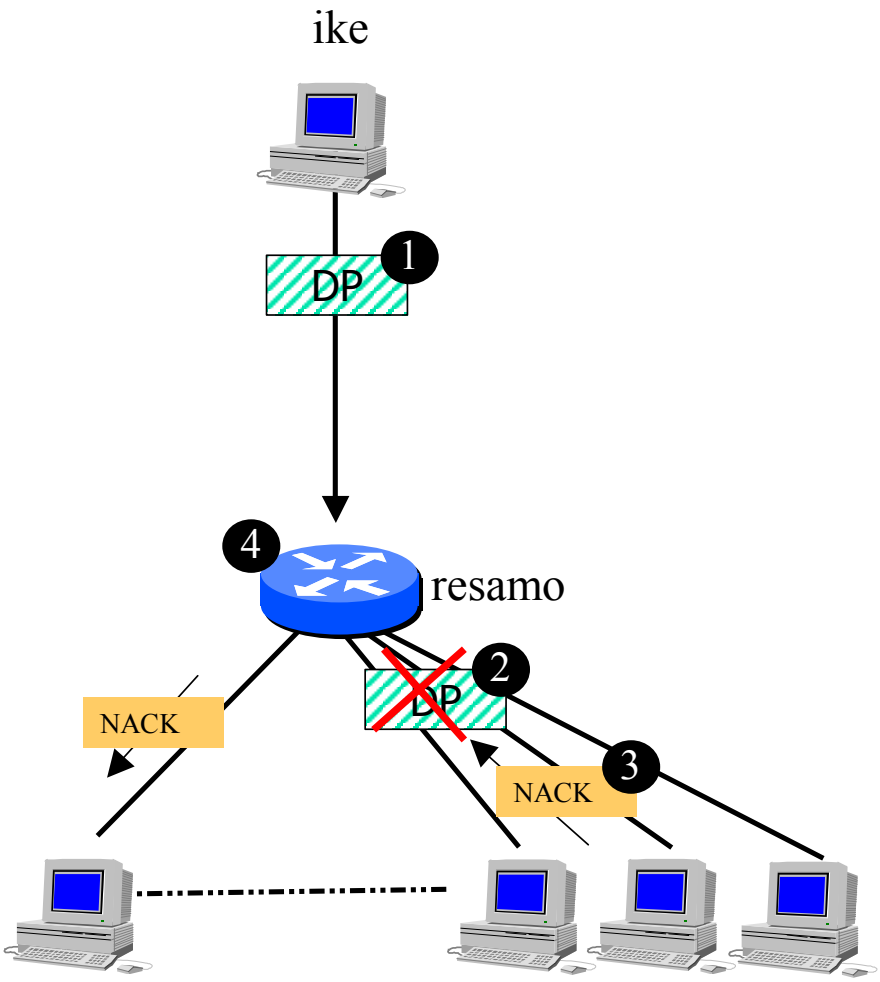


FIG. 14 – Topology 2 : replier election

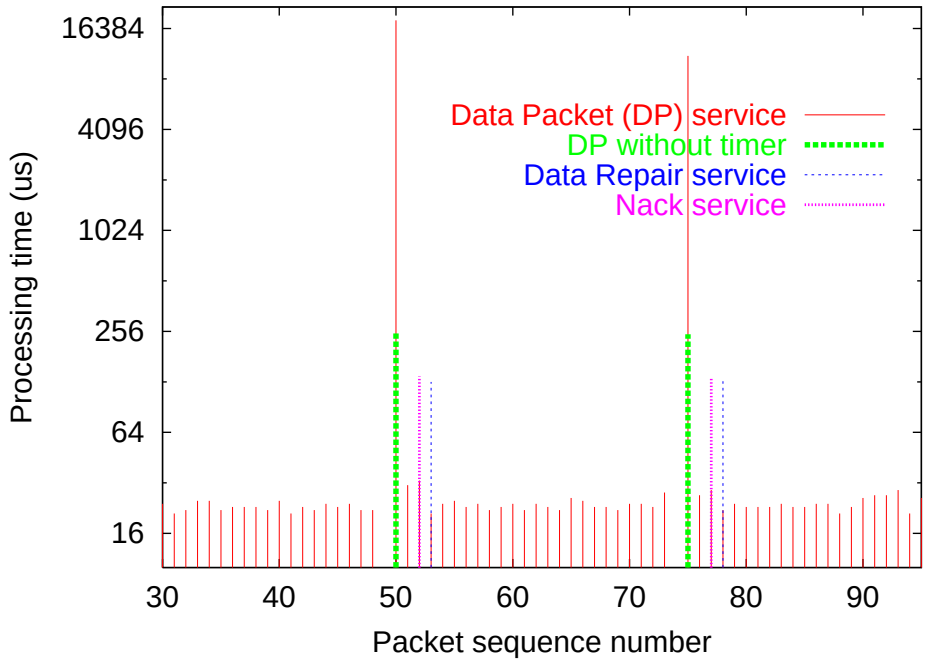


FIG. 15 – Cost of packet services

(for these packets, the x-axis indicates the current data packet sequence number at the moment they are processed in order to respect the chronological order). Varying the message size from 1KByte to 32KByte does not change the results.

Figure 16 shows the processing time to dynamically determine the replier. The number of receivers under the active router ranges from 5 to 25. The x-axis indicates how many receivers are involved in the replier election process. For instance, if we look at the 5-receivers curve, a value of 5 at the x-axis means that a replier is found after having scanned 4 receivers. The curves show that the number of total receivers has little impact of the election process : finding a replier after 4 tries amongst 25 receivers adds $30\mu s$ to the case where the replier is found amongst 5 receivers.

However, we do an on-the-fly election that consists in selecting a default replier (the first receiver in the receiver list which is obtained during the initialization session) when the first NACK arrives and updating the choice of the replier at each NACK reception. The advantage of this approach is to perform most of the election processing within the timer duration for gathering NACK information resulting in no cost from a protocol perspective.

