The Cost of Active Services in Active Reliable Multicast*

M. Maimour, J. Mazuy and C. Pham
RESAM UCBL-INRIA, University Lyon 1
ENS, 46 allée d'Italie
69364 Lyon Cedex 07 - France
e-mail: Congduc.Pham@ens-lyon.fr

Abstract

Associating customized processing to incoming packets, active networking provides a general framework for new protocol deployment and new service implementations. Recently, the use of these active network concepts has been proposed in many research areas including multicast protocols where efficient mechanisms to the reliability problems can be proposed. However, these active services introduce additional processing costs that must be carefully evaluated. This paper presents the preliminary results in measuring, in the context of the DyRAM protocol, the raw processing time required for enabling low-latency multicast communications on the Internet.

1 Introduction

Violating the well-established end-to-end architecture, routers in the active networking approach [13] can execute application-dependent computations on incoming packets. In this perspective, a number of execution environments, most of them based on the Java language, have been proposed to allow the dynamic execution of "user" code [14, 4].

Recently, the use of these active network concepts has been proposed in many research areas including multicast protocols [15, 6, 1], distributed interactive simulations [18] and network management [11]. 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 dynamicity.

In this paper, we address the performance issues in the context of the DyRAM framework [7] developed for providing low latencies on computational grids. The framework consists in a reliable protocol with specialized active services located at the edges of the core network. These services are typically hosted by routers and the main concern now is to be able to quantify the overhead incurred by these active services. The paper is organized as follows. Section 2 presents the motivations behind active reliable multicast protocols and Section 3 presents the DyRAM framework. Section 4 addresses the performance issues and Section 5 concludes.

*This work is supported in part by the french ACI Grid program and by a grant from ANVAR-EZUS Lyon.

2 Review of reliable multicast technologies

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 to the final destination as possible. In contrast to best-effort multicast, that typically tolerates some data losses and is more suited for real-time audio or video for instance, reliable multicast requires that all packets are safely delivered to the destinations.

2.1 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 [2]) has no mechanisms for reliability. Therefore reliability has to be added at a higher layer. In the past, there have been a number of propositions for reliable multicast protocols relying on complex exchanges of feedback messages (ACK or NACK) [16, 3, 10, 17]. They mainly suffer from the implosion of control messages and usually do not scale well. 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 [3]), a designated receiver (RMTP [10], TMTP [17], LMS [9], PGM [12]) or a logging server (LBRM [5]) in a hierarchical structure.

2.2 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 LMS [9] and PGM [12] for instance. 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 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, fully active solutions can provide efficient mechanisms to the reliability problem, and especially to enabling fast local recoveries. Most active services proposed so far in multicast protocols contribute mainly on feedback implosion problems, retransmission scoping and cache of data. For instance, ARM [15], AER [6] and RMANP [1] are protocols that use cache of data packets to permit local recoveries and advanced NACK suppression strategies.

3 The DyRAM Framework

3.1 Overview

DyRAM [7] is a reliable multicast protocol suite with a recovery strategy based on a tree structure constructed on a per-packet basis with the assistance of routers. 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 (In order to allow for flow control and memory management, ACKs are piggybacked by the NACKs. In the absence of such NACKs, periodic ACKs are sent). As opposed to LMS and PGM, DyRAM uses intensively the concept of active networking with active services within routers for implementing several advanced mechanisms:

1. the early detection of packet losses and the emission of the corresponding NACKs.
2. the NACK suppression of duplicated NACKs (from end-hosts) in order to limit the NACK implosion problem.
3. the subcast of the repair packets only to the relevant set of receivers that have experienced a loss. This service limits the scope of the repairs to the affected subtree.
4. the dynamic replier election which consists in choosing a link/host as a replier one to perform local recoveries. Dynamic election provides robustness to host and link failures.

