N° d'ordre 210-2003 Année 2003

University Claude Bernard - Lyon I

DESIGN, ANALYSIS AND VALIDATION OF ROUTER-ASSISTED RELIABLE MULTICAST PROTOCOLS IN WIDE AREA NETWORKS

By
Moufida MAIMOUR

Under Supervision of
Dr. Cong-Duc PHAM and Dr. Pascale VICAT-BLANC/PRIMET

A Dissertation Submitted in Satisfaction
of the Requirements for the Degree of
Doctor of Philosophy
in Computer Science

in the team
RESO (Réseaux haut-débit : Protocoles et Services)
of the
LIP (Laboratoire Informatique du Parallélisme)-ÉCOLE NORMALE SUPÉRIEURE DE LYON

Defence date: November 25th 2003

Committee in Charge: Serge Fdida, President
Guy Leduc, Rapporteur
Jean-Jacques Pansiot, Rapporteur
Alain Milles, Member
Vincent Roca, Member
Cong-Duc Pham, Member
Pascale Vicat-Blanc/Primet, Member

N° d'ordre 210-2003 Année 2003

Université Claude Bernard - Lyon I

CONCEPTION, ANALYSE et VALIDATION DE PROTOCOLES DE MULTICAST FIABLE AVEC L'ASSISTANCE DES ROUTEURS DANS LES RÉSEAUX ETENDUS

Par
Moufida MAIMOUR

Sous la supervision de
Dr. Cong-Duc PHAM et Dr. Pascale VICAT-BLANC/PRIMET

Pour obtenir le titre de
Docteur de l'université Lyon1
en Informatique

Equipe
RESO (Réseaux haut-débit : Protocoles et Services)
du
LIP (Laboratoire Informatique du Parallélisme)-ÉCOLE NORMALE SUPÉRIEURE DE LYON

Date de soutenance : 25 novembre 2003

Jury: Serge Fdida, President
Guy Leduc, Rapporteur
Jean-Jacques Pansiot, Rapporteur
Alain Milles, Membre
Vincent Roca, Membre
Cong-Duc Pham, Membre
Pascale Vicat-Blanc/Primet, Membre

Abstract

In the Internet, multi-point communications (multicast) allow for the exchange of data between different members that belong to the same group. At the network level, a new communication model called *IP Multicast* is introduced, and a given number of membership management and routing protocols are proposed. At the transport level, another class of problems is to be addressed such as reliability, congestion control and the accommodation of the co-existence of heterogeneous receivers in the same multicast session. The purpose of this thesis is to investigate the potential contribution of router-assisted approaches in solving some of the multicast transport level problems. We adopt the active networking technology where routers are able to perform customized services on packets they receive, to validate our proposals.

As a first step, some analytical evaluations are performed to evaluate different router-based services that contribute to solve the problem of reliable multicast. Afterwards, a reliable multicast protocol called DyRAM is proposed and then extensively evaluated using simulations. In DyRAM, a dynamic congruent tree is implicitly built with the assistance of the routers via the election of repliers on a per-packet basis. This allows DyRAM to satisfy both congruence and adaptability requirements of a tree-based reliable multicast protocol. A first implementation of DyRAM is available and some measurements on a local test-bed are performed. Other experiments have also been conducted in the context of a grid computing environment.

Afterwards, a multicast congestion avoidance protocol called AMCA, is proposed to limit data losses due to buffers overflow at the routers. In AMCA, a hop-by-hop measurements are performed with the assistance of the routers in order to learn about network congestion status. The routers are also used to aggregate feedback messages sent by the receivers such that the source ends up by adapting its rate to the most congested member in the group. Simulations showed that AMCA converges, makes use of the available bandwidth and reacts rapidly to dynamic changes in the multicast tree while being fair with TCP.

Finally, a receiver-based replication scheme is proposed to implement a fine-grained multi-rate congestion control in order to accommodate heterogeneity and thus improving the overall receivers' satisfaction. Mainly, a partitioning algorithm is provided to split the set of the receivers into subgroups of similar capacities. Although simple, it achieves or at least approximates the optimal partitioning without requiring the prior knowledge of the receivers' capacities. We show that receiver-based replication compared to single-rate multicast or source-based replication, is fairer and more scalable. Moreover, an extension of AMCA is performed and mainly showed the rapid convergence of the partitioning algorithm when implementing the receiver-based replication scheme.

Résumé

Dans l'Internet, les communications multipoint (multicast) permettent l'échange de données entre plusieurs membres appartenant au même groupe. Au niveau réseau, un nouveau modèle de communication appelé *Multicast IP* a été proposé ainsi qu'un certain nombre de protocoles de routage et de gestion de groupes. Au niveau transport, une autre classe de problèmes se pose. Cette classe comprend en particulier la fiabilité, le contrôle de congestion ainsi que la prise en charge de l'hétérogénéité des récepteurs qui appartiennent au même groupe. L'objectif de cette thèse est d'étudier l'apport de l'implication des routeurs dans la solution de certains problèmes du niveau transport. Pour valider nos propositions nous adoptons la technologie des réseaux actifs où les routeurs sont capables d'exécuter des services personnalisés en fonction des paquets reçus.

En premier lieu, quelques analyses sont faites pour évaluer différents services au niveau des routeurs pour le multicast fiable. Ensuite, un protocole de multicast fiable appelé DyRAM est proposé et évalué par simulations. Dans DyRAM, un arbre dynamique proche de l'arbre de multicast, est construit implicitement avec l'assistance des routeurs via l'élection d'un retransmetteur par paquet perdu. Une première implémentation de DyRAM est disponible et quelques mesures sur une plate-forme locale ont été réalisées. D'autres expérimentations ont été aussi menées dans un environnement de grille de calcul.

Ensuite, un protocole d'évitement de congestion en multicast appelé AMCA, est proposé pour limiter les pertes de paquets dues à des débordements au niveau des routeurs. Dans AMCA, des mesures sont faites avec l'assistance des routeurs pour connaître l'état de congestion du réseau. Les routeurs sont aussi utilisés pour agréger les messages de contrôle émis par les récepteurs de telle sorte que la source finit par adapter son débit au récepteur le plus congestionné du groupe. Des simulations ont montré que AMCA converge, utilise la bande passante disponible et réagit rapidement aux changements dynamiques de l'arbre de multicast tout en étant équitable avec TCP.

Enfin, un schéma de réplication par les récepteurs est proposé pour implémenter un protocole multi-débits, finement contrôlé, afin de prendre en considération l'hétérogénéité. En particulier, un algorithme de partitionnement est fourni pour distribuer les récepteurs sur des groupes de capacités similaires. Malgré sa simplicité, cet algorithme atteint ou au moins approche la solution optimale sans avoir besoin d'une connaissance préalable de la capacité des récepteurs. Nous montrons que cette réplication par les récepteurs comparée à un multicast à un débit ou une réplication par la source, est plus équitable et permet plus de passage à l'échelle. Une extension de AMCA a été réalisée et a montré la convergence rapide de l'algorithme de partitionnement lors de l'implémentation de la réplication par les récepteurs.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Within less than a decade, Internet Protocol (IP) technology has gained popularity and changed many faces of our daily life. Nowadays, the Internet becomes essential for information exchanges in many domains. However, today's supported communication is largely limited to one-to-one data exchanges while a wide variety of emerging Internet applications covering a wide spectrum of domains, involve multiple parties and require a multi-point communication mechanism. These applications include but are not limited to software distribution, replicated database updates, distributed interactive simulations, collaborative computing, multimedia conferencing, distance-learning and distributed games.

One simple solution to implement a multi-point communication mechanism is to open multiple unicast connections. However this solution has significant scaling problems to a large group of receivers. Sending packets on a broadcast address is simpler than the unicast approach. However, broadcasting in a wide area network consumes an excessive quantity of network resources, mainly when only a small group is interested in the reception. An intermediate and better solution known as *multicast*, consists in sending data packets only to a subset of the network nodes. Multicast is a new network service that handles one-to-many and many-to-many communications in a wide area network while saving network bandwidth and senders' processing capacity by avoiding opening multiple connections. Redundant traffic is avoided through packets duplication as close as possible to the final destinations. This model has a direct impact on scalability to an extremely large number of participants in a group communication since it reduces the amount of data injected into the network by a multicast session.

In 1988, Deering proposed an open multicast model [32] where receivers interested in a multicast session have only to subscribe to a group address. Any receiver can join and leave the multicast group as it pleases. The source is not aware of neither the identity nor the number of the subscribed receivers. Moreover, a source can send on a multicast address without having to be subscribed to. In this multicast model known as *IP multicast* (RFC 1112), it is the responsibility of routers to ensure the packets delivery to their destinations through special routing protocols that maintain membership information and build multicast distribution trees.

While the benefit and advantages of such a multicast model over multiple unicast connections is undeniable and despite the considerable research effort spent during the past fifteen years, the IP multicast is just beginning to be deployed by ISPs and carriers. Many network-level issues have stalled the deployment of multicast. These issues include the design of robust inter-domain multicast routing protocol, global multicast address allocation and security concerns. Furthermore, IP multicast is an extension of the traditional best-effort IP unicast and hence there is no reliable delivery guarantees. Reliability is a prime requirement for several multicast applications, such as, software distribution, distributed interactive simulations, collaborative computing and distributed games. The proliferation of such applications motivates the design of many reliable multicast protocols being a topic of intense research and development efforts over the past fifteen years.

Different multicast applications have, in general, different requirements. For instance, real-time multimedia applications can tolerate some data loss while they require short delays. In this work, however, we were mainly concerned with bulk data transfer applications where a full reliability is required while there is no specific latency requirements. As the number of participants in such multicast applications grows potentially to hundreds or even thousands, any reliable multicast protocol has to provide viable mechanisms to achieve scalability. The design of a reliable multicast protocol that is scalable is of paramount importance. Impediments that face the development of scalable reliable multicast are mainly the well-known *feedback implosion problem* and *duplicate retransmissions problem.*

Feedback Implosion: as the number of the receivers grows, the probability of experiencing a loss by at least one receiver increases. A large number of receivers would generate a large number of requests to the source, thus wasting unnecessarily network bandwidth and overwhelming the source which has to supply repairs to the received requests. An increase in data delivery latency could also result from the feedback implosion problem.

Duplicate retransmissions: in a large multicast session, a data packet is likely to be lost by more than one receiver and thus, the source would send multiple repairs as a response to similar requests it receives. Moreover, if the source has to remulticast a lost packet on the entire group, receivers would be overwhelmed by processing already received data packets. In addition to unnecessary processing at the receivers, duplicate retransmissions (repairs) consume a big amount of network bandwidth.

Furthermore, congestion appears to be the main reason for data losses in the Internet and hence a congestion control protocol is a crucial component to achieve reliable multicast with scalable properties. Multicast congestion control has been an intense area of research during the last decade. One of the most challenges is fairness with other flows in the Internet and especially with TCP. Moreover, reliable multicast protocols are challenged by the Internet heterogeneity where a multicast source could be confronted with a wide range of path characteristics to each receiver. Heterogeneity could be simply accommodated by sending data with a rate that matches the worst receiver in the multicast group. Other approaches aim to improve the receivers' satisfaction and adopt a multi-rate mechanism. A multi-rate approach can be implemented either through a layered or a replicated scheme. In addition to waste of network bandwidth, data replication makes replicated schemes aggressive with other flows. Layered schemes are more economical, however they are complicated and do not provide a fine-grained control over bandwidth which only changes at the granularity of a layer.

Organizing the receivers into a hierarchy has proven to be the most successful approach that deals with both reliability and congestion problems. However, knowing that a multicast group is characterized by its dynamicity, the biggest challenge with hierarchical approaches is how to build such a hierarchy and then maintain its structure during all the session. According to the Deering's IP multicast model, the multicast tree is dynamic and hence, the adoption of a static hierarchy is not suitable. Adopting a dynamic recovery tree is better however, such a tree is difficult to be maintained in face of frequent membership changes. Moreover, in order to be efficient, the recovery tree has to be congruent with the underlying multicast distribution tree. The best way to build and easily maintain a hierarchy that satisfies the above constraints, is to exploit information about topology residing in routers. Most recent reliable multicast proposals benefit from the assistance of routers to provide efficient and scalable solutions by adopting the physical multicast tree or alternatively building a hierarchy with a topology as close as possible to the underlying multicast delivery tree.

The end-to-end principle [117] has been adopted for a long period by researchers in the design of network protocols that deal with reliability and congestion control. However, unlike unicast protocols, reliable multicast protocols have other problems and requirements such as scalability and heterogeneity. In fact, end-to-end solutions have found the problem of scaling to a large number of receivers since most of them require multicast members to know the group topology or at least relative locations of other receivers. On the other hand, routers have more detailed knowledge about the underlying multicast tree topology and a hierarchy produced by a router-assisted approach is expected to have better performances. Mainly, the hierarchy would be congruent with the underlying delivery tree and can be effectively used to recover from losses, scope retransmissions and aggregate feedback thus avoiding their implosion.

This work aims to provide some transport-level services for a multicast communication. Due to the potential presence of multiple receivers which could have different requirements and capacities, our main design goals are scalability, heterogeneity and dynamic membership support. In this work, we investigate router-assisted solutions where packets are processed in a customized way in order to provide solutions with better performances. Even though many difficulties still face their deployment, active networks are adopted in this work in order to allow for an easy implementation of our router-assisted solutions. The use of Active network concepts have been proposed in many research areas including multicast protocols, distributed interactive simulations and network management. They mainly have the ability to provide a very general and flexible framework for customizing network functionalities in order to gracefully handle heterogeneity and dynamics. Finally, it is worth noting that our proposals do not necessarily depend on pure active networking technology with their numerous security problems. Rather, our approach is closer to the new trends of introducing light software code in the network[1]. Furthermore, active networks related problems such as their large scale deployment, are beyond the scope of this thesis.

The remainder of this document is organized as follows. Chapter 2 is dedicated to the state-of-the-art. Afterwards, an analysis of already proposed active-based reliable multicast protocols is performed in chapter 3. The purpose is to evaluate the benefit of their different

[1]see INFOCOM 2003 keynote talk by Eric A. Brewer

adopted active services and to identify where contributions could be made. Then, chapter 4 introduces a new active service that consists in the early loss detection by the routers. An evaluation of this new service using a queuing system model is provided. Based on these analysises, a new reliable multicast protocol called DyRAM (standing for *Dynamic Replier Active reliable Multicast*) is proposed in chapter 5. Chapter 6 is dedicated to the performance evaluation of DyRAM. In chapter 7, a fair congestion avoidance algorithm called AMCA (standing for *Active-based Multicast Congestion Avoidance*) is proposed. In chapter 8, we address the problem of heterogeneity and AMCA is extended to support more heterogeneous receivers. Finally, chapter 9 concludes by summarizing the most important contributions of this work and giving some future directions.

Chapter 2

State-of-the-Art

2.1 Introduction

Multicast holds great potential to drastically improve data delivery to multiple participants by avoiding data duplication at the source. In the IP multicast [32] model, it is the responsibility of routers to ensure the packets delivery to their destinations. This is why there are several parameters that have to be addressed in the network level in order to support multicast communications. This mainly includes group addressing and membership management in addition to the presence of multicast-capable routers which assure the routing of multicast data. Multicast network level functionalities are overviewed in sections 2.2 and 2.3. However, many multi-point applications require reliability which is not ensured by IP multicast. Therefore reliability has to be provided at a higher level. The last 15 years have known a proliferation of reliable multicast proposals. An overview of the most significant among them is the subject of section 2.4. More recently, the problem of congestion control in multicast has received more attention. Section 2.5 gives an overview of the problems related to congestion control in a multicast communication with some of the solutions provided in the literature. Another concern inherent to the coexistence of multiple receivers with different capacities, is the accommodation of heterogeneity. This problem is also dealt with in section 2.6 where some of the proposed approaches are discussed.

2.2 Group Addressing and Membership Management

Due to the fact that the multicast address concerns a group of receivers rather than a physical IP address, a new addressing mechanism has been provided. Class D addresses (224.0.0.0 to 239.255.255.255) are reserved to multicast connections and are dynamically assigned via allocation protocols [121] to avoid collision between different sources multicasting on the same multicast address.

Moreover, a mechanism that allows any host to join and leave a multicast group is required. The purpose is to make the network aware of the receivers that are interested in a given multicast session. IGMP (Internet Group Management Protocol) specifies how any host should inform the network that it is joining or leaving a particular multicast group. In IGMPv1 [34], hosts simply join groups by advertising their membership on their local subnetworks. Leaving is achieved by timing out entries for hosts that did not persist sending such messages. IGMPv2 [48] compared to IGMPv1, allows hosts a quicker leave by adding a *leave group* message. IGMPv3 [18] adds support to selective source reception: a host can signal to a router which sources are allowed and which ones are not allowed to send.

2.3 Multicast Routing

IP multicast packets are sent to an arbitrary group of hosts represented by a multicast address which is independent of physical destination addresses. According to the IP multicast model, sources do not know the recipients to whom the packets are being sent. Thus, the network is responsible for appropriately routing data packets to the subscribed receivers. This is the responsibility of a multicast routing protocol which builds packet distribution trees in addition to performing efficient multicast forwarding where each packet exists only one time on any given network.

Two basic types of multicast delivery trees are built by the existing routing protocols: source-based trees and core-based (or shared) trees. A source-based tree is built with the source as the root and branches forming a spanning tree through the network to the receivers. Shared trees use a single common root placed at some chosen point in the network (the rendezvous point (RP)).

Source trees have the advantage of creating the optimal path between the source and the receivers. However, routers must maintain path information for each source which is not scalable due to memory consumption. In the other hand, shared trees have the advantage of requiring a minimum amount of state in each router. However, the placement of the rendezvous point (RP) is a key issue when implementing a shared tree-only environment.

Members of multicast groups can join or leave at any time; therefore the distribution trees must be dynamically updated. When all the active receivers on a particular branch stop requesting the traffic for a particular multicast group, the routers prune that branch from the distribution tree and stop forwarding traffic down that branch. If one receiver on that branch becomes active and requests the multicast traffic, the router will dynamically modify the distribution tree and start forwarding traffic again.

Multicast routing protocols can be divided into intra-domain and inter-domain protocols depending on their operational area:

2.3.1 Intra-domain Protocols

Intra-domain protocols are those used inside of a multicast domain to support multicasting. The most important ones are the following:

The Distance Vector Multicast Routing Protocol [125]

The Distance Vector Multicast Routing Protocol (DVMRP) defined in RFC-1075 is a distance vector routing protocol derived from the Routing Information Protocol (RIP), a unicast routing protocol. The main difference between RIP and DVMRP is that RIP calculates the next-hop toward a destination, whereas DVMRP computes the previous hop back toward a source. DVMRP uses a technique known as Reverse Path Forwarding (RPF) [30] which enables routers to correctly forward multicast traffic down the distribution tree. RPF makes use of the existing unicast routing table to determine the upstream and downstream neighbors. A router will forward a multicast packet only if it is received on the upstream interface. The RPF check helps to guarantee that the distribution tree will be loop-free. If a router is attached to a set of LANs that do not want to receive a particular multicast group, the router can send a *prune* message back up the distribution tree to stop subsequent packets from traveling where there are no members. DVMRP will periodically reflood in order to reach any new hosts that want to receive a particular group.

Compared to MOSPF (presented below), DVMRP is easier to implement and with lower computational complexity. Another feature of DVMRP consists in its support of tunnels between DVMRP routers allowing thus an incremental deployment of IP multicast in the multicast backbone (MBONE)[1]. However, DVMRP has significant scaling problems because of the necessity to flood frequently.

The Multicast Open Shortest Path First [94]

The Multicast Open Shortest Path First protocol (MOSPF) defined in RFC-1584 extends OSPF [95]. A source-rooted shortest-path tree is built by an MOSPF router on the reception of the first datagram from a source. Based on that, the MOSPF router caches multicast information including the (source, group) pair, the upstream node, and the downstream interfaces. The router makes its forwarding decision based on the contents of its forwarding cache. Trees must be recomputed when a link state change occurs or when the cache times out. Since MOSPF requires heavy computation for each source-group combination, MOSPF is best suited for environments with relatively few active source/group pairs at a time. Compared to DVMRP, MOSPF does not provide support for tunnels and does not require the forwarding of the first multicast packet on all the downstream interfaces.

The Core Based Trees [6]

The Core Based Trees routing protocol (CBT) builds a shared tree rooted at a *core* router, a fixed point in the network. Join messages are sent to the core and used by the intermediate routers to update the transient state for the group (incoming and outgoing interfaces). Unlike DVMRP and MOSPF, One shared tree per group in CBT requires maintaining only one state entry per group instead of one state entry per pair of group and source. Moreover CBT is independent of the unicast routing protocol while DVMRP and MOSPF relies on specific unicast routing protocol (RIP and OSPF, respectively). The disadvantage of CBT is the traffic concentration on fewer links due to the use of a shared tree when compared to protocols that use source-based tree schemes.

The Protocol Independent Multicast [33]

The Protocol-Independent Multicast (PIM) uses unicast routing information to perform the multicast forwarding function without depending on a particular unicast routing protocol.