5 Congestion Avoidance in DyRAM

Congestion appears to be the most common reason for packet loss in the Internet. Any reliable multicast protocol requires congestion control to be addressed. A congestion control algorithm has to avoid congestion collapses by quickly reacting to network conditions in a fair manner. Our proposed congestion avoidance algorithm for DyRAM reacts periodically on the reception of feedbacks from the receivers. These feedbacks contain mainly information about RTT variation used to adjust the rate of the transmission. To avoid the problem of rerouting paths, we adopt a hop-by-hop congestion control approach where communications between adjacent nodes including the routers, are performed. Active networks allow us to perform on a per link basis dialogue.

The available bandwidth along a multicast tree is probed in a per-link way based on the RTT variations experienced by every branch in the multicast tree. These RTT variations are then appropriately aggregated before they reach the source. However, instead of comparing the current RTT to the minimum experienced

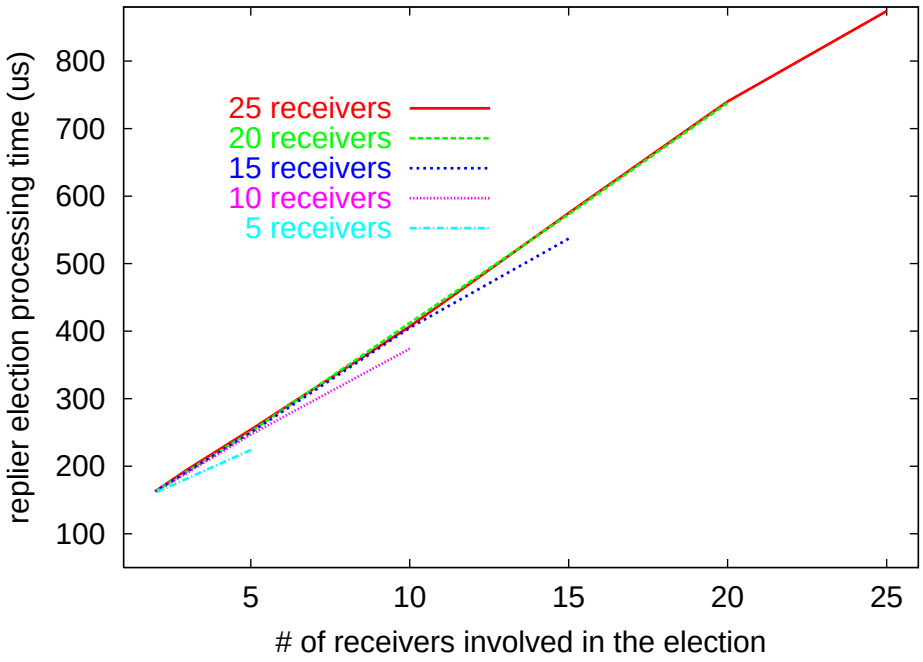


FIG. 16 – Cost of replier election.

one, in our congestion algorithm (i) we use the difference between the current and the previous RTT in order to converge to the available bandwidth in the multicast tree and (ii) we use the difference between the emission and the reception rate to estimate the number of transiting data packets.