DyRAM has been designed with the following motivations in mind: (i) to minimize active routers load to make them supporting more sessions (mainly in unloading them from the cache of data) and (ii) to reduce the recovery latency. Avoiding cache in routers is performed by a dynamic replier election mechanism. 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 [9] which is only updated when the group membership changes, thus justifying the "dynamic replier" property of the protocol. However, one of the main concern is the impact of this dynamic approach on performances.

3.2 Router's soft states and algorithms

Within active routers, the eurly loss detection, the NACK suppression, the subcast and the replier election services can be implemented simply by maintaining information about the data packets and NACKs received. This set of information is uniquely identified by the session source and the multicast address.

3.2.1 Topology information

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. The main information gathered per router are (i) the number of receivers directly connected and (ii) their IP addresses. These information are used by the subcast service, the replier election service and the early loss detection service.

3.2.2 NACK and data services

For each received NACK, the router creates or simply updates an existing NACK state (NS) structure which has a limited life time. This structure maintains for avery nacked

packet, a subcast list ($subList$) that contains the IP addresses of the receivers that experienced a loss.

On receipt of the first NACK packet for a data packet, a router would create a corresponding NS structure, initialize a timer noted DTD (Delay To Decide) that will trigger the election of a replier to whom this first NACK will be sent. All subsequent NACK packets received during the timeout interval are used to properly update the subcast list and are dropped afterward.

DyRAM also enables router 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 structure (TL) 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 order.
- $lastReceived$ is the sequence number of the last received data packet.
- $lostList$ contains the list of data packets not received by the router.

An active router would detect a loss when a gap occurs in the data packet sequence. The data service, in case no error occurred, then simply consists in updating the track list. On a loss detection, the router would immediately generate 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).

3.2.3 Dynamic replier election

Local recoveries, when possible, are performed by elected receivers (repliers) that have correctly received the lost packet. The replier election process executed by an active router uses the $subList$ in the NS structure to determine which receiver can potentially be elected. The algorithm basically works by comparing in the list of receivers which one does not appear in the $subList$ (there is no message exchanges during the election process). 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. Once the election is completed, the router will forward the NACK downstream toward the elected replier. This latter would then unicast the repair packet to its upstream DyRAM router which would in turn subcast the repair packet on all the links in the subcast list.

3.3 Performance of DyRAM

The performances of DyRAM have been extensively studied in [7, 8] 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. This is the purpose of the next section. 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 no timer optimizations nor end-to-end recovery latencies are investigated.

4 The cost of active services in DyRAM

A test-bed consisting of a core protocol library for sender and receiver applications, along with specialized active services has been developed in Java in the context of the Tamanoir [4] execution environment.

4.1 Algorithms, data structures and packet format

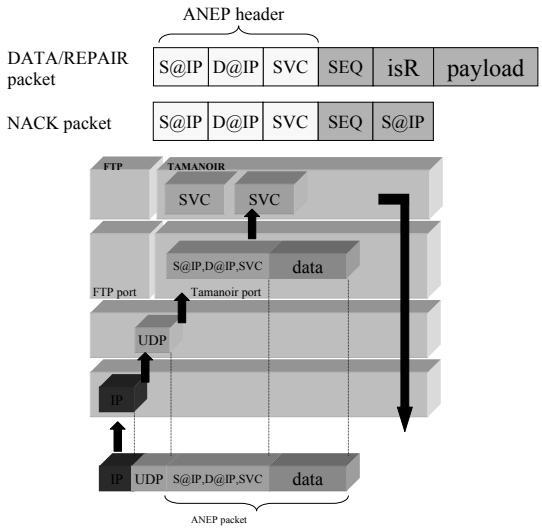

Figure 1. Packet format and Tamanoir architecture.

The track list (TL) 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 paper as the hash key. Each entry refers to a double-linked list of IP addresses ($subList$). 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.

Figure 1 depicts the data path of incoming messages. DyRAM packets used the ANEP [14] format and UDP is used for the transport as reliability is directly handled by DyRAM. We can see in the figure that incoming DyRAM packets are given to the Tamanoir execution environment which selects the appropriate active service based on the `SVC` (service identifier) field.