[1] http://www.lbl.gov/web/MBONE.html

PIM provides two patterns, PIM-DM (PIM-dense mode) and PIM-SM (PIM-sparse mode); each of which is targeted to be used in a different environment. A dense mode protocol is designed to operate in an environment where group members are relatively dense with sufficient bandwidth. A sparse mode protocol would be efficient when members are widely dispersed across many regions of the Internet and bandwidth is not necessarily widely available.

PIM-DM [35] is very similar to DVMRP, however it maintains its own routing table instead of using a unicast table. Moreover, PIM-DM simply forwards multicast traffic on all downstream interfaces until explicit prune messages are received instead of calculating a set of child interfaces for each (source, group) pair.

PIM-SM [42] like CBT builds a shared tree (rooted at a rendezvous point (RP)) where only receivers that have explicitly requested the data will receive the traffic. However, unlike CBT, the built tree is unidirectional in PIM-SM. Another difference is that PIM-SM could switch to a source-rooted tree instead of the shared one. One of the PIM-SM limitations is its complexity when compared to DVMRP or MOSPF and still requires routers to maintain a significant amount of state information to describe sources and groups.

2.3.2 Inter-domain Protocols

Inter-domain protocols are those that operate between multicast domains, used by ISPs to forward multicast traffic on the Internet. One of the first proposals consists in using PIM-DM as the intra-domain protocol and PIM-SM at the inter-domain level.

However, the presence of the sources and RPs in different domains yields the problem of ISPs interdependency. MSDP (Multicast source Discovery Protocol [47]) was introduced as a solution to the interdependence problem. In fact, with MSDP there will be no shared tree built across domains. Moreover, MSDP allows each ISP to have its own local RP(s). Thus an ISP does not have to depend on RPs located in a third-party ISP's domain.

MSDP is seen as the near-term standard; for the long-term, it is envisioned to use BGMP (Border Gateway Multicast Protocol) [43] which has been designed to inter-operate with any multicast routing protocol deployed in the inter-domain level. BGMP is associated with MASC (Multicast Address-Set Claim) for dynamic multicast address allocation. BGMP is implemented at the border routers (that also run intra-domain protocols) and construct a bidirectional shared tree connecting the multicast trees built in the different domains.

2.4 Reliable Multicast Protocols

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 where all packets are safely delivered to the destinations. The problem of reliability in multicast has been quite widely covered during the last 15 years and meeting the objectives of reliable multicast is not an easy task.

Many reliable multicast protocols rely on retransmissions (ARQ, Automatic Repeat reQuest) [52, 102, 137, 27, 110, 57, 101, 118, 74, 64] while some recent proposals are based on the ability of generating some redundant packets (FEC, Forward Error correction) to anticipate potential losses [90, 84, 107, 58, 109, 16]. Other protocols combine FEC with ARQ mechanisms to achieve reliability [96, 114]. Compared to FEC, ARQ-based protocols do not require any processing (coding/decoding) of the data stream. However FEC are necessary when there is no feedback channel and more appropriate to real-time applications where the recovery delay has to be as small as possible. Motivations behind FEC approaches are also their ability to accommodate receivers heterogeneity (section 2.6). However in a FEC approach, the sender sends parity packets even in the absence of losses which consumes network bandwidth in addition to the coding/decoding overhead at the sender and the receivers. Furthermore FECs have proved to be effective in the absence of burst losses.

From another perspective and depending on whether some in-network assistance is exploited, reliable multicast protocols proposed in the literature can be classified into *end-to-end* and *router-assisted* schemes. In what follows, we limit ourselves to ARQ-based approaches and overview both of the end-to-end and router-assisted schemes in subsections 2.4.1 and 2.4.2 respectively.

2.4.1 End-to-end Schemes

Most of the reliable multicast protocols either use a sender or a receiver-initiated mechanism depending on which entity is responsible for the detection of losses. In Sender-initiated protocols, the sender is responsible for both the loss detection and its recovery. A receiver has to send a positive acknowledgement (ACK) for every packet it receives. The sender detects losses on timeout or duplicate ACKs. Sender-initiated protocols do not scale well to a large number of receivers due to the ACK implosion problem. The sender has to maintain and process a large amount of state information for each receiver. The analysis conducted in [103] showed that in a sender-initiated approach, the overall throughput is rapidly severely reduced.

Receiver-initiated protocols move the loss detection responsibility to the receivers. The receivers maintain their status locally and explicitly requests retransmissions by sending negative acknowledgements (NACKs). Compared to a sender-initiated approach, a receiver-initiated one reduces the amount of feedback and thus is more scalable. However, if the sender still be the responsible of retransmitting lost data packets even in a receiver-initiated protocol, the NACK implosion problem will become apparent when a large number of receivers have subscribed to the multicast session.

The next step in the design of reliable multicast protocols was the distribution of the loss recovery task among the receivers in addition to the source. A receiver is able to retransmit (if available) a data packet requested by another receiver in the vicinity, thus reducing the recovery latency. This is referred to as *local error recovery* and has been adopted by many protocols in the literature [119, 52, 102, 137, 27].

Scalable Reliable Multicast (SRM) [52] designed to assure reliability to a white board (wb) application on the MBONE, is one of the first proposed protocols that exploit local recovery from the receivers. Upon the reception of a loss, a request is sent on the multicast address. Any subscriber to the group could process this request if it has already received the corresponding data packet. In order to avoid that all the receivers be flooded by multiple NACKs and/or duplicate repairs, a probabilistic approach is adopted to perform NACK/repair suppression. This approach is similar with some enhancements, to the one originally proposed by [119] in the context of the *eXpress Transport Protocol* (XTP). Upon the detection of a loss, a receiver waits for a random period determined by its distance from the original source of data before sending its NACK. In such a way, the closest host to the point of failure is likely to timeout first and multicast the request even if a number of hosts may miss the same packet. These hosts will hear that request and suppress their own and behave as they have sent the corresponding NACK. Similarly and upon the reception of a request, a receiver (if it has already received that corresponding packet) will set a repair timer to a random value depending on its distance from the request originator. A host close to the affected area is likely to timeout first and send the corresponding repair. This timer will be canceled if the repair is received; otherwise the requested packet is retransmitted on timer expiration.

A NACK-based approach do not allow the sender to free its buffers kept for potential retransmissions. This is why a combined ACK/NACK approach could be a better solution when combined with a structure-based approach like RBP [22], RMTP [102], TMTP [137] or TRAM [27] (each of them will be described below). Organizing the receivers in a logical structure is mainly adopted to perform feedback aggregation/suppression, thus avoiding the feedback implosion problem. Moreover it provides a better control regarding the request/repair exposure. In SRM, a TTL-based approach has been proposed to limit the scope of NACKs and retransmissions, however this still an immature solution. How the TTL field is set is not an easy task since for instance, a request could not reach a potential replier. In structure-based approaches, requests are sent to a dedicated receiver instead of multicasting them to the whole group. Thus avoiding the *crying-baby* syndrome that can be observed in a SRM session where a weak receiver is overwhelming the others with very frequent requests.

RBP (*Reliable Broadcast Protocol*) [22] provides reliable multi-point communication between sites connected via a local area broadcast network. It adopts a virtual token ring-based strategy and combines ACKs and NACKs to assure reliability, ordering and fault tolerance. Messages in RBP are multicast to the group through a dedicated node called the token site which acknowledges each message it receives by multicasting an ACK to the group. ACKs are globally timestamped and used by the receivers to order messages and detect losses. If a message or an ACK is lost, a NACK is sent to the current token site. The message is then retransmitted in unicast to the site that requested it. The role of the token site rotates among all group members in order to balance the ACK load. Before accepting the token, the new site is required to have all the messages earlier than the timestamp of the passing message.

More sophisticated structure-based approach which are better suited to a wide area network have been proposed. For instance RMTP (*Reliable Multicast Transport Protocol*) [102] adopts a tree-based approach where the receivers are organized through a hierarchy of local regions with for each of which a receiver is designated to act as a representative. RMTP supports a multi-level hierarchy of designated receivers (DRs). Receivers and DRs periodically send reports about their reception status to their parents instead of overwhelming the source and the network, thus avoiding the feedback implosion problem. A parent node is responsible for recovering from losses experienced by its children, thus reducing significantly the recovery latency. The source or a DR could decide to retransmit a data packet using multiple unicasts or by multicast to the entire region.

One concern in a logical tree approach is how the hierarchy is built and maintained as far as there are receivers subscribed to the multicast session. In RMTP, each DR as well as the sender periodically sends a special packet (SEND_ACK_TOME) in which the TTL field is set to a predetermined value down to each receiver. A receiver will choose the DR whose SEND_ACK_TOME packet has the largest TTL value (which corresponds to the nearest DR in terms of number of hops). According to the IP multicast model [32], the composition of a multicast group is dynamic since receivers can join and leave the group at any time. However, the RMTP tree is static and does not adapt to dynamic changes in the multicast topology. Moreover, it appears that the TTL-based approach used by the receivers to determine their DRs, could lead to unbalanced load on the different DRs.

Tree-based Multicast Transport Protocol, TMTP [137] is another tree-based protocol which try to dynamically maintain the logical hierarchy with some load balancing on the designated receivers which are referred to as *domain managers* (DMs). TMTP uses an *expanding ring search* to dynamically organize the receivers into the hierarchy as members join and leave the group. Load balancing is achieved by bounding the number of associated receivers to each DM. Furthermore TMTP combines periodic ACKs and restricted NACKs with an SRM-like suppression mechanism.

TRAM [27] is another tree-based protocol that adopts many of the techniques presented in RMTP and TMTP. The tree construction is dynamic as in TMTP. However, TRAM supports many tree construction techniques each of which is suitable to a different network environment. In addition to LAN-based tree formation, unidirectional, bidirectional and combined approaches are provided. To ensure reliability, TRAM uses a selective acknowledgment (SACK) mechanism which combines at the same time positive and negative acknowledgements. TMTP and TRAM adapts to dynamic changes in the group topology thanks to their dynamic tree structure maintenance. Both of them use a TTL-based approach to form and update the control tree. However, current multicast routing protocols may not provide accurate TTL values for that purpose. Moreover, a node's parent could be located downstream in the physical multicast tree which increases the repair latency.

Other enhanced tree-based reliable multicast protocols include *STORM* [44], *Tracer* [76] and *OTERS* [79]. STORM proposes a new approach to organize the receivers in a hierarchy based on observed performance, propagation delay and buffer size. In Tracer, each receiver uses the mtrace utility [49] to advertise its path to near-by receivers using expanding ring search. This mechanism can be used to build congruent recovery trees for tree-based protocols. OTERS (*On-Tree Efficient Recovery using Subcasting*) implements two network layer extensions, the *multicast route backtracing* and the *subcasting*. These two extensions allow for building a congruent recovery tree as well as to repairs scope limitation. A combined group/tree approach to achieve reliability in the context of a many-to-many communication is adopted in GAM *Group-Aided Multicast* [138].

The *Multicast File Transfer Protocol* MFTP [110, 7] adopts a very different ARQ-based approach from those presented until now. In MFTP, the file to be multicast is partitioned into packets and equisized blocks are formed. In a first pass, the source sends the entire file at a pre-defined transmission rate. Subsequent passes are used to retransmit lost packets. Lost packets are determined based on the NACK-bitmap messages sent by the receivers. To avoid feedback implosion, receivers delay the emission of their NACKs with a given period. MFTP is not suited to time-sensitive approaches since the losses are repaired at the end of each pass.

2.4.2 Router-assisted Schemes

As have been previously seen, local error recovery is a key feature in the design of reliable multicast protocols. Local recovery allows for reducing the recovery latency in addition to unloading the source. The end-to-end tree-based approaches have proven to be the most efficient [75]. They provide a better solution to the feedback implosion problem, reduce the amount of duplicate repairs and limit the scope of requests and repairs. However:

- end-to-end related tree construction mechanisms proposed do not provide necessarily an optimal recovery tree,
- due to the dynamic nature of a multicast group, it appears very difficult to maintain such a recovery tree structure.

This is mainly due to the fact that the tree-based protocols overviewed in the previous subsection, are end-to-end and lack to topology information which resides in the routers. This is why, some protocols try to benefit from the help of routers to maintain their loss recovery tree as close as possible to the underlying multicast tree. These protocols can be divided into two classes:

- the first class includes protocols that use the physical multicast tree as the recovery one with in-network data caching [74, 64, 21],
- the second class includes protocols that maintain (thanks to some routers assistance) a logical recovery tree close to (but different from) the physical multicast tree [101, 118, 77].

The first class protocols usually rely on the active networking technology [120] where the routers are able to perform customized services on the packets flowing through them. ARM (*Active Reliable Multicast*) [74], AER (*Active Error Recovery*) [64] and RMANP (*Reliable Multicast Active Network Protocol*) [21] belong to this class.

ARM [74] proposed in 1998 by Lehman et al., is one of the first protocols that benefit from active routers to achieve reliability. The ARM's main active service is its *router-based* local recovery which consists in the *best effort* cache of data packets for possible retransmissions. A receiver experiencing a packet loss, sends immediately a NACK to the source. An active router that receives a NACK, retransmits the corresponding repair if available in its cache; otherwise the NACK is forwarded upstream in the multicast tree. To overcome the NACK implosion problem, an active router forwards only the first NACK received and ignores subsequent similar ones for a given period of time. Furthermore, an ARM active router sends a repair only on the links on which NACKs have been received. In such a way ARM avoids the problem of duplicate repairs.

AER [64] is another active-based reliable multicast protocol. It implements local recovery by caching data packets at servers co-located with active routers. AER differs from ARM in the mechanism adopted to suppress duplicate NACKs. It uses a strategy similar to the one used by SRM 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. Upon the reception of a NACK, an upstream router multicasts this NACK downstream to refrain other downstream receivers from sending their NACKs. As opposed to ARM, a router in AER multicasts repairs on all the downstream links.

RMANP [21] like ARM and AER implements an active-based local recovery, however it tries to solve the problem of freeing retransmission buffers at both the source and the active routers. To do so, in addition to data and repair capsules, RMANP designers introduce positive (ACKs) and *multiple* acknowledgements (MACKs). A router would retrieve a data packet from its cache on the reception of an ACK from all of its descendants. ACKs are sent upstream periodically and contain the accumulative sequence number that has already been acked by all the direct descendants or have been stored by the router itself. In the latter case, an ACK is used to free the cache at upstream routers to avoid that a data packet is being stored at all the routers located between the source and a given receiver. MACKs are sent (on a request from the source) by the routers that have receivers with direct descendants. The aim is to allow the source to free its retransmission buffer. Moreover, RMANP takes advantage of the adaptability and flexibility of active networks to provide different services depending on the type of received capsules (unreliable, reliable and time-constrained reliable data capsules).

The use of in-network elements especially the routers to perform the cache of data packets is not scalable. In fact, caching means at the routers are limited and one router has to support multiple multicast sessions at once. This is really the case in the presence of high loss rates where the routers have to cache more and more data packets in order to satisfy the different requests they receive. LMS (*Lightweight Multicast Services*) [101], PGM (*Pragmatic General Multicast*) [118] and AIM (*Addressable Internet Multicast*) [77] fall in the second class of router-assisted protocols. Instead of adopting the physical multicast tree as the recovery one (with caching within routers), they use a logical tree built implicitly with the assistance of the routers. This has the main advantage of requiring a memory usage as low as possible within network elements and is potentially more scalable.

In LMS, every router in the multicast tree designates one of its downstream links as a replier (leader). Upon loss detection, a receiver multicasts a NACK with a special IP option so it will be intercepted by the routers. Routers forward NACKs they receive on their replier link except those from the replier link which are forwarded upstream. This allows for forwarding only one request per subtree, thus avoiding NACK implosion. Before forwarding a request, a router inserts its address and the identifier of the link from which the request has been received. This router is referred to as the *turning point* and is the root of the affected subtree. A replier that receives a NACK, sends the repair to the corresponding turning point. This latter will in turn forward the repair on the marked links.

LMS suppresses NACKs by allowing only one NACK traveling upstream however there is no NACK suppression within the affected subtree. In fact, a NACK is always forwarded on the replier link even if this latter is itself affected by the same loss. A replier link could see the same NACK as many as there are links (except itself) downstream its upstream router. Furthermore, the non-reception of a packet by the replier link will necessary result in forwarding the NACK upstream even if another receiver downstream has received it. Finally, there is no load-balancing among the links. A replier link could be responsible of retransmitting all the missing packets by its subtree receivers.

PGM (described in RFC-3208) also uses some router assistance to achieve scalability through a hierarchy of PGM routers (identical to the multicast tree when all of its routers are PGM capable) and a hybrid ARQ/FEC approach. PGM hierarchy of routers is maintained thanks to *source path messages* (SPMs) sent periodically by the source. For NACK suppressing, PGM adopts a mechanism similar to the one proposed in AER [64]. Prior to sending a NACK, a receiver waits for a back-off random interval. Upon the reception of a NACK, a PGM router multicasts a NACK confirmation (NCF) to acknowledge the reception of the request in addition to refrain the other receivers from sending their waiting NACKs. This PGM router creates a *retransmit state* with the sequence number of the requested packet with the link on which the NACK has been received. Moreover, this PGM router will unicast repeatedly this NACK to its upstream PGM router until it receives the corresponding NCF. If similar NACKs arrive from other links, a PGM router will add these links to the retransmit state without propagating them upstream.

In addition to NACK suppression, network elements in PGM examine repairs in order to forward them only to the links on which NACKs were received for them. A repair data is either the lost packet or a FEC packet. FEC may also be pro-active where redundant packets are sent with the original data without waiting for feedback from the receivers. Repairs are generated either by the source or a designated local repairer (DLR), thus allowing for a local recovery. A DLR announces its presence with a multicast message and caches all received packets in the transmit window. Afterwards this DLR behaves as the source would do. A DLR could be a host node located upstream of the affected subtree or one PGM hop downstream from the network element (*off-tree* DLR). Off-tree DLRs are discovered via a polling mechanism which is also used by a PGM parent to estimate the number of its children in order to adapt the waiting times at the receivers prior to send NACKs.

PGM provides reliability only within the current transmit window and consequently is better suited to applications that can tolerate some lost packets or support some application-level recovery. This is due to the exclusive use of NACKs.

AIM [77] (*Addressable Internet Multicast*) is an attempt to extend the existing multicast services by assigning per multicast group labels to the routers of a multicast tree. There are three types of labels: *positional*, *distance*, and *stream labels*. A router positional label specifies its location relative to the addressing root of the multicast tree. Positional labels are used to route messages to a subset of the members of the group. Every tree interface is assigned a distance label that specifies the shortest distance to another router with member hosts. Distance labels are used to perform *reachcast* where a request is sent to the closest host. A stream label specifies a subscription by one or multiple receivers to a given traffic. In RMA (*Reliable Multicast Architecture*) based on AIM, a logical recovery tree is implicitly constructed while preserving relationships between the receivers. An affected receiver would ask its local router to reachcast the request. The reachcast could reach other members that have the requested data, which respond by sending a retransmission via positional routing.

The overhead of distributing the labels after a membership change increases with the dynamicity degree of the group. Whenever a new branch is added to the multicast tree, all the routers below the new branch may have to change their labels.

To enhance robustness, authors in [29] proposed a new mechanism called *randomcast* where a router sends a request on a randomly chosen upstream or downstream link. *Search Party* [29] is an LMS-like reliable multicast protocol that uses the randomcast forwarding service. In the active-networking class, we can mention GMTS (*Generic Multicast Transport Services* [20], a set of active networking services intended to help protocol designers to overcome some problems inherent to reliable multicast delivery such as NACK suppression and subcast retransmissions. GMTS has led to GRA (*Generic Router Assist*) [19], an Internet draft. An other active-based reliable multicast that improves the performances of active-based reliable multicast with in-network data caching by introducing FEC mechanisms is APES (*Active Parity Encoding Services protocols*) [113].

2.5 Multicast Congestion Control

A first congestion control algorithm known as *TCP Tahoe* has been proposed by Jacobson [59] after a series of congestion collapses experienced in 1986. Since then, several variants have been implemented. In *TCP Reno* the sender uses the loss of packets as a negative feedback to regulate its window size (used as the regulation parameter). Initially, the congestion window size is set to one and is increased by one packet per one RTT. This is the *slow start* phase which allows for probing available bandwidth by exponentially increasing the congestion window. When the window size reaches a given threshold, the source enters the *congestion avoidance* phase where the window size is increased linearly by the reciprocal of the current window size on the reception of an ACK. A loss is detected on the reception of three duplicate ACKs and the sender, as a result, reduces its congestion window by one half and enters the congestion avoidance phase. A severe congestion is considered on a timeout which causes the congestion window size to be dropped to one.

In TCP Reno, when the source does not detect congestion, it continues to increase its window size by one every one RTT. This is the reason for which TCP Reno is biased against longer delays connections. TCP Vegas [14] introduced in 1994 appears to do not suffer from delay bias and to be less oscillative than Reno while achieving better throughput and fewer losses. On one hand, TCP Vegas performs a new retransmission mechanism which reduces the time to detect lost packets from the third duplicate ACK to the first or second duplicate ACK. On the other hand, TCP Vegas is fundamentally different from TCP Reno. It uses queuing delay as a congestion measure which can be estimated from the experienced end to end delays or alternatively from the difference between the emission and reception rates.