A receiver sends periodically congestion report messages (CRs) to the source. In our congestion avoidance algorithm, every receiver sends a CR to the source every N packets received so the source can learn about the congestion situation of the multicast tree. In the case where no data packet is received, NACK packets are sent instead of the CRs. A NACK is used to indicate that a packet is missed by a receiver. The periodicity of sending the CRs varies because it depends on the current emission rate. To make the source knows the current period, a receiver puts in the *period* field of a CR, the required time to receive the last N data packets. Additionally, each receiver includes the last estimations for the its RTT to its parent and the source. These RTTs are used by the active routers or the source to update their parameters. This includes in addition to some timeout values, the minimum and the maximum RTT of the whole multicast tree or the subtree rooted at an active router.

To achieve scalability, a mechanism of the CRs aggregation is performed in order to unload both the network and the source. We rely on the physical multicast tree to hierarchically aggregating feedbacks at the intermediate nodes (routers). We have to deal with the feedback implosion problem while we have to allow the necessary information to be given to the source. An active router aggregates the received CRs from the downstream links in one CR to be forwarded upstream. The aggregated CR contains the sequence number of the minimum last ordered and the maximum last received data packets among those reported by the CRs received from downstream. For the RTT variations reported by the CRs from downstream in a multi-path connection, we use :

$$\Delta\tau = \Delta\tau_{up} + Max_i(\Delta\tau_i)$$

where $\Delta\tau_i$ are the reported RTT variations from the downstream links and $\Delta\tau_{up}$ is the RTT variation experienced by the upstream link. This latter is extracted from the HB_RESP message sent by the upstream node.

In our congestion avoidance algorithm, the regulation parameter is the rate. A minimum and a maximum rates r_{min} and r_{max} are set depending on the application. Initially the source starts to send data packets with a rate equal to the minimum rate r_{min}. Then the source tries to increase its rate if no congestion

indication is received without exceeding the maximum rate r_{max}. The source uses the information fed back in the CRs and NACKs to update its rate. The RTT variation field of every CR is used by the source to compute the queue size variation per packet Δq_p during the previous period. The goal is to maintain Δq_p as small as possible. During the slow start phase, the source would increase its rate with S/RTT_{max} (bits per second) every time it receives a CR that gives a queue size variation $\Delta q_p < \epsilon$, where ϵ is a positive number to be chosen in $[0, 1]$. Increasing the rate with S/RTT_{max} is equivalent to adding 1 to the congestion window for the largest end-to-end connection of the multicast tree. This corresponds to the slow start phase of TCP and makes our congestion avoidance algorithm fair with TCP from the beginning.

However, the network could be already congested from the beginning of the session. To avoid aggravating the situation, we need an other congestion indicator to be sure that there is no congestion in the network. This can be done by examining the subsequent values of the last ordered field of the CRs. This gives an idea about the difference between the emission and the reception rate. This difference is proportional to the number of data packets not acked yet Δq_a that can be given by :

$$\Delta q_a = S \times \max(0, N - (lo_{i+1} - lo_i)) \tag{2}$$

The source, on the reception of a CR, extracts the current period T and the RTT variation $\Delta\tau$. Afterwards, in addition to Δq_p, Δq_a is also computed using (2). The aim is to maintain Δq_p and Δq_a as small as possible. We try to maintain Δq_a to be in $[\alpha, \beta]$, where α and β are similar to the two parameters of TCP-Vegas.

In the case of a severe congestion or when the network is already congested from the beginning (in addition to the proposed mechanisms), the rate is dropped to its minimum r_{min}. A severe congestion could be detected if no CR, a HB or a NACK is received during a relatively long period which is set to 2 times the current CR period.