4.2 Test-bed and scenario

The test-bed consists in a set of receivers and 2 PC-based routers (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.

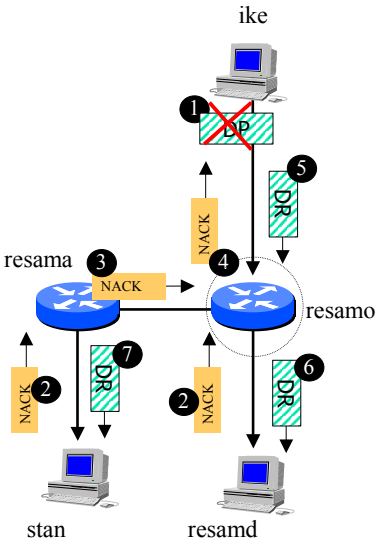

Figure 2. Topology 1: DP, NACK service and DR service.

In the first configuration, the source `ike` multicasts packets to `stan` and `resamd` through 2 active routers. 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 2 shows the steps of the recovery process where the source is the replier: DP and DR refer respectively to Data Packet and Data Repair. 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 an update of the track list when there are no sequence gaps. In case of a gap sequence indicating a packet loss, the data service consists additionally in setting a timer for ignoring subsequent similar NACKs. The cost of the NACK service represents 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.

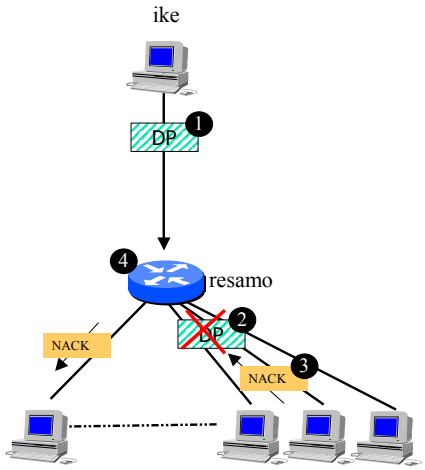

Figure 3. Topology 2: replier election

In topology 2 (Fig. 3), the source `ike` multicasts packets to a set of receivers through the active router `resamo`. The cost of the replier election represents the processing time for comparing the $subList$ list against the receiver list in order to determine a replier for the lost data packet.

4.3 Cost of active services

Preliminary results on our test-bed are illustrated in figures 4 and 5. Figure 4 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 4 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

250μs! 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 135μs and 123μs respectively (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.

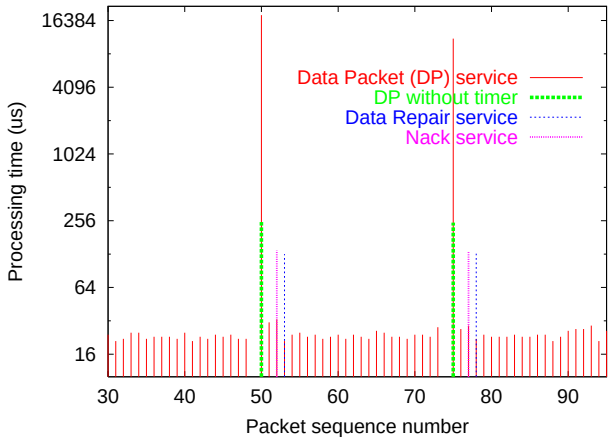

Figure 4. Cost of packet services

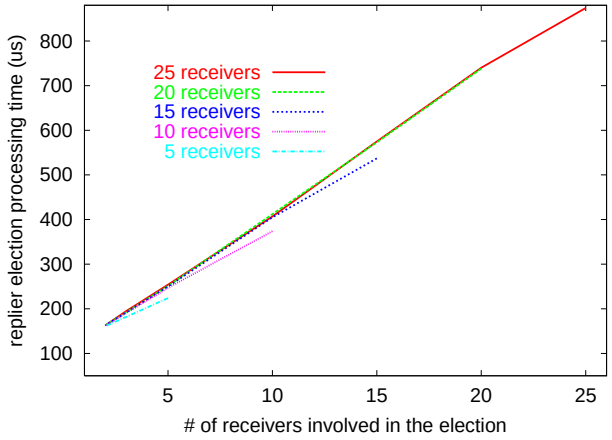

Figure 5. Cost of replier election.