Attempts to model the behavior of TCP have led to equations that give the throughput of a TCP connection as a function of the loss rate, the round trip time and the packet size.

These equations are known by *TCP formulas*. The first proposed formula is known as the TCP simple equation [89] which holds for low loss rates and, under normal operating conditions, is given by:

$$T = \frac{C\ S}{RTT\ \sqrt{p}} \tag{2.1}$$

where T is the throughput in bits per second and S is the data packet size in bits. C is a constant to be taken from $[0.9, 1.3]$. In the steady state, C, RTT, and S are constant. Another more accurate formula known as the complex equation has been demonstrated in [99]. Such an equation has been used to derive a simpler one to be used as the control equation for the congestion control in TFRC (TCP-Friendly Rate Control) [51]:

$$T = \frac{S}{RTT\sqrt{\frac{2p}{3}} + T_{RTO}\ (\sqrt{\frac{3p}{8}})\ p(1+32p^2)} \tag{2.2}$$

where T_{RTO} is the initial timeout for TCP, in seconds. This equation is used to calculate an upper bound on the throughput to be not exceeded by the source so the sending rate will not be aggressive. In a formula-based unicast congestion control, equation parameters have to be determined depending on the chosen throughput function. The loss rate p could be calculated by the receiver. The round trip time RTT could be measured at either the sender or the receiver. The initial timeout t_{RTO} can be deduced from RTT. In TFRC, the receiver calculates p and feeds it back to the source which calculates the throughput using (2.2). The regulation parameter adopted in TFRC is the rate instead of the window size since the TCP equation gives an upper bound for the transmission rate. One limitation of an equation-based approach is that the estimation of the loss rate remains an open issue and is difficult to do.

To improve the performances of TCP, router-assisted schemes have been proposed. For instance, RED (Random Early Detection) was proposed as a different packet drop mechanism in the routers' queues. In RED, a packet is dropped with a probability that increases with the mean queue length in contrast to the traditional drop-tail where packets are dropped when the queue is full. Losses occur randomly which allows for better cooperation among the TCP sources since these latters will not react at the same time to losses. However RED increases packet loss rate even in the absence of congestion which would reduce the throughput. Moreover, the parameters of a RED mechanism are difficult to tune. TCP ECN (Explicit Congestion Notification) [104] is a variant of TCP RED where a router marks packets (with a special field in the IP header) instead of dropping them. In such a way, a receiver is able to detect its connection congestion state. The source is also able to detect a congestion situation without losses thanks to the marks copied in the ACKs sent by the receiver. Moreover, in TCP ECN, a router notifies the sender allowing it to react appropriately. [40] is one proposal that uses ECN and ICMP source quench messages as feedback to inform the source about a network congestion. The proposed mechanisms in [40] are intended to be used either in unicast or multicast

More recently, active networks where routers are able to perform customized services on the packets flowing through them, have been exploited to propose more responsive congestion control algorithms. ACC (Active Congestion Control) [45] is one active scheme that makes feedback congestion control more responsive to network congestion using active networks. ACC includes necessary information in the packets such as the sender's current estimation of RTT and its window size. When a router detects a congestion, it decides what action the endpoint would take if it had detected the congestion. The router calculates the correct window size for the sender from the information provided in the packets. Afterwards the router installs a filter at the previous router on the connection's path. This filter deletes packets that the endpoint would not have sent. In meanwhile, a packet with the new calculated window size is sent to the sender which will react to the congestion accordingly. Simulations performed on ACC (compared to the traditional TCP) shows a throughput improvement up to 20% with bursty traffic and mainly in connections with high delay-bandwidth product. This is very promising since connections with high delay-bandwidth product present an unforeseen behavior. The larger the end-to-end delay in a network, the longer until the sender learns about a congestion. The higher the bandwidth of the network, the larger the amount of data the sender continues to send into a congested network before reacting to the congestion. A limitation of ACC is that it detects congestion due to losses which are only observed at the active routers. Losses experienced by other passive nodes are not taken into consideration by the active mechanism (although considered by the TCP-like end-to-end mechanism).

When a sender has to send the same information to more than one receiver, unicast becomes inefficient and the use of multicast is more appropriate. Congestion control is then much more difficult due to the presence of multiple receivers with different end-to-end paths that could share some branches in the multicast tree. In order to be reactive to congestion situations, feedback from the receivers are required. A feedback could be a NACK to indicate a missed packet or an ACK packet to inform the source that a packet is correctly received. Other feedback forms with different contents can be used. In multicast, feedback from the receivers reflect diverse congestion conditions in various parts of the network. Feedback suppression and/or aggregation is required to be more scalable to a large number of receivers. However feedback suppression could yield to a lack of information at the sender making it very unresponsive to congestion variations. The network becomes then an open loop control system and will be more difficult to be controlled. In other words, we have to deal with the feedback implosion problem while allowing the necessary information to be given to the source. A main issue of a congestion control algorithm is how, where and when such feedback have to be suppressed and/or aggregated.

2.5.1 Congestion Feedback Suppression

In contrast to unicast, in multicast there are multiple different end-to-end paths and a single loss has to be distinguished from multiple losses. A single loss may affect many receivers if their end-to-end paths to the source share the lossy link. If the source reduces its rate in response to every congestion indication it receives, its rate will be severely reduced causing the *drop-to-zero* problem [108]. Motivated by the stochastic nature of loss events particularly in a multicast Internet group, RLA (Random Listening Algorithm) [126] uses a probabilistic approach to suppress feedback. A loss is detected by the sender on timeout or discontinuous ACKs. The sender keeps track of the loss rates and RTTs to the different receivers. Based on these information, the sender reacts to a congestion indication with probability $1/n$ where n is the estimated number of the most congested receivers. In this way, the sender responds according to the average instead of the maximum loss rate experienced by the receivers.

In LTRC [92], receivers estimate their average loss rate and feed it piggybacked on NACKs to the source which will change the rate accordingly. LTRC tries to react only to the most congested paths by using a threshold-based mechanism. The source would apply a big rate change only for persistent losses. LTRC reacts to only one loss indication in a time period estimated as the required time by a rate change to take effect in the system. This is due to the fact that reacting to network conditions is appropriate if these conditions are observed over a sufficiently long period of time, consisting of at least several RTTs. One issue related to this approach is how to make the tradeoff between the "drop-to-zero" problem and the responsiveness of the system. Furthermore, LTRC performs no feedback suppression before reaching the source, hence it still suffer from feedback implosion problem. Consequently, we have to address the feedback implosion problem before reaching the source by adopting a feedback suppression combined or not to an aggregation mechanism. One suppression approach consists in choosing one or more receivers to act as representatives of the whole group. These representatives would send feedback to the source instead of all the receivers do.

In [36], a small set of receivers are dynamically chosen as representatives by the sender based on their feedback. Representatives provide immediate feedback to the source allowing it to adjust the current state of the session. A probabilistic approach is used to limit feedback from subtrees not yet covered by the representatives. Feedback from the non-representative receivers is scheduled over a random interval. To avoid congesting the network with ACKs, representatives send ACKs with probability $1/R$ where R is the size of the representative set.

PGMCC [108] selects one receiver to act as a representative designated by the "*acker*". Every receiver estimates its capacity using the simple TCP formula (2.1) and feeds it back to the source piggybacked on NACKs. Based on that, the sender selects as the acker the receiver with the lowest capacity. Once the acker is chosen, a closed control loop is run between the sender and the worst receiver in the group. The acker acknowledges every received packet allowing the source to react to congestion in a TCP-like manner. NACK suppression among the receivers is performed in a randomized manner to avoid their implosion.

One concern with a representative approach is to choose appropriately the representatives so they reflect the congestion situations. How to select the right size and the suitable set of representatives needs further investigation. Other schemes deal with the congestion feedback implosion problem by adopting a hierarchical structure in a similar way that reliable multicast protocols do. Feedback are propagated and aggregated through the tree structure. For instance, TRAM [25] and MTCP [106] are based on a logical recovery tree constructed with the receivers. An intermediate receiver node stores a received packet until it has been acked by all of its children. Feedback are also aggregated through these intermediate nodes.

An MTCP parent maintains two windows, a congestion window which is managed similarly to that of TCP (with slow start and congestion avoidance) and a transit window that keeps track of the amount of data that the children of a parent have not yet acked. For each ACK, a parent node transmits a congestion summary (CS) to its parent. CSs are aggregated in intermediate nodes thus avoiding the feedback implosion problem. A CS contains the minimum of its own congestion window size and those reported by its children and the maximum of its transit window size and those reported by its children. The sender is allowed to transmit the difference between the minimum congestion window and the maximum transit window.

In TRAM [25], the sender adjusts its rate according to receivers feedback (congestion reports). Receivers send congestion reports per one window to their parents (repair heads of the recovery tree [27]). A TRAM ACK contains the highest consecutively received packet sequence number H_r and the highest allowed sequence number H_a. Repair heads aggregate the congestion reports on the way to the sender by sending its own H_r and the minimum H_a of those reported from its children. A compromise between optimizing the speed of updating the sender's window and the potential overhead of additional ACKs is achieved by just sending ACKs where the new computed H_a^{new} satisfies $(H_r^{new} - H_a) > W_a/2$.

2.5.2 Fairness with TCP

As have been pointed out, feedback suppression is essential and has to be performed appropriately. Inadequate suppression of feedback makes the system very unresponsive to variations in the network which is a potential source of unfairness. However, fairness with other multicast flows and especially with TCP is required since most of the Internet traffic is transported by TCP. Responsiveness is more required in multicast than in unicast since the network multiplies the traffic injected by the source along the distribution tree. The larger the responsiveness, the smaller the potential oscillations and self-synchronizing behaviors are. A multicast congestion control algorithm has to be as responsive as the TCP does in order to be TCP-friendly. A reliable multicast connection is considered to be TCP-fair if it is fair with TCP flows on the worst link that it crosses. TCP is known to be window-based fair which means that each connection is given a throughput inversely proportional to its RTT. This fairness is the same as the "global fairness" introduced in [128].

To be TCP-friendly, congestion control algorithms either emulate the behavior of TCP by using a similar window-based approach or by using the TCP formula. In a formula-based approach, information about the packet loss rate and RTTs are collected. Afterwards the TCP formula is used to compute the corresponding rate. It generally adopts a rate-based control in opposition to the window-based one adopted in TCP. One of the congestion control algorithms that uses the TCP formula is TFMCC [131] which is an extension to multicast of TFRC [51]. Each receiver measures its own loss rate and its RTT to the sender and computes the corresponding rate using equation (2.2) as if it was connected to the sender via TCP. If a receiver sends feedback with rate less than the current emission rate then the latter is dropped to the fed back one. To avoid feedback implosion, a receiver would send its rate only if it is less than the current emission one. The receiver which has the lowest computed rate is considered as the CLR (Current Limiting Receiver). A CLR is then allowed to send feedback without any suppression, so the sender can use the CLR's feedback to increase the transmission rate. A CLR will change if another receiver sends feedback indicating that a lower transmission rate is required. The use of TCP equation is promising to be TCP fair, provided that the receivers RTTs and loss rates are known. However measuring accurately the RTTs and especially the loss rate is a real challenge. There is a trade-off to be considered between responsiveness and accuracy when trying to measure these quantities.

It is worth mentioning that measuring the loss rate experienced by every receiver can be avoided if a TCP-like window-based scheme is adopted. In MTCP [106] for instance, in addition to the source, every intermediate node manages its congestion window with a slow-start and congestion avoidance phases. The congestion window size is incremented only on receipt of ACKs from all its children in the logical tree. A TCP-Vegas like congestion control is also used in conjunction with the previous one in order to detect congestion even during slow-start. RLA [126] is another window-based regulation scheme that achieves an acceptable degree of fairness designated by "essential fairness" in opposition to an absolute fairness very difficult to be obtained. Absolute fairness requires that the multicast session is treated as a single session and equally shares the bottleneck bandwidth with competing TCP connections on its soft bottleneck paths. According to [126], a multicast session is essentially fair to TCP if its average throughput λ in the long run is bounded by:

$$a\ \lambda^{TCP} < \lambda < b\ \lambda^{TCP}$$

where λ^{TCP} is the average throughput of the competing TCP connections on the soft bottleneck path. a and b are functions of N (number of receivers) such that $a \leq b < N$. It is worth noting that absolute fairness is a special case of essential fairness with $a = b = 1$.

2.5.3 Window-based versus Rate-based

As have already been seen, the regulation parameter of a congestion control algorithm is either the size of a window [106, 108, 126] or the transmission rate [27, 92, 131]. In a window-based regulation, the source can send new packets into the Internet whenever there is space in the window. The main feature of a window-based congestion control is that it integrates naturally with error and flow control and reacts rapidly to network changes. In a rate-based approach, the source can inject new data no higher than a given rate. The main feature of a rate-based control is that it generates traffic into the network smoothly and is better suited to continuous media applications. A rate-based control is better suited to multicast since it is more scalable and easier to be implemented. Moreover, the rate attainable can be controlled to be independent of the RTT (hence insuring a rate-based fairness which tries to achieve equal throughput at the bottleneck link). However TCP fairness is known to be *window-based* [55] where packets tend to arrive in clusters and any different pattern (such as a rate-based regulation) will produce different loss rates through drop-tail gateways [126]. The RTT must be known to a rate-based approach to be TCP-compatible fair (window-based fairness). RTT estimation is difficult in multicast because of the problem of feedback (ACKs) implosion at the source. This problem can be overcome using a hierarchical approach.

The proposal of [136] is an attempt to benefit from the advantages of the two approaches and combines the fast dynamics of window scheme with the flow smoothness of a rate scheme. It uses the rate as the regulation parameter. However the rate is computed by every receiver from a congestion window size updated in a TCP manner. Each receiver performs a standard TCP congestion control and updates the window size w according to the experienced losses. The transmitting rate r is then computed using $r = w/RTT$.

2.5.4 Receiver-driven Approaches

In congestion control algorithms, the sender (like in TCP) usually controls the rate for the whole group. Such an approach requires explicit monitoring at the source in regards to every individual receiver. On one hand this is not appropriate for most of the applications and not compatible with the IP multicast model [32]. On the other hand the larger the receivers group is, the more complex the congestion control tasks are. Furthermore because of local recovery and feedback suppression performed in many congestion control algorithms, the loss information to all the receivers is not automatically available at the source.

A step in gaining congestion control algorithms complexity is to distribute responsibilities between the sender and the receivers to be more scalable. In fact a congestion situation could be detected by the receivers instead of the source. This is mainly interesting when a drop decision has to be taken, a receiver drops itself from the multicast group if it experiences losses even when the source transmits with a specified minimum rate. The receivers could also contribute to the congestion control tasks by estimating their loss rates as in [131] (see section 2.5.2). In [55], an AIMD [24] window-based congestion control is run at every receiver. A receiver updates its window size and sends the necessary feedback to the source. A window at each receiver is equivalent to one window at the source with the worst representative. It is worth mentioning that layered approaches (see sections 2.5.5 and 2.6) are the main schemes that involve actively the receivers in the congestion control tasks.

2.5.5 Single-rate versus Multiple-rates

In all the presented congestion control algorithms until now, the sender controls the rate dictated by the worst receiver in response to the received feedback. These controllers are known as *single-rate* schemes where all the receivers get the same data rate. The main advantage of such approach is that the source is not required to transmit multiple streams or use special data encoding. However multiple receivers in a multicast session means dealing with heterogeneity. First, in presence of large and heterogeneous groups, the problem is how to determine the slowest receiver knowing that there may be local recovery, feedback suppression and correlation in loss rates. Secondly, the worst receiver will slow down the throughput to all the receivers. Additionally as have been seen in a previous section, congestion control decisions are not always implemented by the source. For instance, a congestion control algorithm decision can be the action of dropping a receiver which must be implemented by the receiver itself.

Multi-rate schemes are based on the ability to generate the same data at different rates over multicast streams. Each receiver controls the rate at which it receives the data, usually by using multiple multicast groups. The receivers join and leave groups depending on their path congestion state so the amount of data being received is always appropriate. This is commonly known by layered multicast [123, 60]. A multi-rate scheme is more suitable to audio and video streams and reliable data transfer by using proper coding techniques. One advantage is that receivers with different needs can be served at a rate closer to their needs rather than having to much the speed of the slowest receiver. The main drawback is that its flexibility is paid in terms of coding costs and overheads relied to join and leave actions. Other limitations of such schemes consist mainly in their slow convergence time and lack of inter-protocol fairness mainly to TCP which becomes critical when the delay to join/leave a group is high.

Multi-rate protocols schemes are addressed in more details in section 2.6 dedicated to the problem of heterogeneity in a multicast session.

2.5.6 Active networking based approaches

In opposition to an end-to-end congestion control, router-assisted schemes provide additional mechanisms to multicast congestion control. Even if in the current Internet, such schemes require changes to be done in routers and are not deployed yet, are very promising. The main contribution of active routers to multicast consists in the possibility to benefit from the physical multicast tree and thus to avoid the complexity inherent to building and maintaining a logical tree structure [27, 106]. More specifically, multicast protocols can benefit from special features such as feedback aggregation [64], hierarchical RTT estimation [55], cache of data [4] or the management of multiple groups in case of a layered multicast [133].

NCA (Nominee-based Congestion Avoidance) is the congestion control algorithm associated to AER (Active Error Recovery) [64]. In AER, repair servers co-located with active routers are used to cache data packets in order to insure locale recoveries when losses occur. NCA adopts a representative (nominee) strategy where the sender's congestion window is updated according to packet loss indications from this nominee. The nominee is the worst receiver in the multicast group chosen based on congestion status messages (CSM) sent periodically by the receivers. A CSM sent by a receiver to its upstream repair server contains its experienced loss rate and its RTT to the multicast source. The end-to-end loss rate p is estimated using a fixed size sliding window and the RTT is estimated using a hierarchical algorithm. The CSMs are aggregated at the intermediate repair servers. Every repair server determines the worst receiver according to a congestion metric ($RTT \times \sqrt{p}$) derived from the simple TCP equation (2.1). A repair server maintains the identity of the worst downstream receiver, as well as the most recent loss rate and RTT estimates for that receiver and unicasts this information periodically to its closest upstream repair server. Once the source has identified the worst receiver based on the same congestion measure, it unicasts a message to this receiver soliciting per-packet acknowledgments from it. Afterwards, it reduces its rate in response to loss indications from only this receiver, and ignores those from all other receivers. The source updates its congestion window in a manner similar to TCP New Reno [50].

RMANP (Reliable Multicast Active Network protocol) [21] is another congestion control that benefits from processing performed at the routers using active networks. Active routers in RMANP perform cache of data packets for potential retransmissions. Additionally, routers are able to detect congestion which makes the system reaction to congestion faster. When a congestion occurs, an active router retransmits lost data packets from its cache at a controlled rate and holds newly received ones in a queue for subsequent forwarding. When congestion decreases, an active router forwards data packets from its cache at an increasing output rate, until the cache is empty. An active router is able to detect congestion based on the status of its buffer used to store data packets which have not been absorbed by the outgoing interfaces. Performing sequence number control, the active routers and the receivers are also able to detect congestion based on their experienced losses. A congestion detected by an active router or a receiver, is indicated explicitly to the source using congestion indication (CI) messages, each of them contains a rate value computed according to the experienced losses. A CI is processed by all the active routers upstream the congestion location hence collaborating to congestion control tasks with the source in a distributed manner. CIs are filtered by active nodes based on the stamped rate avoiding multiple reaction to a single congestion instance as well as an implosion of CIs. Besides the congestion situations which are explicitly indicated, the source requests periodically feedback state information (FSI) from its direct active routers and those with receivers as direct descendants. A FSI contains information about the RTT of its originator to the source and its highest acked sequence number. The source will increase its output rate if the highest acked sequence number received increments the global one. A severe congestion is detected by the source or an active router if no FSI is observed. In such a situation, the output rate is reduced to its minimum specified value. A receiver drops itself if it considers that it can not receive data packet with the session minimum rate.

The active networks technology has also proved to be effective in a layered[2] multicast congestion control like ALMA (Active Layered Multicast Adaptation [133]). Active networks are used in ALMA to overcome the lack of mechanisms to obtain the required information about resource availability. One ALMA feature is its fast convergence due to its capsule-based congestion indication mechanism. Active network capsules are used for bandwidth probing, and layer pruning information. ALMA is based on an economic supply and demand model. Every downstream link exports a price as a function of its characteristics and load (the average queue length). Data capsules filter themselves by comparing price and budget and prune multicast branches that have persistent congestion. A receiver could send subscribe capsules to probe for additional bandwidth.

2.6 Dealing with Heterogeneity

As the Internet is moving to more and more heterogeneous networks, the presence in a multicast session of multiple members with different characteristics (in terms of available bandwidth, propagation delay, loss rate and background traffic) is a real challenge. Multicast protocols have to deal with the problem of heterogeneity in order to improve network utilization in addition to the overall *satisfaction*[3] of the different subscribers.