6 Further Details

The constraints put on a reliable multicast protocol are numerous and difficult, and a "good" protocol must provide efficient solutions to several problems such as feedback implosion, recovery latency, scope of the retransmissions and duplicate repairs. In this section, we discuss some other aspects and details of our approach.

6.1 NACK reverse-path requirements

One key assumption to take advantage of the DyRAM active services is that a NACK follows the reverse-path of the data packet. This assumption is essential for taking advantage of active services. This behavior can be implemented by using a dedicated routing service for NACK packets or to make a receiver know the the identity of the active router to which it is attached [8].

6.2 Memory usage

One of our design goals was to unload the routers from caching data packets to allow more scalability. In our approach there is no data packets caching, instead some soft-state structures are maintained in the routers in order to perform the different active services on which DyRAM relies. We argue that a few amount of memory is sufficient to keep these data structures so our approach will perform well.

In fact, a router maintains only one TL structure per multicast session and one LS structure per downstream links. For the NS structure, we create an NS structure per loss and is removed on the receipt of the corresponding repair. We estimate that active routers need only a small amount of memory to maintain the NS structures. The life time of a NS depends on how soon the router expects to receive the corresponding repair. A repair can come either from the original source or from a replier which is more likely to be the closest source of data packets. Consequently maintaining a NS structure until the reception of the corresponding repair is not expensive. We can easily demonstrate analytically that the life time of a NS is sufficiently short so that a few amount of memory is sufficient to keep a NS without replacing it by another until the reception of the corresponding repair.

To prevent the fact that a NS structure has a non finite life time due to the loss of the corresponding repair, this NS is removed. In fact, on the expiration of the corresponding NST, the NS structure is removed after a given amount of time if no NACK for the requested data packet has been received. Nevertheless, if on the reception of the corresponding repair, the NS structure is not found then the repair is simply multicast on all the links downstream if it was received from the upstream link ; otherwise, it is sent on all the links downstream except the link originator of the NACK.

6.3 Flow Control and Memory Management

For the purpose of flow control, every receiver includes in the $Maxseq$ field of a CR or a NACK, the maximum data packet sequence number that corresponds to its available buffering means. The MaxSeq field in a CR or NACK packet is used by the source to regulate the transmission flow. An active router includes the minimum of the received $MaxSeq$ in the CR to be sent upstream to the source. The source then sends data packets with the current rate until the minimum $MaxSeq$ of all the receivers.

The lo field is also used to release buffers at both the sender and the receivers. With the proposed mechanism of aggregating the CRs, the source ends-up by receiving the minimum lo of the whole group. This is an acknowledgement that all data packets up to lo have been correctly received by all the receivers. The source could then release buffers of data packets up to loth one. The source includes the current lo of the whole group in the $MaxSeq$ field of the data packets it sends. A receiver on the reception of a data packet with a $MaxSeq$ field of seq would update its flow control window accordingly. Moreover and since a receiver would keep data packets in order to be able to act as an eventual replier, this mechanism is helpful to releasing memory. in fact, a receiver could remove from its memory all the data packets with a sequence number less or equal to the specified $MaxSeq$.

7 Conclusion and Future Work

In this report we have reported our experience on the design, evaluation and the implementation of DyRAM, an active reliable multicast protocol. DyRAM performs the local recovery from the receivers with the assistance of the routers without any in-network caching facilities. A link is elected as a replier on a per-packet basis to perform the retransmission of a data packet. An estimation of the link capacities is performed in order to have a fairer replier election with a nice load balancing feature among the links downstream a router.

Knowing that losses are mainly due to congestion, a congestion control algorithm is already proposed to strengthen our reliability mechanism. In this report we focused on the reliability mechanism with a brief overview of the congestion control algorithm. However an extensive evaluation study using both simulations and analysis has been performed on DyRAM. In addition a first prototype of DyRAM has been presented with preliminary measurements of the raw processing time required by the different services.

Regarding the problem of scalability, our approach does not use caching facilities within routers but only a few amount of RAM memory for maintaining some data structures per multicast session. As the amount of memory needed for this purpose is small (few KBytes per session) we believe our approach to be more scalable than an approach like ARM. Based on small data structures we showed via the implementation of DyRAM that its associated active services introduce only low overheads in routers and that implementations without much performance degradation on the routing and forwarding functions are possible. In fact in this report we addressed this performance issue by measuring the per-router processing cost of elementary services proposed in the DyRAM protocol. The results are very encouraging as most processing costs range from tens of μs to hundreds of μs on a Pentium II 400MHz PC-based router.