Figure 5 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 30μs to the case where the replier is found amongst 5 receivers.

It must be noted that the replier election is performed on-the-fly during the timer duration for gathering NACK information. In practice, this timer should be set to at least the propagation time to the farthest receiver in the local group. Therefore, it is possible that the election cost is masked by the timer duration, resulting in no cost from a protocol perspective.

5 Conclusions

Active multicast protocols can propose efficient mechanisms to solve in a scalable manner the reliability problem. However, their main drawback is the extra processing cost introduced within network elements that can possibly be overwhelming. This paper 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. The topologies we used have a very limited number of sources, routers and receivers and therefore the study is not intended to catch any end-to-end performances, as mentioned previously in the paper.

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. Additionally, the dynamic replier election service that can potentially be very costly appears not to be the bottleneck, especially if we consider the case where there is a hierarchical tree of active routers, each one managing from 1 to 50 receivers.

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.

References

[1] M. Calderón, M. Sedano, A. Azcorra, C. Alonso. Active Networks Support for Multicast Applications. *IEEE Networks, May/June 1998.*

[2] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs newblock In *ACM SIGCOMM'88* and *ACM Trans. on Comp. Sys., Vol. 8, No. 2.*

[3] S. Floyd, V. Jacobson, and Liu C. G. A reliable multicast framework for light weight session and application level framing. In *ACM SIGCOMM'95*, pp342–356.

[4] J.P. Gelas and L. Lefèvre. TAMANOIR : A High Performance Active Network Framework. *Workshop on Active Middleware Services 2000, 9th IEEE International HPDC, Pittsburgh.*

[5] H. W. Holbrook, S. K. Singhal, D. R. Cheriton. Log-Based receiver-Reliable Multicast for Distributed Interactive Simulation. *Proceeding of ACM SigComm'95*.

[6] S. K. Kasera et al. Scalable fair reliable multicast using active services. *IEEE Networks, Special Issue on Multicast*, 2000.

[7] M. Maimour and C. Pham. Dynamic Replier Active Reliable Multicast (DyRAM) *Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC 2002)*.

[8] M. Maimour and C. Pham. A Loss Detection Service for Active Reliable Multicast Protocols, *Proceedings of the International Network Conference (INC 2002),July 16-18 2002, Plymouth, UK*.

[9] Christos Papadopoulos, Guru M. Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *IEEE INFOCOM'98*, pp1188–1996.

[10] S. Paul and K. K. Sabnani. Reliable multicast transport protocol (rmtp). *IEEE JSAC, Special Issue on Network Support for Multipoint Comm.*, 15(3):407–421.

[11] R. State, O. Festor, E. Nataf. A Programmable Network Based Approach for Managing Dynamic Virtual Private Networks In *Proceedings of PDPTA 2000*, Las Vegas, June 26-29.

[12] T. Speakman et al. Pgm reliable transport protocol specification. Internet draft.

[13] D. L. Tennehouse et al. A survey of active network research. *IEEE Comm. Mag.*, pp80–86, January 1997.

[14] D.J. Wetherall, J.V. Guttag and D.L. Tennehouse. ANTS: a Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98, San Francisco, April 1998.*

[15] L. Wei, H. Lehman, S. J. Garland, and D. L. Tennenhouse. Active reliable multicast. In *IEEE INFOCOM'98*.

[16] XTP Forum. *Xpress Transport Protocol Specification*, March 1995.

[17] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia'95*.

[18] S. Zabele et al. Improving Distributed Simulation Performance Using Active Networks. In *World Multi Conference, 2000.*