Earlier protocols accommodate heterogeneity by adapting the source rate to the worst receiver in the group, eventually associated to an ejection or a pruning mechanism [116] [26] to insure an acceptable QoS for the subscribed receivers. On one hand, transmitting with a rate that matches the most constrained receiver will slow down the overall communication throughput which results in an underutilization of the high-bandwidth receivers. On the other hand, sending with the maximum supported rate makes the weak receivers suffering from losses in addition to bothering the source[4] with frequent requests. As a result, an efficient mechanism has to be provided in order to accommodate heterogeneity in a multicast communication while preserving receivers satisfaction.

Improving the receivers satisfaction can be achieved by adopting a multi-rate mechanism where receivers with different needs can be served at a rate closer to their needs rather than having to match the speed of the slowest receiver. In a multi-rate session, the multicast source can transmit at different rates to different receivers either through a hierarchical scheme (layering) [60, 80, 123, 69] or a replicated scheme (destination set grouping, DSG) [62].

[2]see section 2.6

[3]also referred to as *utility* or *inter-receiver fairness* [61].

[4]and eventually other members such as the receivers or even network elements in reliable multicast protocols with local error recovery (section 2.4.1).

In a layered scheme[5], the source encodes its data into a number of layers that can be incrementally combined. A receiver has to subscribe to all the layers up to and including the highest one. In a replicated scheme, the source transmits multiple copies of the same data at different rates. A receiver needs only to subscribe to one group. Layered schemes provide more economical bandwidth usage than DSG schemes, however layering is more complicated and requires efficient hierarchical encoding/decoding algorithms and synchronization among different layers.

Other solutions to receivers heterogeneity in a multicast session is the use of proxies [23] where the multicast session is split into multicast data groups of co-located homogeneous participants. *Reliable Multicast Proxies* (RMX) organize these data groups into a spanning tree using an overlay network of TCP connections.

In what follows, some layered protocols are overviewed before dealing with the problem of receivers' partitioning and adaptive multi-rate multicast.

2.6.1 Layered Multicast

In layered multicast, the source transmits a hierarchically encoded scheme. Data is encoded into a number of layers that can be incrementally combined. Every layer is sent on a separate multicast address and the receivers join and leave groups depending on their path congestion state so the amount of data being received is always appropriate. Historically, Deering [31] is the first suggesting the IP multicast model to be used for layered transmission where each layer is individually mapped onto a multicast address.

Receiver-driven Layered Multicast (RLM) [60] is one of the first proposed layered multicast protocols. Motivated by the fact that a single-rate approach could never meet the requirements of heterogeneous receivers, authors of [60] propose to benefit from layered encoding where receivers adapt their reception rate by joining/leaving layers each of which is transmitted using a separate multicast address. More precisely, a receiver probes for extra bandwidth by subscribing to the next higher layer (referred to as *join-experiments*) at "well chosen" times. If a join-experiment causes congestion, the receiver will drop from the previously subscribed layer.

RLM proposes a learning algorithm where each receiver determines the level of subscription that causes congestion. This learning algorithm is implemented through the use of *join-timers* for each level of subscription associated to an exponential back-off for critical layers. To be more scalable, the join-experiments rate is set inversely proportional to the group size. In such a way, the amount of join messages is kept independent of the session size. However, join-experiments have to be synchronized. The absence of coordination between the receivers could result in some indesirable situations. For instance, a receiver that performs a join-experiment could cause a congestion situation making the other receivers located behind the same bottleneck (believing that the congestion has been experienced due to their subscription to a higher layer) leave a layer. Consequently, a *shared learning* is necessary.

[5]see [70] for a comparison between the layered and replicated approaches.

A receiver willing to subscribe to a given layer, notifies the whole group by multicasting a special message containing the concerned layer.

A more scalable solution than the shared learning of RLM, is proposed by Vicisano, Rizzo and Crowcroft in *Receiver-driven Layered Congestion Control* (RLC) [123], another layered multicast protocol. As opposed to RLM, RLC adopts an implicit coordination using *synchronization points* (SPs). A receiver is not allowed to perform join-experiments unless a synchronization packet from the source is received. SPs at each level are always a subset of the SPs at the previous level. Consequently, when a receiver does a join attempt, all receivers at the same or lower layers will do an attempt as well.

Even if all the receivers dropped the layer that causes the congestion, they would continue dropping subsequent layers due to the long time, a leave action could take. This is avoided by introducing *deaf periods* during which a receiver does not react further to congestion signals (after a layer is already being dropped). A deaf period is set slightly higher than a leave action takes effect.

RLC also addresses friendliness towards TCP flows. The join and leave actions are performed in such a way to emulate the TCP behavior (Additive Increase Multiplicative Decrease [24]). Moreover, a sender-initiated probing is performed by inserting periodic short bursts in order to emulate the behavior of TCP. Congestion signals that result from those bursts are used by the receivers to avoid subscribing to a higher layer rather than a real congestion.

Nevertheless RLC is only fair with TCP connections whose RTTs are close to 1 second and not in general. In fact RLC is aggressive with TCP connections with $RTT << 1$ and gets much less bandwidth in the presence of a TCP flow with $RTT >> 1$. Furthermore it insures intra-fairness for RLC sessions of only equal SPs spacing.

Authors of [15] proposed *Fair Layered Increase/Decrease with Dynamic Layering* (FLID-DL) that enhances RLC through the use of Dynamic Layering in order to deal with the problem of large IGMP leave latencies. In Dynamic Layering, the rate of each layer decreases over the time. A receiver can reduce its reception rate simply by not joining any additional layers; thus avoiding leave-related overheads. A reception rate is maintained by joining layers and is increased by joining additional layers beyond those needed to maintain a constant rate.

Coding-Independent Fair Layered Multicast (CIFL) [69] is a layered multicast protocol that attempts to achieve better fairness with TCP. For this purpose, each receiver computes a target rate using the TCP formula. Afterwards, it determines and joins the appropriate number of layers so the obtained rate is the closest to the target one. In order to make the receivers know the rates of the different layers, the source includes these rates into the SPs. SPs are like in RLC, used to allow coordination among the receivers; but unlike RLC, SPs are present only in the base layer. Finally CIFL avoids useless join attempts by learning from unsuccessful previous attempts experienced in same or better network conditions.

At this stage, it is worth mentioning that a layered approach can also be used for bulk data transfer using appropriate data encoding. Authors of [10] proposed bulk data encoding in a cumulative layered approach thus avoiding data duplication. Many other layered multicast congestion control schemes have been proposed. For example, the *packet Pair receiver-driven Layered Multicast* scheme (PLM) [73]. Several other proposals can be found in [78] that contains an overview of multi-rate video multicast. Existing approaches are classified into stream replication, cumulative layering and non-cumulative layering. Some trade-offs such as bandwidth economy, adaptation granularity and coding complexity, are studied.

2.6.2 Adaptive Multi-rate Multicast

Layered multicast has proven to be an effective way to accommodate heterogeneity. Layered approaches overviewed in the previous subsection (referred hereafter as *traditional* layered multicast) only provides a coarse-grained control (at the granularity of a layer) while available bandwidth in the Internet changes rapidly over short time intervals. This is why a more fine-grained control over bandwidth has to be considered. Furthermore in traditional layered multicast, throughputs assigned to each layer are predefined independently of the multicast group population. A multi-rate scheme has to set its transmission rates to the capacity of the subscribed receivers.

We will designate by *adaptive multi-rate multicast* every multi-rate scheme (layered or replicated) where the transmission rate of the different layers are set according to the capacity of the different receivers. To this class of multi-rate multicast can also be included, those that perform a fine-grained control within each layer such as the one proposed in [72] (an extension of TFMCC [131]) and WEBRC *Wave and Equation Based Rate Control* [83]. In this latter, data are transmitted with waves which have the advantage of overcoming the problem of large IGMP leave latency, allowing for fine-grained control over the receiver reception rate. Another work [17] demonstrates that fine grained adaptation can also be achieved by a Fibonacci allocation scheme. Schemes like SAMM (*Source-Adaptive Multi-layered Multicast*) [124] investigate a model where the server dynamically adjusts the number and rate of each subscription layer in response to congestion feedback which are aggregated in routers or other network elements. Other multi-rate multicast protocols which are an enhancement of traditional layered multicast protocols where a fine-grained control is addressed include works in [105] and [12].

Most of the adaptive multi-rate multicast (as defined above) require performing a partition of the receivers depending on their capacities so an overall satisfaction measure is maximized. To do so, we first need to be able to estimate the receivers' capacity and then to provide an algorithm to partition optimally the receivers into subgroups of similar capacities.

Determining the capacity of a receiver is not obvious and is not easily obtained in the current Internet. Based on the fact that a receiver's capacity is inversely proportional to its RTT, [53] proposed to use RTT measures to estimate the capacity of the receivers. Jiang et al. [63] relied on the losses experienced by the receivers to estimate their capacities while Bonmariage and Leduc [13] combined both of the RTTs and the loss rates through applying the TCP formula for this purpose.

Once the receivers' requirements are known, the next step is to split them into subgroups where the transmission rate of each subgroup (or layer) is set accordingly. The receivers' partitioning problem have been addressed in the context of a replicated scheme in [62] where a number of grouping heuristics to make splitting decisions have been proposed. In the context of bulk data transfer, [10] considered the problem of finding the optimal rate at each layer so the completion time of a fixed-size file is minimized.

The authors in [135] considered the optimal partitioning of receivers into groups for multi-rate schemes. They formulated, for a general class of utility function, the partitioning problem as an optimization problem to maximize the sum of receiver utilities. They presented a dynamic programming algorithm to solve the partitioning problem, and proved that the solution it finds is optimal. They considered in their work both fully reliable and fault-tolerant multicast.

Gau et al. [53] proposed a similar partitioning algorithm to the one previously proposed in [135]. Additionally, they proposed to periodically perform the partitioning algorithm in order to adapt to time-varying capacities of the receivers. They also analyzed the impact of the receivers' capacity distribution on the improvement that results from receivers partition. They for instance, found that for a uniform distribution with any parameters (meaning that independently of the heterogeneity degree), only a two-subgroups partition is sufficient to achieve at least 50% of the maximum utility value.

As have been mentioned above, scalability is a main concern for any multicast protocol, particularly when addressing the problem of heterogeneous receivers. In fact, the source will be rapidly overwhelmed if it has to estimate the capacity of the different receivers in addition to performing the partition algorithm. Authors of [13] proposed a probabilistic approach to address the problem of scalability. A receiver sends a request to the source with a probability p so that only a limited number $R = pG$ of requests would be processed by the source. G is the number of the receivers that belong to the multicast group. This probabilistic approach unloads the source from processing a huge amount of feedback in addition to reducing the partitioning algorithm complexity.

Schemes like the ones proposed in [3] and [53] suggest to use a hierarchical structure in order to be more scalable. In [53], the receivers are organized in a hierarchy of zones. Every zone has a group manager which is responsible for performing the partitioning algorithm of its zone receivers. This partitioning is performed based on feedback periodically sent by the receivers to their zone manager. In [3], the multicast tree is used as it is where intermediate nodes (routers) are those who perform the partition in addition to merging functionalities.

Due to the aggregation performed at each level of the hierarchy, some information about downstream receivers capacity are lost. Consequently the final partition performed by the source is not necessarily the optimal one. This problem has been addressed in [3] by introducing virtual layers at each merging level. However their partition algorithm has some complexity especially with the introduction of virtual layers. They proposed to compute a satisfaction metric for each candidate solution instead of adopting a dynamic algorithm like those proposed in [135] [53] [13] which have less computation complexity.

2.7 Standardization Effort

Many multicast-related topics have been considered in the standardization effort of the Internet Engineering Task Force (IETF). A number of working groups have been formed and assigned specific tasks. Most of them are concerned with network-level problems of multicast such as group membership management and routing protocols. Multicast security has also been considered in a dedicated working group. This is also the case of reliable multicast which is considered within the Reliable Multicast Transport (rmt) working group.

Reliable multicast standardization is challenging due to a couple of reasons. First, there is no single reliable multicast protocol that fits all applications due to the wide variety of application requirements. Secondly, fairness with other multicast and unicast sessions especially with TCP is essential. Aggressive multicast traffic is considered as a real threat to the current Internet operating mode. As a result, the rmt working group has specified some criteria for evaluating reliable multicast protocols (RFC-2357 [88]) in addition to some guidelines to be followed when designing reliable multicast standards [56].

Due to the different requirements of multicast applications, a number of reliable multicast protocols are standardized within the IETF rather than a single protocol. The rmt group has developed a building block approach (RFC-3048 [129]) which is based partly on abstracting components from existing reliable multicast proposals. These building blocks will be used to generate different instantiations of reliable multicast protocols. The RFC-3269 [68] provides guidelines to be respected by authors of drafts when specifying building blocks and instantiating protocols. The main building blocks developed are *NACK-Oriented Reliable Multicast (NORM)* [1], *TRee ACKnowledgement based protocol (TRACK)* [130], *Layered Coding Transport (LCT)* [82], *Layered Congestion Control* (LCC) [86], and *FEC* [85] building blocks. Examples of instantiating these building blocks include *NORM* [2], a single-rate receiver reliable (NACK-based) multicast protocol and Asynchronous Layered Coding (ALC) protocol [81], a multi-rate receiver reliable protocol that uses the LCT, LCC and FEC building blocks. [112] contains an implementation of a multicast file transfer tool on top of ALC.

PGM reliable multicast protocol [118] has a commercial support from vendors such as Cisco and Microsoft. As a result, it has been granted the "EXPERIMENTAL RFC status" by the IETF. Another router-assisted approach for reliable multicast protocols called Generic Router Assist (GRA) [19] is proposed as a new building block. More recently another file delivery protocol particularly suited to multicast networks, called FLUTE (*File Delivery over Unidirectional Transport*) [100] that builds on top of ALC has also been proposed in an Internet draft.

2.8 Conclusion

Multicast provides new opportunities to multiple Internet applications. This chapter is an attempt to overview the most significant proposals mainly at the transport level with some attention to the network level problems. We mainly focused on the IP Multicast architecture without considering other alternatives which are, indeed, very important for the near future since IP multicast is facing impediments to be deployed in the Internet (see [39] for deployment issues and [41] on alternative group communication services). A survey on multipoint communication can be found in [38] and [111]. In this work, we were not concerned with security problems in multicast. The interested reader can refer to [71] for a survey.

At the transport level, we consider three major problems: reliability, congestion control and heterogeneity support. We of course, were not able to consider all the proposed solutions to these problems. However, the most significant ones are dealt with. For a survey of multicast transport, the reader can refer to [37] and [98]. A survey on reliable multicast congestion control algorithms is available in [136], and [78] contains an overview of multi-rate video multicast for receivers heterogeneity support.

Chapter 3

An Analysis of Router-assisted Reliable Multicast Protocols

3.1 Introduction

Desirable features of reliable multicast include low end-to-end delays, high throughput and scalability. Earlier reliable multicast protocols adopted an end-to-end solution to perform loss recovery. However these schemes do not provide solutions to all the loss recovery problems. This is mainly due to the lack of topology information at the end hosts. Recently, the use of active network concepts [120, 127] where routers themselves could contribute to enhance the network services by customized functionalities, have been proposed in the multicast research community. Contributing mainly on feedback implosion problem, retransmission scoping and cache of data, active reliable multicast protocols open new perspectives to achieve high throughput and low latency on wide-area networks.

ARM (Active Reliable Multicast) [74] and AER (Active Error Recovery) [64] are two protocols that use active services. In both protocols, the main contribution of active services is the cache of data packets to implement local recovery. However, they differ in the strategy adopted for solving the NACK implosion problem. ARM adopts a *global* suppression strategy where a receiver experiencing a packet loss sends immediately a NACK to the source. Active services in the routers consist then in the aggregation of the multiple NACKs. In contrast, AER uses a *local* suppression strategy inspired from 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. If it receives a NACK for the same packet, it will cancel the timer, suppress its NACK and behave exactly as if it has sent the NACK itself. In this case, the active routers have in charge to indicate to the receivers that one of them has issued a NACK for a given packet. The implementation of this service is done by multicasting the NACK packet to the other downstream receivers. Finally, an active router in ARM sends 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.

Cache of data at the routers permits the local recovery of loss packets. Kasera et al. [65] have analytically shown that protocols with local recovery exhibit better performances. They compared 3 different strategies, noted L1, L2 and L3. L1 is called *server-based* and benefits from the cache of packets from *repair-servers* co-located with traditional routers at the edge of the core network. L2 is a generic version of SRM and L3 uses a logical tree structure for the recovery. They use an analytical framework originally proposed by Pingali et al. [103]. This framework was also adopted in the literature to perform other analysis of reliable multicast protocols [75, 97, 66]. In [66], the authors have analyzed three protocols, noted P1, P2 and P3 that benefit from subcast facilities by using multiple multicast groups or channels. Whereas P1 benefits only from the subcast facility, P2 and P3 also perform the NACK suppression. However the local recovery and the global NACK suppression were not investigated. Authors of [9] have quantified the performances of recovery strategies with active services. Their analysis is based on the recovery latency and has shown that active approaches always perform better provided that a minimum storage capacity is available in the active routers.

None of these works have analytically investigated the three active services together: local recovery, the NACK suppression strategies and the subcast. In this chapter, a throughput analysis is performed on generic reliable multicast protocols that benefit from active services at the routers. The motivations behind that are to make possible the comparison between various mechanisms in active reliable multicast protocols and to better understand the contribution of active networking services. Our main contributions are in comparing various active strategies and identifying those that exhibit the best performances. In an attempt to help for active networks dimensioning, we also address the constraints put on active routers processing power for achieving better performances than traditional non active approaches. The rest of the chapter is organized as follows. Section 3.2 presents the network model and the general assumptions associated to the study. Section 3.3 provides a description of the generic reliable multicast protocols to be analyzed. The throughput analysis is presented in section 3.4. Section 3.5 presents the numerical results before concluding.

3.2 Network Model and Hypothesis

The network model used in the study is similar to the one used in [65]. One source multicasts data packets to R receivers through a packet network composed of a fast core network and several slower edge access networks. We designate by the *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 3.1). In our network model, we only consider active routers at the edge of the core network. This is for two reasons:

- The core network is reliable. It has been shown in [132] that the losses occur mainly at the edges of the core network, i.e. at the source link and the tail links with respect to our network model.
- The core network is a very high-speed network. Adding complex processing functions inside the core network will slow down the packet forwarding functions.

Our network model differs from the one used in [65] as F routers among the N possible ones are active, $0 \leq F \leq N$ (figure 3.1). This will be useful to evaluate the impact of the density of active routers. Particular cases where $F = 0$ and $F = N$ will be underlined in the study. Downstream each active router A_i, there are B receivers $R_{i1}, \cdots, R_{iB}$ which form a

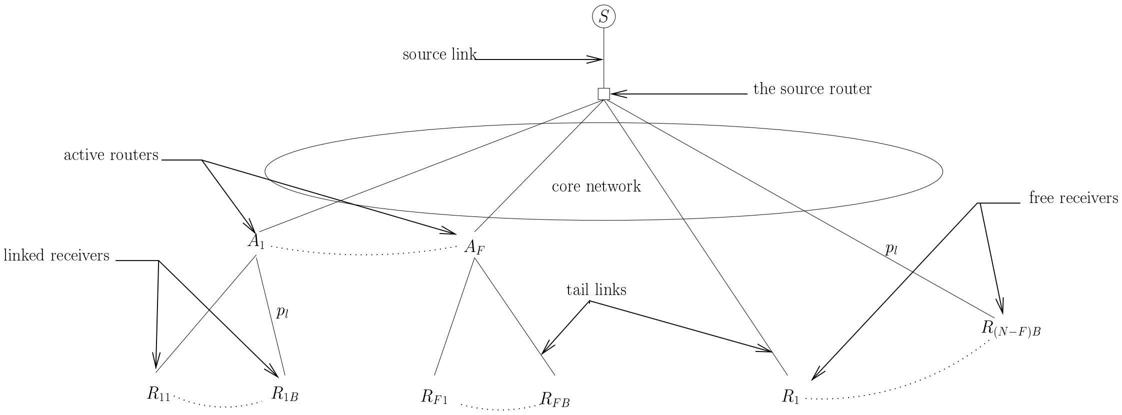


Figure 3.1: Network Model.

local group. In this study, the receivers associated with the F active routers are said *linked* and the other receivers are said *free*.

For the loss model, we will consider that the core network is reliable, as mentioned previously. The other links (the source and tail links) are supposed to have a loss probability of p_l which results in an end-to-end loss rate (perceived by a receiver) of $p = 1 - (1 - p_l)^2$. The losses are assumed to be temporally independent and those at the tail links are assumed to be mutually independent. We will also assume that NACK packets are never lost.