Given these results, we believe it is very possible to build active routers from regular PCs (using the most up-to-date processor and clock rate) and still get performances at high bit rates. This possibility would certainly help to disseminate the use of active networking technologies for a large range of applications as the deployment would be easier and faster than a dedicated router solution.

The topologies we used have a very limited number of routers and receivers and therefore the study is not intended to catch any end-to-end performances, as mentioned previously in the paper. As a future work, we

plan to perform experimentations in a wide area network to validate the DyRAM approach.The congestion control algorithm will also be integrated in the DyRAM framework in the near future.

Références

[1] R. Bagrodia et al. Parsec : A parallel simulation environment for complex systems. *Computer Magazine*, 1998.

[2] M. Calderón, M. Sedano, A. Azcorra, and C. Alonso. Active networks support for multicast applications. *IEEE Networks*, May/June 1998.

[3] S. E. Deering and D. R. Cheriton. Multicast routing in datagrams internetworks and extended lans. In *ACM Transactions on Computer Systems*, May 1990.

[4] S. Zabele et al. Improving distributed simulation performance using active networks. In *World Multi Conference*, 2000.

[5] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), 1997.

[6] J.P. Gelas and L. Lefévre. Tamanoir : A high performance active network framework. Workshop on Active Middleware Services 2000, 9th IEEE International HPDC, Pittsburgh.

[7] Hugh W. Holbrook, Sandeep K. Singhal, and David R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *SIGCOMM*, Oct. 1995.

[8] S. Kasera and S. Bhattacharya. Scalabe fair reliable multicast using active services. *IEEE Network Magazine's Special Issue on Multicast*, 2000.

[9] L. Lehman, S. Garland, and D. Tennehouse. Active reliable multicast. In *Proc. of the IEEE INFOCOM, San Francisco, CA*, March 1998.

[10] M. Maimour and C. Pham. A throughput analysis of reliable multicast protocols in an active networking environment. In *Proc. of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001)*, July 2001.

[11] M. Maimour and C. Pham. An active-based multicast congestion avoidance algorithm. Technical Report TR01-2002, RESO, http ://www.ens-lyon.fr/~mmaimour/Paper/TR/TR01-2002.ps.gz, 2002.

[12] M. Maimour and C. Pham. An analysis of a router-based loss detection service for active reliable multicast protocols. In *Proc. of the International Conference on Networks (ICON 2002)*, August 2002.

[13] Christos Papadopoulos, Guru M. Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *Proc. of the IEEE INFOCOM*, March 1998.

[14] S. Paul and K. Sabnani. Reliable multicast transport protocol (RMTP). *IEEE JSAC, Spec. Issue on Network Support for Multipoint Communications*, 15(3), April 1997.

[15] T. Speakman et al. Pgm reliable transport protocol specification. internet draft, 1998.

[16] R. State, O. Festor, and E. Nataf. A programmable network based approach for managing dynamic virtual private networks. In *Proceedings of PDPTA'02, Las Vegas, June 26-29*, 2000.

[17] D. L. Tennehouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Winden. A survey of active network research. *IEEE Communication Magazine*, pages 80–86, January 1997.

[18] M. Yajnick, J. Kurose, and D. Tosley. Packet loss correlation in the mbone multicast network. In *Proc. of Global Internet conference*, November 1996.

[19] M. Yamamoto, J. Kurose, D. Towsley, and H. Ikeda. A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols. In *Proc. of IEEE INFOCOM'97, Los Alamitos*, April 1997.

[20] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, 1995.

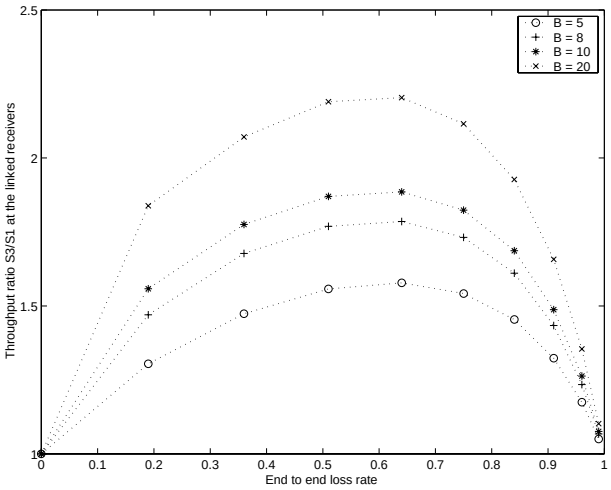


FIG. 17 – Throughput benefit of the subcast for the receivers.