For this analysis, we will consider that active routers are always able to perform local recovery. This is motivated by the following observation: the mean holding time of a data packet in an active router is bounded as receivers will correctly receive the packet in a finite time for reasonable loss probabilities. Therefore the probability that the source eventually performs the retransmission is low even with a small size of cache memory within routers. As it has been shown in [65], about a hundred KBytes is sufficient per multicast session so that the probability of a local recovery failure from the repair servers is negligible.

A NACK packet is assumed to follow 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 where every node in the multicast tree is aware of its parent node in the multicast tree. Symmetric routing protocols such as MOSPF [93] and HBH [28] allow for an easy implementation of this desirable behavior.

3.3 Generic Reliable Multicast Protocols

In this section, we describe five generic reliable multicast protocols noted S_1, S_2, S_2^s, S_3 and S_3^s. All of them benefit from the cache of packets at the active routers and are NACK-based. After the emission of a request, a receiver initializes a timer and the request is renewed when the timer goes off while the corresponding repair is not received yet. In order to limit the processing overheads of duplicate NACKs, the source and the active routers discard all similar NACKs for a given amount of time. For the analysis, we will assume that the "duplicate discard" period is well chosen. Therefore, no duplicate NACKs will trigger a new retransmission. These protocols differ from each other in the strategy used for the NACK suppression and whether subcast is used or not.

How to implement the subcast service is beyond the scope of this analysis. We will assume at this stage, that the active routers implement subcast by storing necessary information based on the NACKs they receive [66, 74, 87]. It has been shown in [66] that active routers need only a few hundreds of bytes per multicast session to support the subcast functionality.

3.3.1 Description of Protocol S_1

S_1 uses the global suppression of NACK packets. A receiver experiencing a packet loss sends immediately a NACK to the source. The active routers are responsible for the suppression of NACK packets so only one NACK per loss is forwarded to the source. S_1 has the following properties:

- The source multicasts data packets to the multicast address subscribed to by all the receivers.
- Upon the reception of a data packet, an active router stores the packet in its cache and forwards it downstream in the multicast tree.
- On the detection of a packet loss, a receiver sends immediately a NACK towards the source.
- Upon the reception of a NACK packet, an active router sends the corresponding repair packet (if available) to all the receivers composing its local group. Otherwise, it sends the NACK to the source.
- The reception of a NACK packet by the source triggers the retransmission by multicast of the corresponding packet to all the receivers subscribed to the multicast address.

3.3.2 Description of Protocols S_2 and S_2^s

As opposed to S_1, S_2 adopts a local suppression of NACKs. A receiver waits for a random amount of time prior to sending a NACK to the source. The goal is to generate only one NACK per group of receivers in the multicast tree. S_2 is similar to L1 described in [65] (which is itself a generic version of AER [64]) since both of them adopt the NACK local suppression. Whereas S_2 performs this suppression only within the end receivers linked to the same active router, L1 also performs local suppression among parent children in the multicast tree. It is worth noting that the local suppression of NACKs at the free receivers is not efficient due to the absence of upstream active routers. In this case, the free receivers would behave as an SRM receiver if we decide to multicast the NACK to the whole group (with or without scoping). However in this analysis, we assume that the free receivers send their requests directly to the source via the source active router. This latter forwards the NACKs it receives without any suppression to the source. This router is only responsible for implementing the subcast service (the subcast from the source in S_2^s and S_3^s). Like P2 and P3 [66], S_2 performs NACK suppression based on timers [52]. Whereas P2 and P3 do not perform any local recovery and adopt a strategy based on multiple group channels, S_2 benefits from the contribution of active routers [64] as will be described below:

- The source multicasts data packets to the multicast address subscribed to by all the receivers.
- Upon the reception of a data packet, an active router stores the packet in its cache and forwards it downstream in the multicast tree.
- On the detection of a packet loss, a receiver waits for a random amount of time. After which it would send a NACK packet to the source.
- When a receiver is waiting for sending a NACK, if it receives a similar one from its active router, it will cancel its request and behave as it has sent the NACK itself. If a repair packet is received while a receiver is waiting for it, then the receiver will quit definitely the recovery process of the corresponding packet.
- Upon the reception of a NACK from downstream, an active router multicasts the repair packet from its cache to its local group. Otherwise, it multicasts the NACK to both the source and its local group, except the NACK incoming link. The aim of that is to perform the NACK local suppression.
- The reception of a NACK packet by the source results in multicasting the repair packet to all the receivers of the group.

S_2^s is identical to S_2 in performing a local NACK suppression among receivers of the same local group. However, it benefits from the subcast facility from the source. It is difficult for active routers that already perform the local NACK suppression to also implement a subcast service. In fact, they are unable to know the identity of the receivers that have experienced a loss since they do not receive all the generated NACKs in their local group. According to the proposed network model, receivers linked to an active router that performs local NACK suppressions do not benefit from the subcast feature whereas free receivers will benefit from the subcast directly from the source in S_2^s.

3.3.3 Description of Protocols S_3 and S_3^s

Active routers that perform a global NACK suppression strategy as in S_1 can easily implement the subcast service. This is because these active routers are able to know the identity of the receivers that experience a loss since they receive all the corresponding NACKs. Therefore we define here the protocol S_3 which is similar to S_1 in performing a global NACK suppression strategy but also implements the subcast service within active routers. We also define the protocol S_3^s that behaves as S_3 but also benefits from the subcast facility from the source. It is worth mentioning that S_3 and S_3^s are generic versions of ARM [74].

3.4 Throughput Analysis

3.4.1 Analysis Approach and Notation

In this subsection, we explain the approach taken to perform our analysis in addition to the introduction of the different notations used in the study. For reader convenience, all the notations are listed in table 3.1. Table 3.2 summarizes the probability distribution and the mean of each random variable used.

Similar to the approach taken by Pingali et al. in [103], our analysis uses the processing requirements of the protocols rather than the bandwidth requirements. The maximum throughput achievable by a protocol depends on the processing time per packet at the various nodes in the multicast tree. The overall throughput Λ^ω achieved by every protocol $\omega \in \{S_1, S_2, S_3, S_2^s, S_3^s\}$ can be seen as the minimum throughput achieved by the different nodes present in the network model, that is:

$$\Lambda^\omega = Min(\Lambda_x^\omega)$$

where Λ_x^ω is the throughput achieved by node $x \in \{X, Y, Y_*, A_1, \cdots, A_F\}$ under protocol ω. x can be the source node (X), a linked receiver (Y), a free receiver (Y_*) or an active router (A_i, $i = 1 \dots F$). The throughput Λ_x^ω achieved by node x is inversely proportional to the mean processing time per packet $E[x^\omega]$ at node x under the protocol ω:

$$\Lambda_x^\omega = 1/E[x^\omega]$$

In what follows, we designate by X_p and X_n the required time to process a data packet emission and a NACK reception respectively. The required time to receive a data packet is noted Y_p while Y_n is used to designate the processing time to send a NACK packet. At the active routers, we use X_p^a and X_n^a to designate the processing time for sending a data packet and to receive a NACK packet respectively. Similarly, Y_p^a and Y_n^a are respectively the processing time for receiving a data packet and to send a NACK packet at an active router.

The total processing time per packet at the source X^{S_j} under protocol S_j ($j = 1 \dots 3$) includes the initial transmission time, all the retransmissions and NACK overheads necessary to achieve a correct reception of the packet by all the receivers. Let M_S be a random variable that represents the required number of the transmissions required from the source until all its downstream nodes (i.e. the F active routers and the $(N-F)B$ free receivers) have correctly received a randomly chosen packet. Hence, the probability distribution of M_S can be expressed as follows:

$$P[M_S \le m] = \prod_{i=1}^{F} P[M_a \le m] \prod_{j=1}^{(N-F)B} P[M_r \le m] \tag{3.1}$$

where M_a is the required number of transmissions required from the source until an active router has correctly received a randomly chosen packet. We have:

$$\begin{aligned} P[M_a \le m] = & \quad 1 - P[all\ the\ first\ m\ packet\ transmissions\ are\ lost] \\ = & \quad 1 - p_l^m \end{aligned} \tag{3.2}$$

Using the next formula to compute the mean $E[M]$ of a random variable M:

$$E[M] = 1 + \sum_{m=1}^{\infty} 1 - P[M \le m] \tag{3.3}$$

X^{S_j}	the total processing time per packet at the source for protocol S_j, $j = 1..3$.
Y^{S_j}	the total processing time per packet at a linked receiver for protocol S_j, $j = 1..3$.
$Y_*^{S_j}, Y_*^{S_k^s}$	the total processing time per packet at a free receiver under protocol S_j and S_k^s respectively, $j = 1..3$ and $k = 2, 3$.
$A_i^{S_j}, A_i^{S_k^s}$	the total processing time per packet at an active router A_i ($i = 1..F$) under protocol S_j and S_k^s protocols respectively, $j = 1..3$ and $k = 2, 3$.
Λ_x^ω	the mean throughput achieved by node x under protocol ω.
Λ^ω	the mean throughput achieved by the ω protocol.
X_p, X_n	the processing time for sending a data packet and to receive a NACK packet respectively.
Y_p, Y_n	the processing time for receiving a data packet and to send a NACK packet respectively.
Y_t	the amount of time required to process a timeout
X_p^a, X_n^a	the time required by an active router to send a data packet and to receive a NACK packet respectively.
Y_p^a, Y_n^a	the required time by an active router to receive a data packet and to send a NACK packet respectively.
M_S	the number of a packet transmissions by the source until all the active routers and the free receivers have correctly received it.
$M_R,\ M_A$	same as M_S when $F = 0$, respectively $F = N$.
$M_b,\ M_B$	the number of packet transmissions by an active router until one, respectively all the associated linked receivers have correctly received the packet.
M_a	the number of packet transmissions required at the source until an active router has correctly received it.
M_r	the number of packet transmissions at the source until a receiver (linked or free) has correctly received it.

Table 3.1: Notation used in the throughput analysis

we get:

$$E[M_a] \quad = \quad 1 + \sum_{m=1}^{\infty} p_l^m = \sum_{m=0}^{\infty} p_l^m = \frac{1}{1-p_l}$$

On the other hand M_r is the number of transmissions required from the source until a receiver (linked or free) has correctly received the packet. Similarly to (3.2), we have:

$$P[M_r \leq m] = 1 - p^m \tag{3.4}$$

and for the mean:

$$E[M_r] \quad = \quad 1 + \sum_{m=1}^{\infty} 1 - p^m = \frac{1}{1-p}$$

Replacing the probability distributions of M_r (3.4) and M_a (3.2) in equation (3.1), we get:

$$P[M_S \leq m] = (1 - p_l^m)^F \ (1 - p^m)^{(N-F)B} \tag{3.5}$$

We also introduce M_R (respectively M_A) which is the same as M_S when none (respectively all) of the routers are active. Setting F to respectively 0 and R in equation (3.5):

$$P[M_R \leq m] = (1 - p^m)^R$$

and

$$P[M_A \leq m] = (1 - p_l^m)^R$$

Therefore:

$$E[M_R] = 1 + \sum_{m=1}^{\infty} 1 - (1 - p^m)^R$$

and

$$E[M_A] = 1 + \sum_{m=1}^{\infty} 1 - (1 - p_l^m)^N$$

For a receiver, the total processing time per packet includes all the processing overheads until a randomly chosen packet is correctly received. This includes the overheads for receiving the initial packet, all the necessary repair packets, the overheads for sending and receiving the NACK packets and the overheads due to timeouts. We use Y^{S_j} and $Y_*^{S_j}$ to designate the processing requirements per packet at a linked and a free receiver respectively. Additionally, we introduce $Y_*^{S_k^s}$ $(k = 1, 2)$, the required processing time at a free receiver under protocols which benefit from the subcast from the source[1]. Since there are timeouts to be processed at the receivers, we introduce Y_t to designate the required time to process one timeout.

[1]We introduce this notation just for the free receivers since the subcast service from the source has no effect on the linked receivers. Even if an active router could receive an unwanted retransmission, this latter is never forwarded downstream.

Random var. X	Prob. distribution $P[X \leq m]$	Mean $E[X]$
$M_b = M_a$	$1 - p_l^m$	$1/(1 - p_l)$
M_r	$1 - p^m$	$1/(1-p)$ and $E[(M_r - 2)^+] = p^2/(1-p)$
M_B	$(1 - {p_l}^m)^B$	$1 + \sum_{m=1}^{\infty} 1 - (1 - {p_l}^m)^B$
M_S	$(1 - p_l^m)^F \ (1 - p^m)^{(N-F)B}$	$1 + \sum_{m=1}^{\infty} 1 - P[M_S \leq m]$
M_R	$(1 - p^m)^R$	$1 + \sum_{m=1}^{\infty} 1 - (1 - p^m)^R$
M_A	$(1 - p_l^m)^N$	$1 + \sum_{m=1}^{\infty} 1 - (1 - {p_l}^m)^N$

Table 3.2: Useful probability distributions.

For an active router, the total processing time per packet includes the overheads for sending/receiving and processing the data packet and the corresponding NACKs in case of losses. We use $A_i^{S_j}$ and $A_i^{S_k^s}$ to note the processing requirements per packet at the active router A_i under protocol S_j $(j = 1 \ldots 3)$ and S_k^s $(k = 1, 2)$ respectively. For an active router, we need to estimate the required number of packet transmissions required so one of (all the) linked receivers has (have) correctly received this packet. We use M_b (respectively M_B) to designate the number of packet transmissions from an active router until one (respectively all the B) downstream receiver(s) has (have) correctly received the data packet. We have:

$$E[M_b] \quad = \quad E[M_a] = \frac{1}{1 - p_l}$$

and for M_B:

$$P[M_B \leq m] \quad = \quad \prod_{i=1}^{B} P[M_b \leq m] = (1 - p_l^m)^B.$$

giving:

$$E[M_B] = 1 + \sum_{m=1}^{\infty} 1 - (1 - p_l^m)^B$$

3.4.2 Analysis of Protocol S_1

We begin by analyzing the processing requirements in protocol S_1 at the source. A data packet is sent M_S times until all the active routers and free receivers have correctly received it. The source receives for this packet F $(M_a - 1)$ NACKs from the F active routers and $(N - F)B$ $(M_r - 1)$ NACKs from the $(N - F)B$ free receivers. The processing time per packet at the source can therefore be written:

$$E[X^{S_1}] = E[M_S]\ E[X_p] + (F(E[M_a]-1) + (N-F)B\ (E[M_r]-1))\ E[X_n] \tag{3.6}$$

Replacing $E[M_a]$ and $E[M_r]$ by their expressions, we get:

$$E[X^{S_1}] = E[M_S]\ E[X_p] + \left(F\ \frac{p_l}{1-p_l} + (N-F)B\ \frac{p}{1-p}\right) E[X_n] \tag{3.7}$$

Note that the protocol S_1 with no active router ($F = 0$) is identical to protocol N_1 described in [103]. By setting $F = 0$ in equation (3.7) we obtain a similar result as equation (11) stated in [103]:

$$E[X^{S_1}] = E[M_R]\ E[X_p] + R\ \frac{p}{1-p}\ E[X_n] \tag{3.8}$$

When all the routers are active, we find:

$$E[X^{S_1}] = E[M_A]\ E[X_p] + N\ \frac{p_l}{1-p_l}\ E[X_n] \tag{3.9}$$

The processing time at a receiver depends on whether it is linked or free. In S_1, a linked receiver receives a data packet M_B times with probability $(1-p_l)$. It sends back $(M_r - 1)$ NACKs to the source for which it processes $(M_r - 2)^+$ timeouts. We have:

$$\begin{aligned} P[M_r - 1 = m] &= P[exactly\ the\ first\ m\ transmissions\ are\ lost] \\ &= p^m(1-p) \end{aligned} \tag{3.10}$$

thus:

$$\begin{aligned} E[M_r - 1] &= \sum_{m=1}^{\infty} m\ P[M_r - 1 = m] \\ &= (1-p)\sum_{m=1}^{\infty} m\ p^m \\ &= (1-p)\ p\sum_{m=1}^{\infty} m\ p^{m-1} \\ &= (1-p)\ p\frac{1}{(1-p)^2} = \frac{p}{1-p} \end{aligned}$$

and for $M_r - 2$, we have:

$$\begin{aligned} P[M_r - 2 = m] &= P[exactly\ the\ first\ (m+1)\ transmissions\ are\ lost] \\ &= p^{m+1}\ (1-p) \end{aligned} \tag{3.11}$$

Hence,

$$
\begin{aligned}
E[M_r - 2] &= \sum_{m=1}^{\infty} m \; P[M_r - 2 = m] \\
&= (1-p)\; p^2 \sum_{m=1}^{\infty} m \; p^{m-1} \\
&= (1-p)\; p^2 \; \frac{1}{(1-p)^2} = \frac{p^2}{1-p}
\end{aligned}
$$

Finally, we get the following expression for the processing cost at a linked receiver:

$$E[Y^{S_1}] = (1-p_l)\; E[M_B]\; E[Y_p] \; + \; \frac{p}{1-p}\; E[Y_n] + \frac{p^2}{1-p}\; E[Y_t] \tag{3.12}$$

A free receiver receives the data packet M_S times with probability $(1-p)$. It sends (M_r-1) NACKs and processes $(M_r-2)^+$ timeouts, giving the following equation:

$$E[Y_*^{S_1}] = (1-p)\; E[M_S]\; E[Y_p] \; + \; \frac{p}{1-p}\; E[Y_n] + \frac{p^2}{1-p}\; E[Y_t] \tag{3.13}$$

If $F = 0$, meaning that all the receivers are free, we have an expression similar to that of protocol N_1 evaluated in [103]:

$$E[Y_*^{S_1}] = (1-p)\; E[M_R]\; E[Y_p] \; + \; \frac{p}{1-p}\; E[Y_n] + \frac{p^2}{1-p}\; E[Y_t] \tag{3.14}$$

For an active router A_i, a data packet is received M_S times with probability $(1-p_l)$. It sends the packet M_B times to its B linked receivers until they correctly receive the packet. This is done as a response to $B\;(E[M_r]-1)$ NACKs that it receives from its B linked receivers. Additionally, an active router sends back to the source (M_b-1) NACKs. These NACKs have been issued by the linked receivers due to a packet loss at the source link, that is:

$$
\begin{aligned}
E[A_i^{S_1}] = \;& (1-p_l)\; E[M_S]\; E[Y_p^a] \; + E[M_B]\; E[X_p^a] \\
& + B\; \frac{p}{1-p}\; E[X_n^a] + \frac{p_l}{1-p_l}\; E[Y_n^a]
\end{aligned} \tag{3.15}
$$

If all the routers are active, we find:

$$
\begin{aligned}
E[A_i^{S_1}] = \;& (1-p_l)\; E[M_A]\; E[Y_p^a] \; + E[M_B]\; E[X_p^a] \\
& + B\; \frac{p}{1-p}\; E[X_n^a] + \frac{p_l}{1-p_l}\; E[Y_n^a]
\end{aligned} \tag{3.16}
$$

This includes duplicate repairs due to losses experienced at the other linked receivers located downstream the same active router.

3.4.3 Analysis of Protocols S_2 and S_2^s

For the analysis of protocols S_2 and S_2^s, we will assume that the random waiting times at the receivers are well chosen in order to generate only one NACK per local group.

In protocol S_2, the source and the free receivers behave like in S_1; consequently, we have respectively for them:

$$E[X^{S_2}] = \quad E[M_S]\ E[X_p] + \left(F\ \frac{p_l}{1-p_l} + (N-F)B\ \frac{p}{1-p}\right) E[X_n] \tag{3.17}$$

and

$$E[Y_*^{S_2}] = (1-p)\ E[M_S]\ E[Y_p]\ +\ \frac{p}{1-p}\ E[Y_n] + \frac{p^2}{1-p}\ E[Y_t] \tag{3.18}$$

For a linked receiver, we have:

$$\begin{aligned} E[Y^{S_2}] = \quad & (1-p_l)\ E[M_B]\ E[Y_p]\ + \\ & \frac{E[M_B]-1}{B}\ (E[Y_n] +\ E[X_n])\ +\ \frac{p^2}{1-p}\ E[Y_t] \end{aligned} \tag{3.19}$$

The first term indicates how many times a linked receiver receives a data packet. The second and the third terms correspond respectively to NACKs that were sent and received by a linked receiver in the process of the NACK local suppression. The last term represents the processing overhead for the $(M_r-2)^+$ timeouts.

An active router receives a data packet from the source M_S times with probability $(1-p_l)$. It sends the packets to its linked receivers M_B times. Thanks to the NACK local suppression task performed by the receivers, the active router receives only one NACK per packet loss. In total, it receives (M_B-1) NACKs. The active router will forward the NACKs to the source and to the other receivers if it does not have the packet, thus making $p_l\ (M_B-1)$ NACKs where p_l is the loss probability at the source link. Finally, we find:

$$\begin{aligned} E[A_i^{S_2}] = \quad & (1-p_l)\ E[M_S]\ E[Y_p^a]\ + E[M_B]\ E[X_p^a] + \\ & (E[M_B]-1)\ E[X_n^a] +\ p_l\ (E[M_B]-1)\ E[Y_n^a] \end{aligned} \tag{3.20}$$

If all the routers are active, we get the following expression:

$$\begin{aligned} E[A_i^{S_2}] = \quad & (1-p_l)\ E[M_A]\ E[Y_p^a]\ + E[M_B]\ E[X_p^a] + \\ & (E[M_B]-1)\ E[X_n^a] +\ p_l\ (E[M_B]-1)\ E[Y_n^a] \end{aligned} \tag{3.21}$$

For S_2^s that benefits from the subcast from the source, only the processing cost at the active routers and the free receivers will change. Since they receive only once each data packet, we can deduce from (3.18) and (3.20) the following statements:

$$E[Y_*^{S_2^s}] = E[Y_p] \; + \; \frac{p}{1-p} \; E[Y_n] + \; \frac{p^2}{1-p} \; E[Y_t] \tag{3.22}$$

and

$$\begin{aligned} E[A_i^{S_2^s}] = \quad & E[Y_p^a] \; + E[M_B] \; E[X_p^a] + \\ & (E[M_B]-1) \; E[X_n^a] + \; p_l \; (E[M_B]-1) \; E[Y_n^a] \end{aligned} \tag{3.23}$$

3.4.4 Analysis Protocols S_3 and S_3^s

In protocol S_3, the linked receivers benefit from the subcast feature so they receive a data packet only once. Therefore we have (based on (3.12)):

$$E[Y^{S_3}] = E[Y_p] \; + \; \frac{p}{1-p} \; E[Y_n] + \; \frac{p^2}{1-p} \; E[Y_t] \tag{3.24}$$

For the other nodes, we have the same results as for the S_1 protocol, that is:

$$E[X^{S_3}] = E[M_S] \; E[X_p] + \left(F \; \frac{p_l}{1-p_l} + (N-F)B \; \frac{p}{1-p} \right) E[X_n] \tag{3.25}$$

$$E[Y_*^{S_3}] = (1-p) \; E[M_S] \; E[Y_p] \; + \; \frac{p}{1-p} \; E[Y_n] + \; \frac{p^2}{1-p} \; E[Y_t] \tag{3.26}$$

$$\begin{aligned} E[A_i^{S_3}] = \; & (1-p_l) \; E[M_S] \; E[Y_p^a] \; + E[M_B] \; E[X_p^a] \\ & + B \; \frac{p}{1-p} \;\; E[X_n^a] + \; \frac{p_l}{1-p_l} \; E[Y_n^a] \end{aligned} \tag{3.27}$$

In S_3^s, the free receivers behave as in S_2^s (equation (3.22)):

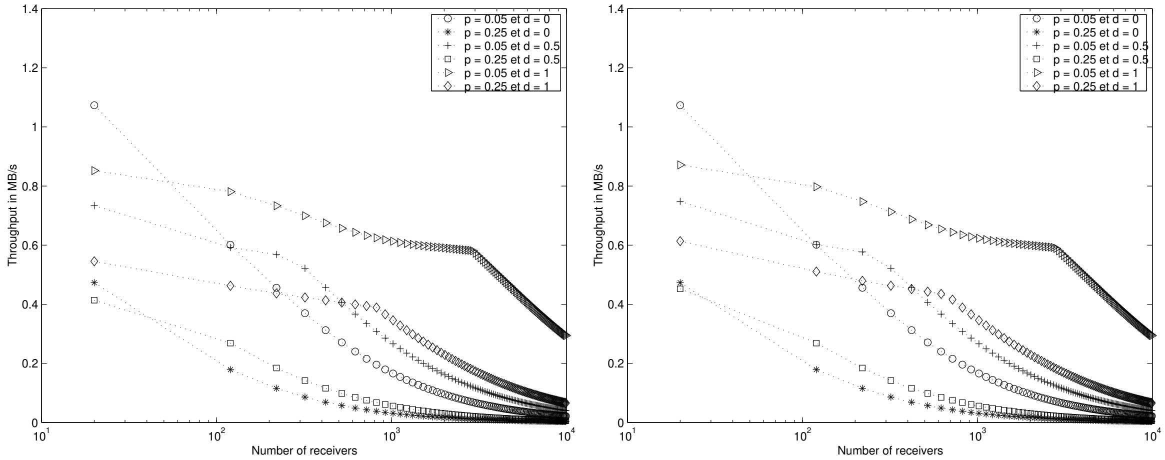
$$E[Y_*^{S_3^s}] = E[Y_p] \; + \; \frac{p}{1-p} \; E[Y_n] + \; \frac{p^2}{1-p} \; E[Y_t] \tag{3.28}$$

For the active routers, we can deduce from (3.15):

$$\begin{aligned} E[A_i^{S_3^s}] = \quad & E[Y_p^a] \; + E[M_B] \; E[X_p^a] \\ & + B \; \frac{p}{1-p} \;\; E[X_n^a] + \; \frac{p_l}{1-p_l} \; E[Y_n^a] \end{aligned} \tag{3.29}$$

For the other nodes, we have the same results as for the S_3 protocol:

$$E[X^{S_3^s}] = E[M_S] \; E[X_p] + \left(F \; \frac{p_l}{1-p_l} + (N-F)B \; \frac{p}{1-p} \right) E[X_n] \tag{3.30}$$



(a) (b)

Figure 3.2: Throughput achieved by (a) S_1 and (b) S_2.

$$E[Y^{S_3^s}] = E[Y_p] \ + \ \frac{p}{1-p} \ E[Y_n] + \ \frac{p^2}{1-p} \ E[Y_t] \tag{3.31}$$

$$E[Y_*^{S_3^s}] = (1-p) \ E[M_S] \ E[Y_p] \ + \ \frac{p}{1-p} \ E[Y_n] + \ \frac{p^2}{1-p} \ E[Y_t] \tag{3.32}$$

3.5 Numerical Results

In this section, we investigate the numerical results of our analysis of the generic protocols S_1, S_2, S_2^s, S_3 and S_3^s. For each protocol, we first examine the overall throughput achieved by the protocol and then we focus on the throughput achieved separately by each node. In addition, we will first consider that the active routers have the same processing time as the receivers, then we study the impact of varying the processing time of the active routers. The benefit of the subcast and the influence of the active routers density are considered too. The routers density (d) is defined as the ratio of active routers (F) to the number of all the routers (N) in the network model, thus $d = F/N$.

For the numerical evaluation, we take $E[X_p] = E[Y_p] = 500\ \mu s$, $E[X_n] = E[Y_n] = 85\ \mu s$ and $E[Y_t] = 32\ \mu s$. These values are those experimentally measured in [66]. For the mean processing time at the active routers, we set $E[X_p^a] = E[Y_p^a] = 500\ \mu s$ and $E[X_n^a] = E[Y_n^a] = 85\ \mu s$ as for the processing time of the end nodes. In [74] for instance, the authors have measured a data packet processing time within an active router of about 4 ms. They argue that this processing time can be reduced at least by a factor of 10. The study will also present results when varying the processing power at the active routers. For the throughput computation, the packet size is set to 1024 bytes.

We begin by examining the overall throughput achieved by the protocols S_1, S_2 and S_3. Figure 3.2 shows the overall throughput achieved by S_1 and S_2 as a function of the number of the multicast group size and this for different loss rates (p) and active routers density (d). We can see the benefit of active routers ($d \neq 0$) which permit a higher overall throughput, even with a high loss rate, and especially in the case of a large number of receivers. Local or global NACK suppression strategies have very close performances in terms of the overall throughput for very low loss rates. However S_2 performs better than S_1 when the loss probability increases. Although not shown, S_3 is identical to S_1 in terms of the overall throughput.

In order to know which nodes yield the minimum throughput, figure 3.3 plots the throughput achieved by each node in the protocols S_1 and S_3 as a function of the number of receivers with 50% of active routers and an end-to-end loss rate of 5%. We can observe that the minimal throughput in the system is first introduced by the active routers. Then the source turns out to become the bottleneck. This is due to the increasing number of the free receivers since only 50% of the routers are active. In order to be more scalable according to our network model, we require at least one active router in the path between every receiver and the source. We can observe in figure 3.3 that in S_2, a linked receiver has a similar throughput as in S_1. However in S_2, all the other nodes (free receivers, active routers and the source) exhibit more throughput than the same nodes in S_1 and S_3. In S_3, the linked receivers achieve a higher throughput compared to S_1 which does not benefit from the subcast facility.

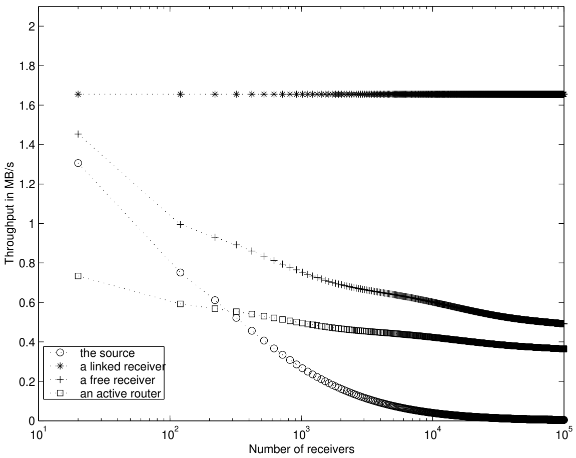
3.5.1 Local Suppression vs Global Suppression

Local suppression appears to provide slightly higher performances in terms of the overall throughput. It has been also observed that S_2 performs better than S_1 when the loss rate increases. In order to deeply compare the two suppression strategies, figure 3.4 plots the ratio of the linked receivers and the active routers throughput as a function of the end-to-end loss probability. Different local group sizes (B) are used. Figure 3.4a plots the throughput ratio of S_2 to S_1. It shows that S_2 always performs better than S_1 (at the linked receivers and the active routers). We observe that the benefit of the local suppression in S_2 compared to S_1, increases with the loss rate. In the local suppression strategy, NACKs are suppressed before they reach an active router thus justifying the benefit of this strategy for high loss probabilities.

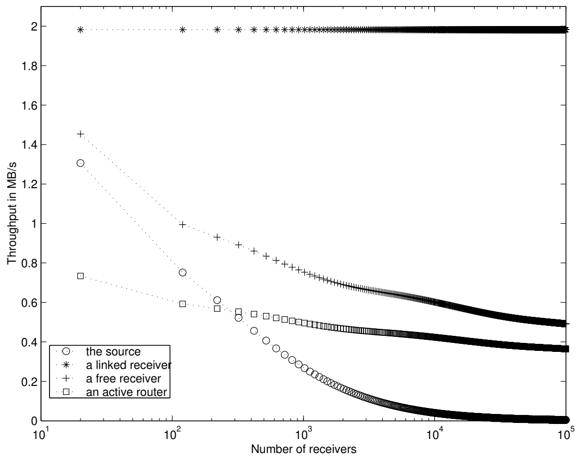
We plot in figure 3.4b, the throughput ratio of protocol S_2 to S_3 in order to be able to compare the two suppression strategies. We can see that independently of the loss rate, at the active routers, S_2 performs always better than S_3. However, S_3 performs better than S_2 at the linked receivers end till a high loss rate of about 90%. This is because the linked receivers under S_3 benefit from the subcast service. In S_3, a linked receiver receives only once a data packet. However in S_2, a linked receiver could receive more than one copy of the same data packet. Moreover, it continues to receive NACKs from its active router every time a receiver in its local group has experienced a loss.

3.5.2 Benefit of the Subcast

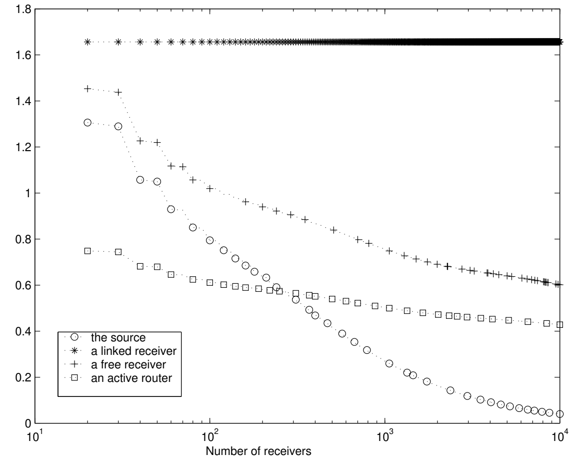
The subcast facility has the advantage of unloading the receivers and/or the active routers depending on whether we benefit from this facility from the source or not. To see the benefit of performing the subcast from the active routers associated to the linked receivers, figure 3.5 plots the throughput ratio at a linked receiver of S_3 to S_2. We can see that the subcast



(a)

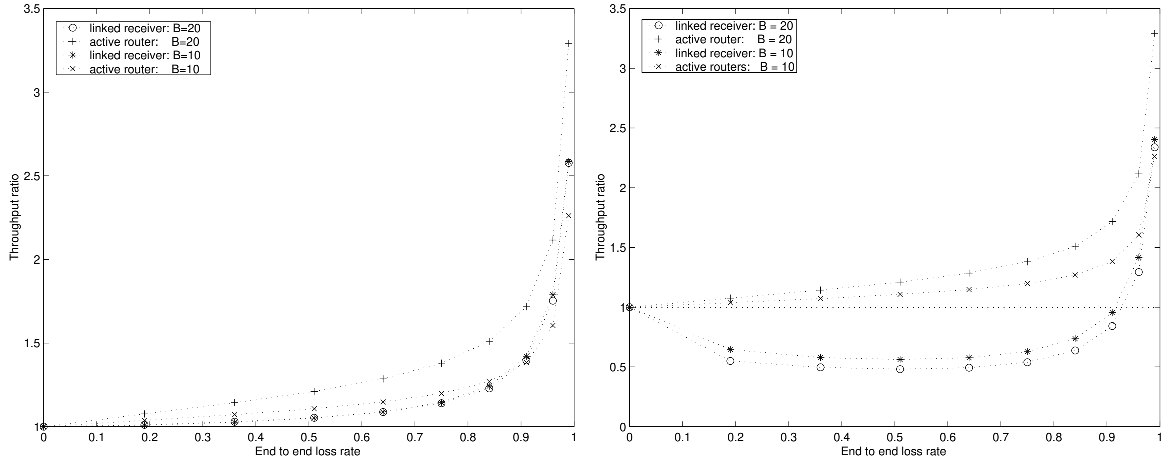


(b)



(c)

Figure 3.3: Throughput of the various nodes with (a) S_1, (b) S_3 and (b) S_2. p = 0.05, B = 10, 50% of active routers.



(a) (b)

Figure 3.4: Linked receivers and active routers' throughput ratio: (a) S_2/S_1 (b) S_2/S_3.

permits a higher throughput at the linked receivers in S_3. 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.

In addition to the linked receivers, performing the subcast from the source allows to unload both the active routers and the free receivers. Figure 3.6 shows that the subcast gives a constant and higher throughput at all the nodes that benefit from this facility. Note that in S_3^s, a linked and a free receiver achieve the same throughput. Both of them benefit from the subcast feature. Figure 3.7 shows the overall throughput achieved by protocols S_2^s and S_3^s. Protocols with the subcast from the source, perform better in terms of the overall throughput than those that do not benefit from this functionality (figure 3.2).

To show the benefit of the subcast from the source, figures 3.8a and 3.8b plot the throughput gain at the free receivers and the active routers as a function of the loss probability for S_2^s compared to S_2, and for S_3^s against S_3 respectively. Two different local group sizes are used ($B = 10$ and $B = 20$). At this stage, it is worth mentioning that the number of free receivers ($(N-F)B$) is proportional to the local group size (B). At the free receivers side, we can achieve a gain of 5 or even 6 for a loss rate of 20% and 50% respectively. We notice also that similar to the case of the linked receivers in figure 3.5, the subcast is more beneficial in the presence of a large number of free receivers. Moreover, for very low and high loss rates,

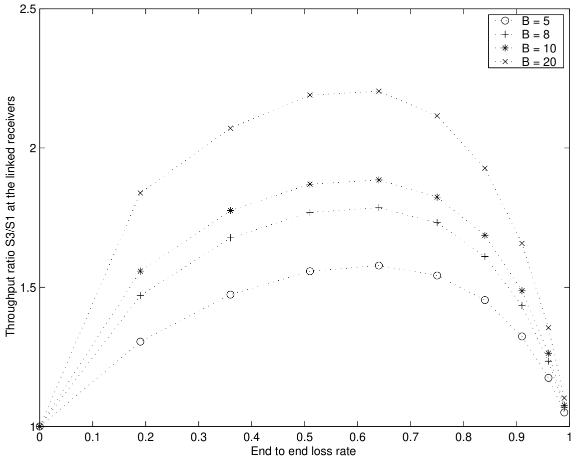
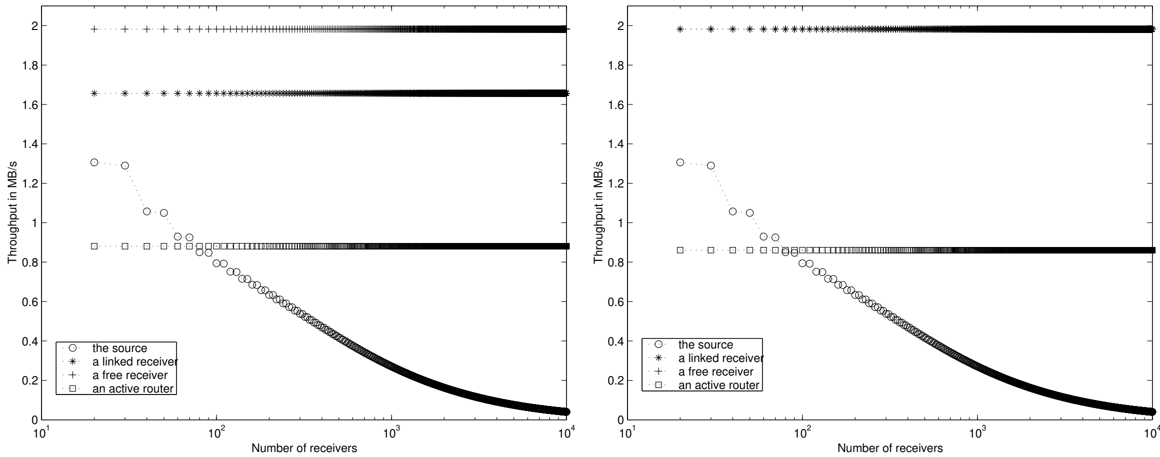


Figure 3.5: Benefit of the subcast for the linked receivers (S_3/S_2 ratio).



(a) (b)

Figure 3.6: Throughput of the various nodes in (a) S_2^s and (b) S_3^s, p = 0.05, B = 10, 50% of active routers.

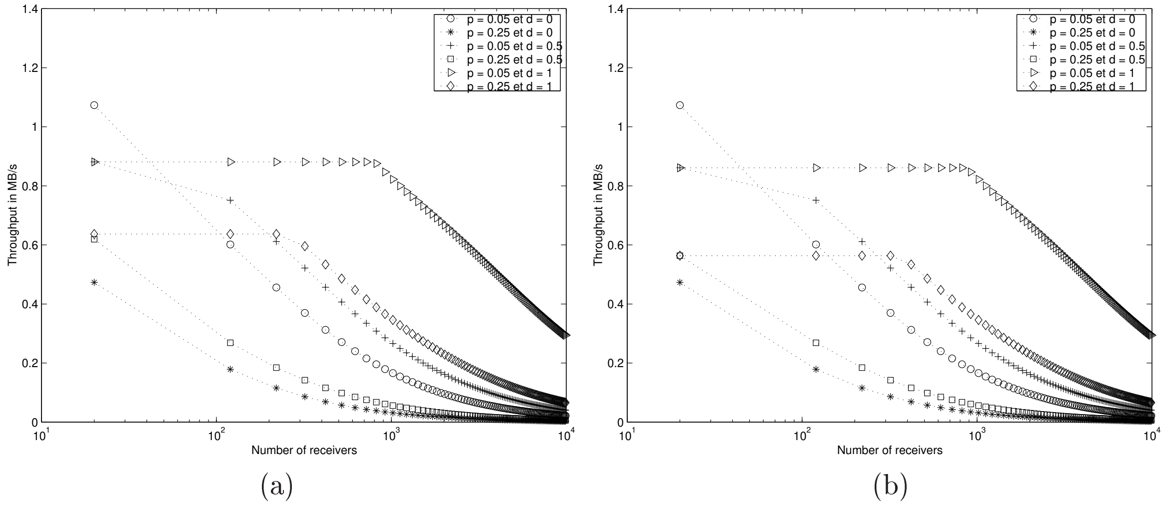


Figure 3.7: Throughput achieved by (a) S_2^s and (b) S_3^s.