A Some other Analytical Results

These analytical results are taken from [10, 12], two previous evaluation works performed on different active reliable multicast protocols. In [12] as has been already spoken about in section 3.3, the evaluation is performed via a delay analysis of two protocols A and D.

In [10], we performed a throughput analysis performed on generic versions of different active reliable multicast protocols. Among them we mention here S_1 and S_2 which are generic versions of ARM [9] and AER [8] respectively. These two generic protocols were evaluated using the network model of figure 3. It is worth mentioning that there is no loss detection in the active routers. The analysis (see [10]) uses the processing requirements of the protocols to derive the achievable throughput at the various nodes in the multicast tree.

A.1 Benefit of the subcast

The subcast facility has the advantage of unloading the receivers and the active routers. To see the benefit of performing the subcast from the active routers, Figure 17 plots the throughput gain a receiver would experience thanks to the subcast. It is clear that the subcast allows a higher throughput. We observe that the gain obtained with the subcast depends on the local group size and the loss rate. These two parameters gives an idea on the number of the receivers that have experienced a loss. Therefore, it is more beneficial to perform the subcast when the local group size is bigger. In the case of very low and high loss rates, the subcast does not allow a remarkable gain. In fact for very low loss probabilities, i.e. when p is close to 0, the number of recoveries performed by an active router are very few. Therefore a receiver that does not benefit from the subcast facility is not overwhelmed by receiving several copies of a data packet. For very high loss rates, i.e. when p is close to 1, the number of the receivers that experience a loss within a local group increases. Thus there are few receivers that are not concerned by the repair and the subcast does not allow a noticeable gain.

A.2 Active routers density

The performances of DyRAM when all the present routers are active have been extensively studied. Here figure 18 shows the impact of the active routers density on the performances of a reliable multicast protocol in terms of the overall throughput achieved by all the nodes [10]. The figure plots the overall throughput gain as the number of active routers is increased compared to the no active routers case. Several multiplicative factors to the active routers' processing power are applied. We can see that with the same processing time

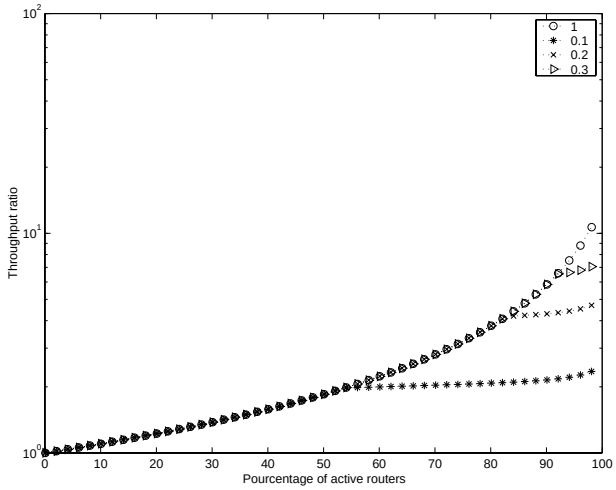


FIG. 18 – Gain in terms of the overall throughput achieved by an active reliable multicast protocol when varying the active routers density. $p = 0.05$, $B = 10$ and $N = 1000$.

at the active routers and the receivers, the overall throughput can be an order of magnitude higher if all the routers are active. Most interestingly, if the active router's processing power is divided by 10, we can still double the overall throughput provided that 55 % of routers are active.

A.3 Active routers processing power

A.3.1 Required Processing power

The throughput analysis [10] has shown that the overall throughput is limited by the active routers. Figure 19 is an attempt to answer the question : "How much the active routers processing power must be increased so that the network is never the bottleneck?". We introduce the ratio between the throughput achieved by a receiver (L) and an active router (A) (the receivers are the end nodes that introduce the lowest throughput). This ratio gives the multiplicative factor put on an active router's processing power in order to achieve the same level of performances as a receiver.