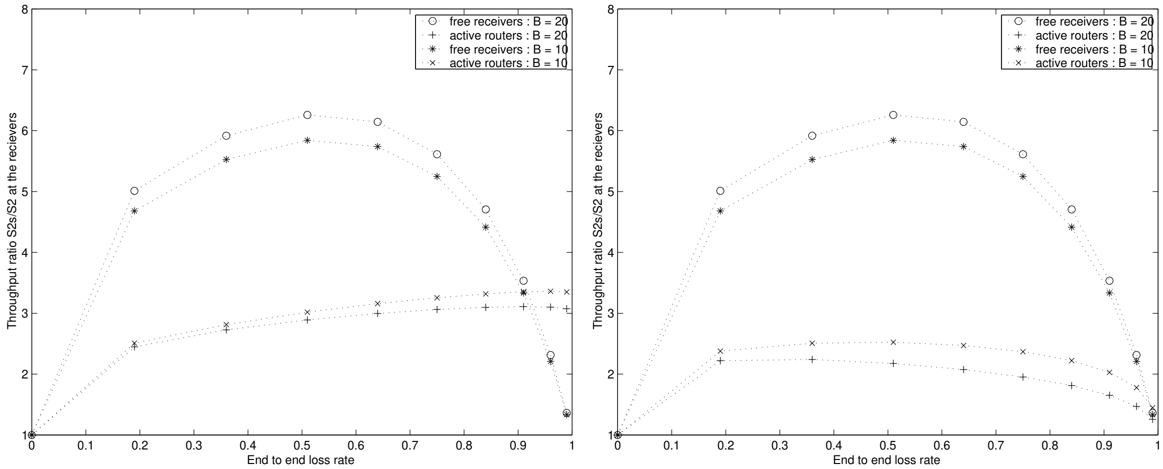
the subcast does not permit a noticeable gain. For the active routers, we can see that unlike S_3^s and S_3, S_2^s performs better than S_2 even for high loss rates. This is due to the fact that active routers in S_3^s receive all the NACKs generated in the local group which considerably increases with p. However, S_2 suppresses NACKs before they reach the active routers.

3.5.3 Active Routers Density

Figure 3.9 shows the impact of the active routers density on the protocol's performances in terms of the overall throughput. 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 at the active routers and the receivers, the overall throughput can be an order of magnitude higher if all the receivers are linked. Most interestingly, if the active router's processing power is divided by 10 in S_3 (figure 3.9a), we can still double the overall throughput provided that 55% of the routers are active. Figure 3.9b shows that in S_3^s, 55% of active routers are sufficient to double the performances even when dividing the processing power by 20. Although not shown S_1 and S_2 behave as S_3. S_2^s presents a slightly higher gain than S_3^s for low loss probabilities (eg. 0.05).

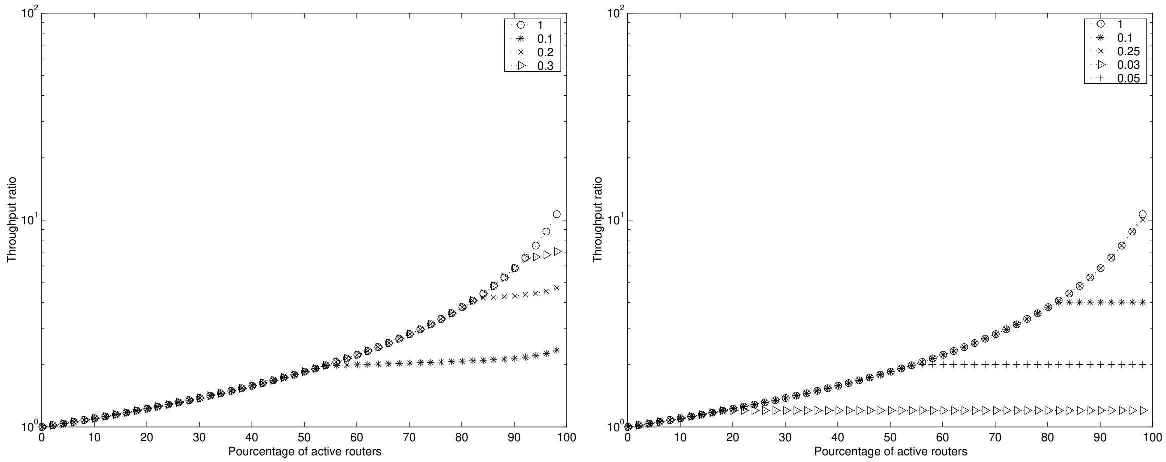
Conclusion

Reliable multicast appears to be a difficult problem. Active routers within the multicast tree can be used to perform additional functionalities such as the cache of data, feedback aggregation or the subcast in order to enhance performances of reliable multicast protocols. To evaluate the potential of these mechanisms, we did an analytical performance study of different active reliable multicast protocols. S_1 adopts the global NACK suppression strategy, S_2 performs a local suppression instead, and S_3 that differs from S_1 by the subcast service. S_2^s and S_3^s behave as S_2 and S_3 respectively but additionally benefit from a subcast facility from the source. All of the five evaluated protocols benefit from the cache of data at the active



(a) (b)

Figure 3.8: Benefit of the subcast from the source (a) S_2^s/S_2 and (b) S_3^s/S_3.



(a) (b)

Figure 3.9: Gain in terms of the overall throughput achieved by (a) S_3, (b) S_3^s when varying the active routers density. $p = 0.05$, $B = 10$ and $N = 1000$.

routers.

Our analysis uses the processing requirements of the protocols to derive the achievable throughput at the various nodes in the multicast tree. Numerical results have shown that local NACK suppression, under the assumption that the random waiting time at a receiver is well-chosen, requires less processing power than the global NACK suppression mainly for high loss rates. We have also shown the benefit of the subcast feature which allows a higher and constant throughput. Thus unloading the nodes that benefit from this facility, especially when it is applied from the source. Regarding the impact of the active routers density on performances, all protocols have the same behavior: the achievable throughput increases as the number of the active routers increases. Most interestingly, even with slower processing power at the active routers, increasing their number allows for more performances.

Chapter 4

An Analysis of a Router-based Loss Detection Service

4.1 Introduction

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 its 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 [52, 102, 137, 101, 118, 57].

Going a step further, we propose to enable loss detection at intermediate nodes (routers). This functionality can be easily implemented using an active networking environment [120] where routers themselves could contribute to enhance the network services by customized functionalities. In the previous chapter, we analyzed some of these functionalities (feedback suppression, cache of data and the subcast). In this chapter we investigate analytically the loss detection as a service performed by the routers themselves. In this case, routers are capable to detect packet losses and consequently generate the corresponding NACKs to be sent to the source.

Although simple, moving the loss detection into routers arises many questions such as "where to place such detection-capable routers?" and "what is the overhead of such an additional service?". The results presented in this chapter show that one must be careful when doing so in order to get any real benefit. The analytical study of the loss detection service is based on the processing overhead at both the end hosts and the active routers to derive the overall delay required by any receiver to correctly receive a data packet. The rest of the chapter is organized as follows. Section 4.2 presents the delay analysis of the loss detection service. Section 4.3 presents the numerical results and section 4.4 concludes.

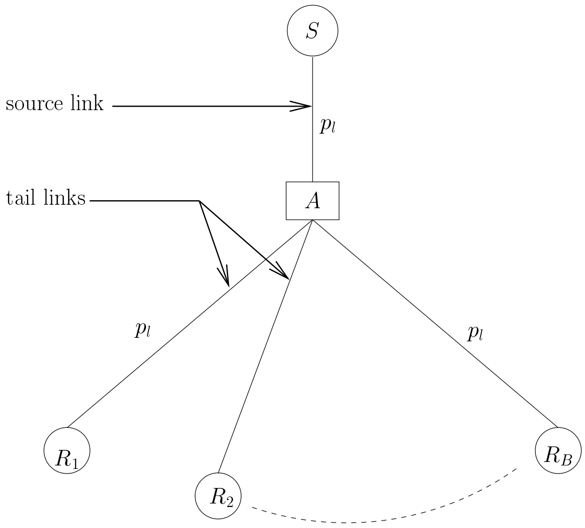


Figure 4.1: Simple network model.

4.2 Delay Analysis

4.2.1 Network Model

A commonly used model for evaluating multicast protocols is a multicast tree rooted at the source with receivers as leaves. Intermediate nodes are routers that (in the context of this study) are able to perform customized processing (services) on the packets (data packets and NACKs) flowing through them. The first supported service is the NACK suppression service which consists in ignoring subsequent NACKs for the same data packet during a given amount of time. For our analysis, we will assume that this "*duplicate discard*" period is well chosen. Therefore only one NACK is forwarded towards the source for each loss. The second active service consists in the subcast functionality where repair packets are sent only to the affected receivers. The subcast is performed thanks to the soft state information maintained at the routers so that a repair is only retransmitted to the affected receivers.

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 generates immediately a NACK packet towards the source and initializes a timer. The expiration of this timer without receiving the data packet, triggers the emission of another NACK. All NACK packets received for this data packet from the downstream links are ignored until the expiration of the corresponding timer.

For this delay analysis, we consider a simple network model with a two level multicast tree. One source multicasts data packets to one group composed of B receivers $R_1, R_2, \ldots R_B$ connected to the source via an active router A (figure 4.1). We consider that the source link and the tail links have a loss probability of p_l. Therefore the end-to-end loss probability perceived by a receiver is $p = 1-(1-p_l)^2$. The losses are assumed to be temporally independent and those at the tail links are assumed to be mutually independent. Once again, we will assume for this analysis that the NACKs will never be lost.

The computational framework adopted in our analysis is similar to the one provided in [134]. 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 what follows, let M (respectively M_r) be the number of transmissions of a data packet from the source until all the receivers (one receiver) have (has) correctly received the data packet. M_a is the number of transmissions of a data packet from the source until the active router has correctly received the data packet. We have:

$$P[M \leq m] = (1-p^m)^B$$

thus:

$$E[M] = 1 + \sum_{m=0}^{\infty} 1 - (1-p^m)^B$$

For M_r we have:

$$P[M_r \leq m] = 1 - p^m$$

thus,

$$E[M_r] = \frac{1}{1-p}$$

Similarly, we get for M_a:

$$E[M_a] = \frac{1}{1-p_l}$$

Finally, we will note X (respectively X_a) and Y (respectively Y_a) the service time required for processing a data packet and a NACK at the source and the receivers (respectively the active router). We assume that the source is multicasting at a rate of λ packets per unit of time. Table 4.1 summarizes most of the used notations.

B	the number of the receivers.
p_l, p	respectively, the loss rate at a link (source or tail) and the end-to-end loss rate perceived by a receiver.
λ	data packet transmission rate at the source.
$\lambda_d^{x,\omega}$	data packet arrival rate at node $x \in \{s, a, r\}$ in protocol $\omega \in \{A, D\}$
$\lambda_n^{x,\omega}$	NACK packet arrival rate at node x in protocol ω
ρ_x^ω	the load at the node x in protocol ω
W_x^ω	the mean waiting time at node x in protocol ω
X, Y	respectively, the service time required for processing a data packet and a NACK at the source and the receivers.
X_a, Y_a	respectively, the service time required for processing a data packet and a NACK at the active router.
M, M_r	the number of transmissions of a data packet from the source until all the receivers (one receiver) have (has) correctly received the data packet.
M_a	the number of transmissions of a data packet from the source until the active router has correctly received the data packet.
L_r	For a lost packet, L_r is the number of subsequent packets which are lost by both the randomly chosen receiver and all the other receivers that have also lost the considered data packet.
L_a	the number of subsequent packets lost by the active router.
T_{sa} (T_{ar})	the propagation delay from the source (a receiver) to the active router.
T_{sr}	the propagation delay from the source to a receiver. $T_{sr} = T_{sa} + T_{ar}$.
dpd^ω	the required delay to receive a data packet from the source by a receiver in protocol ω. $dpd^\omega = T_{sr} + 2X_a + W_a^\omega$.
npd^ω	the required delay to receive a NACK packet from any receiver by the source in protocol ω. $npd^\omega = T_{sr} + 2Y_a + W_a^\omega$.
$Tout_r$	the value of the timeout set at the receivers when requesting a retransmission.
Δ^ω	the overall delay so a randomly chosen packet is received by a randomly chosen receiver.
Δ_{det}^ω	the time required by a receiver to detect the loss of a data packet.
Δ_{recov}^ω	the time that a recovery would take.
$\Delta_{det,receiver}^D$	the time required to detect a loss by a receiver.
$\Delta_{det,router}^D$	the time required to detect a loss by an active router.

Table 4.1: Notation used in the delay analysis

4.2.2 Waiting Times

In protocol A, the data packet arrival rate at the source is $\lambda_d^{s,A} = \lambda$ with a mean requirement service of $E[X]$. The NACK packet arrival rate at the source is $\lambda_n^{s,A} = \lambda \ (E[M]-1)/B$ with a mean requirement service of $E[X]+E[Y]$. The NACK arrival rate corresponds to the $(E[M]-1)/B$ NACKs received from the receivers (divided by B taking into consideration the NACK suppression service at the router before reaching the source). The processing requirements correspond to the reception of a NACK and the reemission of the corresponding repair. Therefore the load at the source is:

$$\rho_s^A = \lambda \ E[X] + \lambda \ \frac{E[M]-1}{B} \ (E[X]+E[Y])$$

giving:

$$\rho_s^A = \lambda \ E[X](1+\frac{E[M]-1}{B}) + \lambda \ E[Y] \ \frac{E[M]-1}{B} \tag{4.1}$$

The mean waiting time at the source is:

$$E[W_s^A] = \frac{\lambda \ E[X^2] + \lambda \ \frac{E[M]-1}{B} \ E[(X+Y)^2]}{2(1-\rho_s^A)}$$

Since $E[(X+Y)^2] = E[X^2] + 2E[X]E[Y] + E[Y^2]$, we get:

$$E[W_s^A] = \lambda \ \frac{E[X^2](1+\frac{E[M]-1}{B}) + E[Y^2] + 2E[X]E[Y]}{2(1-\rho_s^A)} \tag{4.2}$$

The active router receives on average $E[M]$ times a packet with a probability of $(1-p_l)$. Therefore, the data packet arrival rate at the router is $\lambda_d^{a,A} = \lambda E[M](1-p_l)$ with a mean requirement service of $2E[X_a]$ (the router receives the packet and then forwards it). On average, the active router receives $(M_r-1)B$ NACK packets from the B receivers downstream. Therefore, the NACK packet arrival rate is $\lambda_n^{a,A} = \lambda(E[M_r]-1)B$ with a mean requirement service of $E[Y_a] + E[Y_a]/B$ (the router receives all the NACKs generated for a lost data packet but forwards only one NACK upstream because of NACK aggregation). The load at the active router for protocol A is:

$$\rho_a^A = 2\lambda \ E[M](1-p_l)E[X_a] + \lambda(E[M_r]-1)(1+B)E[Y_a] \tag{4.3}$$

The mean waiting time at the active router is:

$$E[W_a^A] = \frac{4\lambda_d^{a,A} \ E[X_a^2] + \lambda_n^{a,A} \ E[Y_a^2](B+1)^2/B^2}{2(1-\rho_a^A)} \tag{4.4}$$

At a randomly chosen receiver the data packet arrival rate is $\lambda_d^{r,A} = \lambda$ with a required service of $E[X]$. This is due to the fact that the receiver receives only once each data packet thanks to the subcast functionality. The NACK packet arrival rate is $\lambda_n^{r,A} = \lambda(E[M_r] - 1)$ with a mean requirement service of $E[Y]$. The load at the receiver's side for protocol A is:

$$\rho_r^A = \lambda E[X] + \lambda(E[M_r] - 1)E[Y] \tag{4.5}$$

The mean waiting time at a receiver under protocol A is given by:

$$E[W_r^A] = \lambda \frac{E[X^2] + (E[M_r] - 1)E[Y^2]}{2(1 - \rho_r^A)} \tag{4.6}$$

The waiting times in protocol D for both the sender and the receivers are identical to those of protocol A. At the active router the data packet arrival rate is $\lambda_d^{a,D} = \lambda E[M](1-p_l)$ with a mean requirement service of $2E[X_a]$. On average the active router receives $(M_r - 1)B$ NACK packets from the receivers. $M_a - 1$ NACK packets are ignored by the router because it has recently sent a similar NACK. Therefore, only $(M_r - 1)B - M_a + 1$ will be forwarded towards the source. Hence, the arrival rate of received but ignored NACKs is $\lambda_g^{a,D} = \lambda(E[M_a] - 1)$ with a mean requirement service of $E[Y_a]$. For the received and forwarded NACKs, we have $\lambda_n^{a,D} = \lambda((E[M_r] - 1)B - E[M_a] + 1)$ with a mean requirement service of $(E[Y_a] + E[Y_a]/B)$. Consequently, the load at the active router for protocol D is:

$$\begin{aligned} \rho_a^D = \; & 2\lambda E[M](1 - p_l)E[X_a] + \\ & \lambda(E[M_a] - 1 + (B + 1)(E[M_r] - 1) - (M_a - 1)/B)E[Y_a] \end{aligned} \tag{4.7}$$

and the mean waiting time at the active router for protocol D is:

$$E[W_a^D] = \frac{4\lambda_d^{a,D} E[X_a^2] + (\lambda_g^{a,D} + \lambda_n^{a,D}(B + 1)^2/B^2)E[Y_a^2]}{2(1 - \rho_a^D)} \tag{4.8}$$

4.2.3 Overall Delay

For protocol $\omega \in \{A, D\}$, the overall delay Δ^ω for a data packet to be received by a randomly chosen receiver includes the time required to detect the loss Δ_{det}^ω and the time required to perform the recovery Δ_{recov}^ω, therefore we have:

$$E[\Delta^\omega] = E[\Delta_{det}^\omega] + E[\Delta_{recov}^\omega]$$

To compute these times, we need to introduce two random variables L_r and L_a. Assuming that the lost packet has a sequence number of i, L_r is the number of subsequent packets with a sequence number greater than i which are lost by both the randomly chosen receiver and all the other receivers that have also lost the ith data packet. The expression for the mean of L_r is given in [134]:

$$E[L_r] = \sum_{k=0}^{B-1} C_{B-1}^k \; p^k (1 - p)^{(B-k-1)} \frac{p^{k+1}}{1 - p^{k+1}}$$

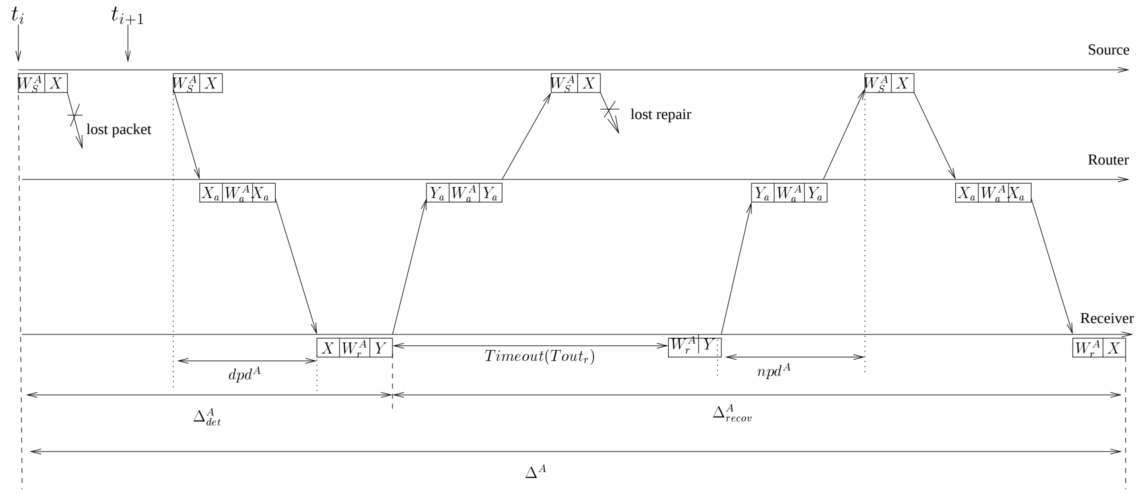


Figure 4.2: Overall delay diagram for protocol A.

Similarly L_a is the number of subsequent packets with a sequence number greater than i that are lost by the active router. Since we have only one active router in our model then $E[L_a] = p_l/(1-p_l)$. For the overall delay analysis, we note T_{sa} (T_{ar}) as the propagation delay from the source (a receiver) to the active router. The propagation delay from the source to a receiver is noted $T_{sr} = T_{sa} + T_{ar}$. The required delay to receive a data packet from the source by a receiver in protocol ω is $dpd^\omega = T_{sr} + 2X_a + W_a^\omega$. The required delay to receive a NACK packet from any receiver by the source in protocol ω is noted $npd^\omega = T_{sr} + 2Y_a + W_a^\omega$.