Figure 19 shows the L/A ratio for S1 and S2. For instance, for one local group (i.e. 10 receivers) active routers in S1 need to be almost 4 times faster to achieve the same level of performance as the receivers. More interestingly, we can observe that the required processing power so the routers are not the bottleneck decreases with the receivers number.

A.3.2 Maximum Supported loss rate

Results here are derived from the delay analysis of [12]. Figure 20 plots the maximum supported p_l by the D protocol as a function of the processing power. Figure 20(a) shows that the maximum supported loss rate increases as the processing power increases from 0.1 (corresponding to a reduction by a factor of 10) to 1 (the router has the same processing power as the end hosts). Figure 20(b) shows that one does not need to increase the processing power infinitely. In fact, for 5 receivers the maximum supported loss rate does not increase if the processing power is increased beyond 9 times. Even if the number of receivers is multiplied by 20, increasing the processing power beyond 16 will not increase the supported loss rate. This is due to the fact that the routers are not the bottleneck.

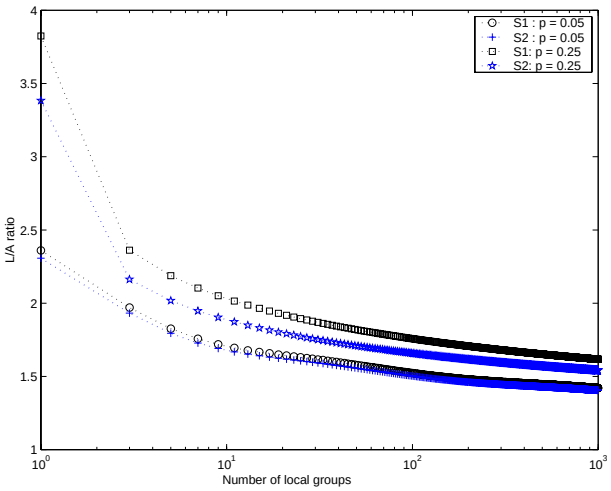
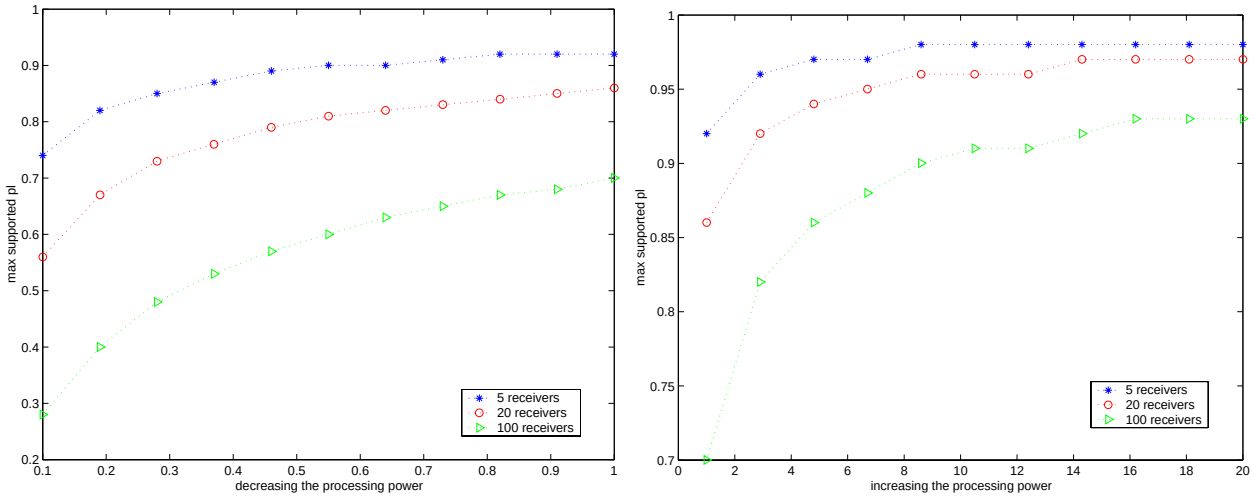


FIG. 19 – Receiver to active router throughput ratio. $B = 10$ and 50% of active routers.



(a) (b)

FIG. 20 – Maximum p_l supported by D as a function of the routers processing power.