Figure 4.2 and 4.3 show the overall delay diagrams for protocol A and D. We have accordingly $\lambda = t_{i+1} - t_i \ \forall i$. For protocol A, the loss detection time is given by :

$$E[\Delta^A_{det}] = \quad E[L_r]/\lambda + E[W_s^A] + E[X] + E[dpd^A] + E[Y] + E[W_r^A] + E[X] \tag{4.9}$$

To estimate a mean for Δ^A_{recov}, we take into consideration the fact that a repair packet may be lost. Consequently Δ^A_{recov} includes the delay incurred by a given number $(j-1)$ of timeout expirations (if the data packet needs to be transmitted j times), the time required to receive the last NACK by the source and the time required to receive the repair by the receiver. From figure 4.2, we can see that in addition to this time, there is the delay required for the NACK to be received by the source (npd^A), the processing time at the source ($W_s^A + X$), the delay required for the repair to be received by the receiver (dpd^A) and the processing time at the receiver ($W_r^A + X$). Therefore, we can write:

$$E[\Delta^A_{recov}] = \quad E[npd^A] + (E[W_s^A] + E[X]) + E[dpd^A] + (E[W_r^A] + E[X]) + (Tout_r + E[W_r^A] + E[Y]) \sum_{j=1}^{\infty} (j-1) p^{j-1}(1-p)$$

where $p^{j-1}(1-p)$ is the probability that there are $(j-1)$ retransmissions of the data packet until its correct reception by the randomly chosen receiver. $Tout_r$ is the value of the timeout set at the receivers. Finally, we have :

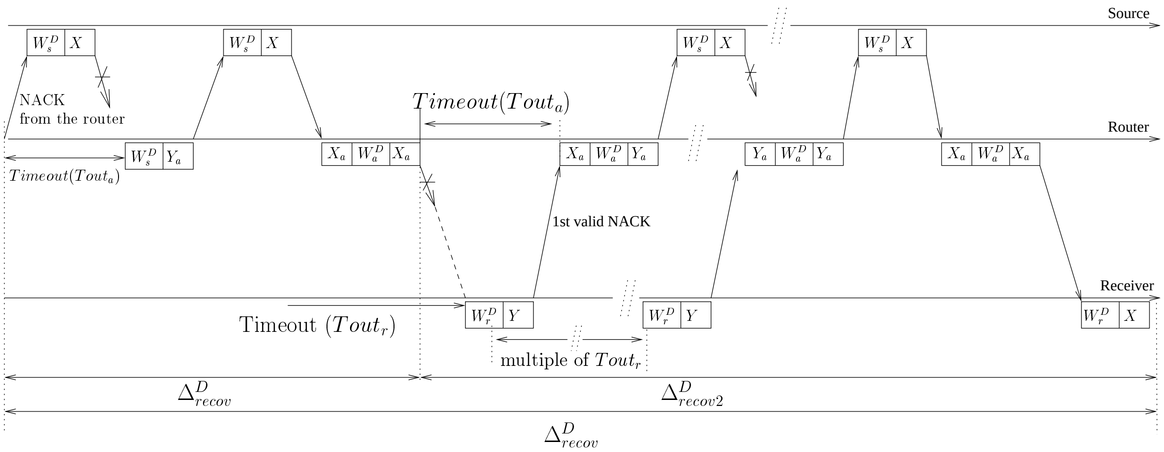


Figure 4.3: Overall delay diagram for protocol D.

$$\begin{aligned} E[\Delta^A_{recov}] = \quad & E[npd^A] + E[dpd^A] + E[W^A_s] + E[W^A_r] + \\ & 2\ E[X] + (Tout_r + E[W^A_r] + E[Y])\ p/(1-p) \end{aligned} \tag{4.10}$$

For protocol D, a loss may be detected by the router with a probability p_l (the probability that the data packet is lost at the source) or by any of the affected receivers otherwise. Therefore, Δ^D_{det} is a function of $\Delta^D_{det,receiver}$ and $\Delta^D_{det,router}$ which are respectively the time required to detect a loss by a receiver and by an active router. Therefore, we have :

$$E[\Delta^D_{det}] = (1-p_l)E[\Delta^D_{det,receiver}] + p_l E[\Delta^D_{det,router}]$$

Using the same method to obtain the loss detection delay in protocol A, we find that for protocol D:

$$E[\Delta^D_{det,receiver}] = \quad E[L_r]/\lambda + E[W^D_s] + E[dpd^D] + E[Y] + E[W^D_r] + 2E[X] \tag{4.11}$$

and finally,

$$E[\Delta^D_{det,router}] = \quad E[L_a]/\lambda + E[W^D_s] + E[X] + T_{sa} + E[X_a] + E[Y_a] + E[W^D_a]$$

which gives:

$$E[\Delta^D_{det,router}] = \quad p_l(1-p_l)/\lambda + E[W^D_s] + E[X] + T_{sa} + E[X_a] + E[Y_a] + E[W^D_a] \tag{4.12}$$

Similarly, the recovery delay can be written as follows:

$$E[\Delta^D_{recov}] = (1-p_l)E[\Delta^D_{recov,receiver}] + p_l\ E[\Delta^D_{recov,router}]$$

where $\Delta^D_{recov,receiver}$ is the required time to recover a missing packet if the loss is detected by the router. Similarly $\Delta^D_{recov,router}$ is the required time to recover a lost packet if the loss is detected by a receiver. $\Delta^D_{recov,receiver}$ is similar to Δ^A_{recov}, so:

$$\begin{aligned} E[\Delta^D_{recov,receiver}] = \quad & E[npd^D] + E[dpd^D] + E[W^D_s] + E[W^D_r] + 2\ E[X] + \\ & (Tout_r + E[W^D_r] + E[Y])\ p/(1-p) \end{aligned}$$

Since a router sets a timeout and retransmits the NACK if the required data packet has not been received yet, $\Delta^D_{recov,router}$ can be expressed by:

$$E[\Delta^D_{recov,router}] = E[\Delta^D_{recov1}] + E[\Delta^D_{recov2}]$$

where $E[\Delta^D_{recov1}]$ is the required delay until the active router has received the data packet so it will never generate a NACK for it. $E[\Delta^D_{recov2}]$ is the required delay to receive the data packet by the randomly chosen receiver after the active router has already received it. Referring to figure 4.3 and following the same method used for Δ^A_{recov}, we find:

$$\begin{aligned} E[\Delta^D_{recov1}] = \quad & 2T_{sa} + E[W^D_s] + E[X] + E[W^D_a] + 2E[X_a] + \\ & (Tout_a + E[W^D_a] + E[Y_a])\ p_l/(1-p_l) \end{aligned} \quad (4.13)$$

Under the assumption that the first valid NACK is received just at the expiration of the timeout at the router, and using the same method applied to Δ^A_{recov}, we can derive:

$$\begin{aligned} E[\Delta^D_{recov2}] = \quad & E[npd^D] + E[dpd^D] + E[W^D_s] + E[W^D_r] + 2E[X] + \\ & Tout_a - T_{ar} + (Tout_r + E[W^D_r] + E[Y])\ p/(1-p) \end{aligned} \quad (4.14)$$

4.3 Numerical Results

For the numerical study, we set the values of the different parameters to those measured in [67] and normalized to the time needed to transmit a data packet ($E[X] = 1$) which is of about 500 μsecs. We have accordingly $E[Y] = 0.2$ corresponding to 100 μsecs. The processing overhead at the routers is considered to be the double of the time required to process a data packet at the end hosts, $E[X_a] = E[Y_a] = 2E[X] = 2$.

4.3.1 Router Location

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. Particular values of α are $\alpha_c = (T_{sr} - 1)/T_{sr}$ which corresponds to the case 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.

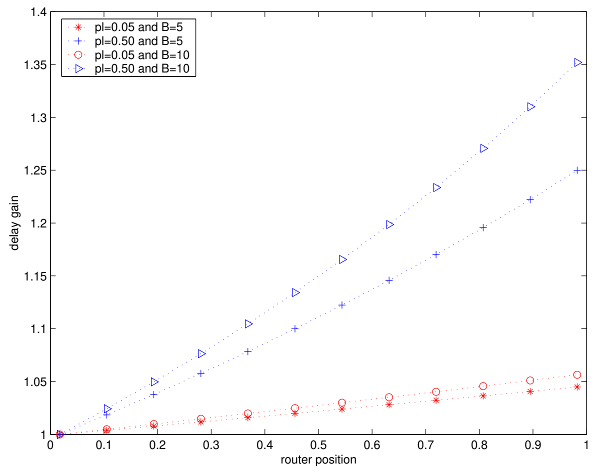


Figure 4.4: The delay gain in D as a function of the router position.

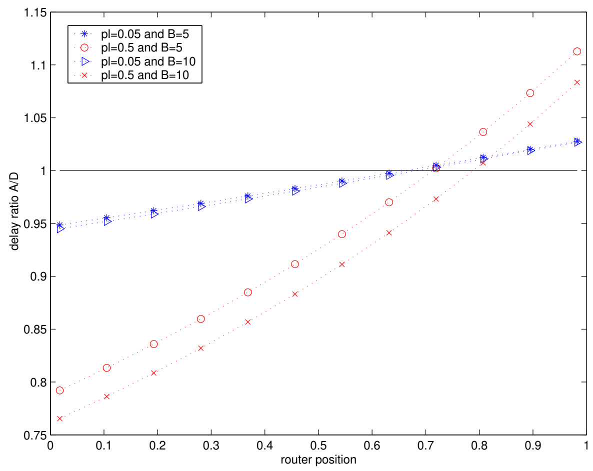


Figure 4.5: The delay ratio A/D as a function of the router position.

Figure 4.4 plots the ratio of the delay achieved by D when we change the router position (the α parameter) 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 the largest when the router is the closest to the source mainly 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.

Figure 4.5 plots the delay ratio of A to D as a function of the router position. We can see that D performs better than A only when the router is sufficiently close to the source. Otherwise, the overhead introduced by generating NACKs in D becomes unjustified. This is due to the longer distance that must be crossed by the NACK packet before reaching the source.

Based on these first observations, we will consider in what follows, only the case where the active router is sufficiently close to the source ($\alpha = \alpha_c$).

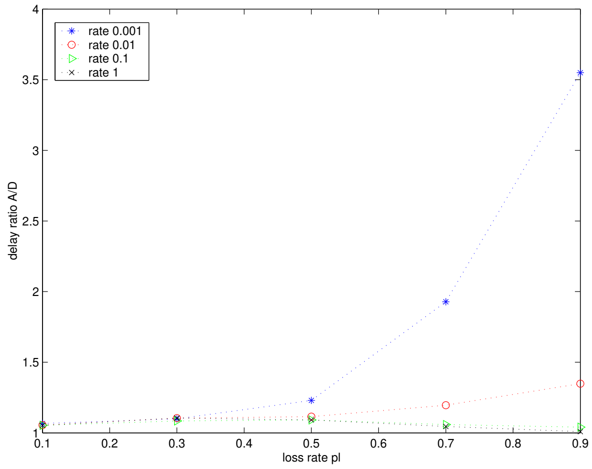


Figure 4.6: The delay ratio A/D as a function of the loss rate p_l.

4.3.2 Comparing A and D

Figure 4.6 plots the gain achieved by D with respect to A as a function of p_l for different sending rates. B is set to 5. The benefit of D over A increases as the loss rate increases. For instance, for a sending rate of 0.001, protocol D reduces the delay compared to protocol A by a factor of 3.5 for a loss rate of 90%. In the other cases, even if the gain is not significant D still performs better than A.

To accurately examine the behavior of the different nodes and to know which node is overloaded before the others, Figure 4.7 shows the load at the different nodes in A and D as a function of the loss rate. The processing overhead at the routers is considered to be twice the required processing time at the end hosts. We can see that the load at the source and the receivers is the same in both A and D. The load at the router in D is only slightly greater than A so the loss detection service does not introduce a significant processing overhead at the routers. The routers are the most overloaded nodes because of the processing overhead introduced by the active services. The source is more loaded than the receivers since it is responsible of the retransmissions. We can see from figure 4.7b where we increased the sending rate that the load increases in all nodes of the system.

Another deciding factor is the number of receivers associated to the active router. Figure 4.8 shows that D has lower end-to-end delay than A. For low loss rates (i.e. 5% to 10%) protocol D introduces a gain of only 5% that does not vary even if we increase the number of receivers. The benefit of D over A can be better seen when the loss rate is increased. In fact, for a loss rate of 25%, the gain of D over A increases from 7% for 1 receiver to 27% for 50 receivers. Protocol D shows better performances than A especially for high loss rates and a large number of receivers.

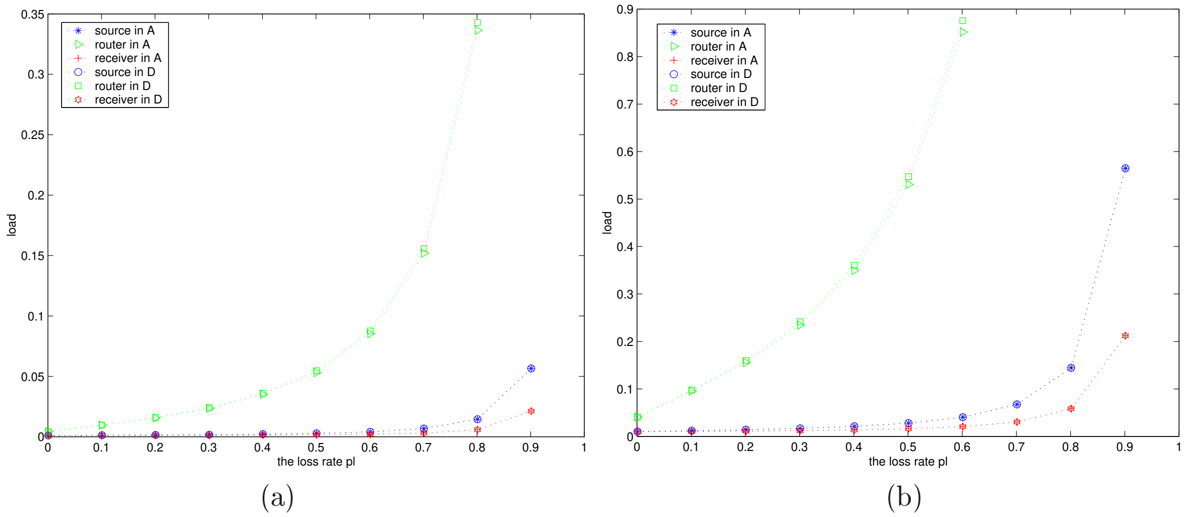


Figure 4.7: Load at the different nodes with 5 receivers (a) $\lambda = 0.001$ (b) $\lambda = 0.01$.

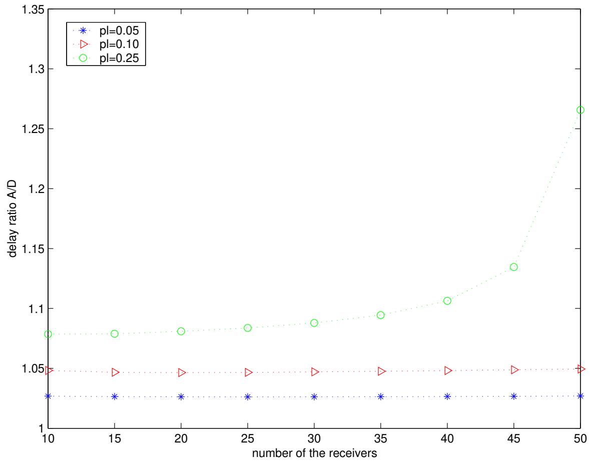


Figure 4.8: The gain for $\lambda = 0.01$ with $B = 5$.

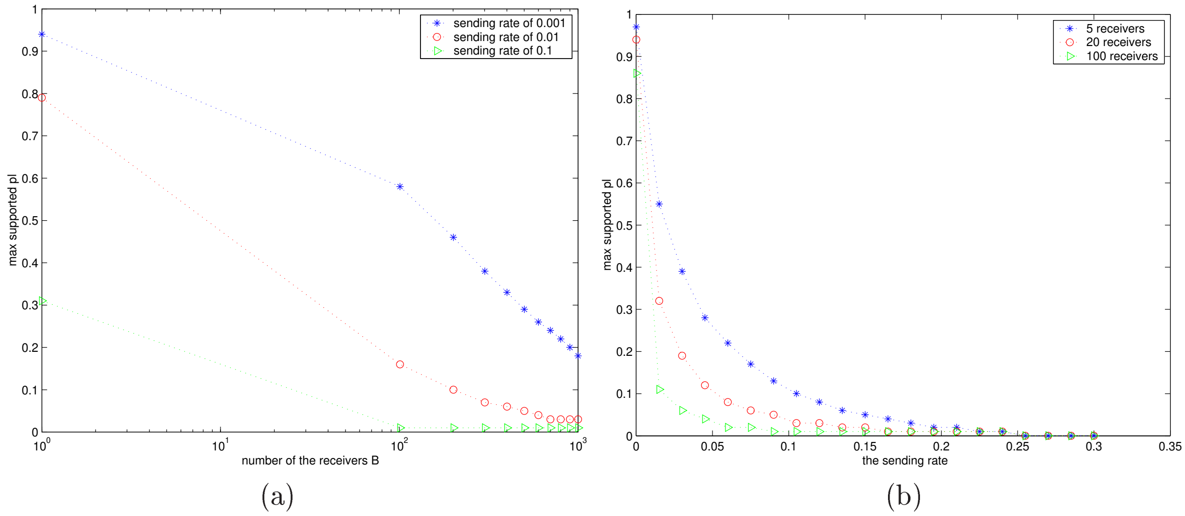


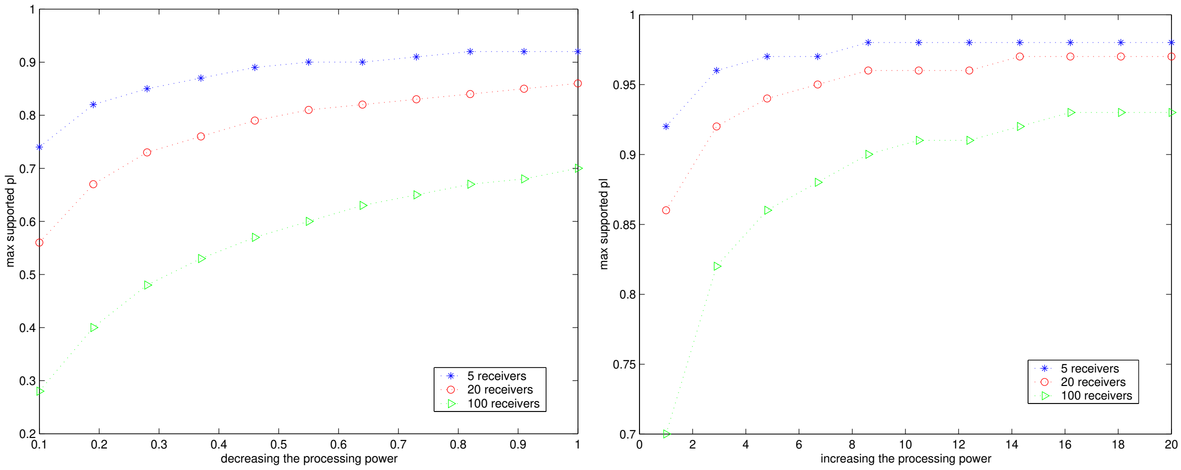
Figure 4.9: Maximum supported p_l in D as a function of (a) B, the number of the receivers and (b) λ, the sending rate.

4.3.3 Maximum Supported Loss Rate

Another important aspect to examine is the maximum loss rate supported by the entire system before one of the nodes is saturated. Figure 4.9a shows the maximum supported loss rate (p_l) as a function of the number of receivers. We can see that for a sending rate of $\lambda = 0.001$ with one receiver, the system will be able to support a loss rate higher than 90%. However when the number of receivers increases, the supported loss rate decreases. To see the impact of the sending rate, figure 4.9b plots the maximum supported loss rate as a function of the sending rate for several group sizes. As expected, we can see that the maximum supported loss rate decreases as the sending rate increases.

4.3.4 Routers' Processing Power

Till now we have considered only the case where the processing overhead of the active router is twice the processing time required at the end hosts. Figure 4.10 plots the maximum supported p_l as a function of the processing power. Figure 4.10a 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 4.10b 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 still not be the bottleneck.



(a) (b)

Figure 4.10: Maximum p_l supported by D as a function of the routers processing power.

4.4 Conclusion

Reliable multicast protocols can benefit from the active networking technology where routers can execute additional functionalities. In this chapter, we proposed one service which consists in the loss detection by the routers themselves. In order to evaluate the potential of this proposed service, we provided a delay analysis of two active reliable protocols noted A and D.

The numerical results showed that 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. This result can be used to propose an active multicast architecture with specialized routers (see section 5.6). For instance the closest router to the source should perform the loss detection while the others will only perform the other active services such as the NACK suppression and the subcasting. The load at the different nodes was also examined and we observed that the routers are the bottleneck when their processing overhead is set to twice the processing requirement at the end hosts. Nevertheless, we showed that we do not need to increase the routers' processing power infinitely so they are no longer the bottleneck.

Chapter 5

DyRAM : a Reliable Multicast Protocol

5.1 Introduction

Since it is required by many emerging Internet applications, reliable multicast has received significant attention in the networking research community. Scalability to a large number of receivers is one of the key challenges to be met by reliable multicast protocols. Tree-based approaches have been proven to be the most efficient. Organizing the receivers in a hierarchy allows for the implementation of local recoveries, thus unloading the source from performing retransmissions in addition to reducing the recovery latency. Moreover, receivers' feedback can be suppressed through the hierarchy, thus avoiding feedback implosion. Finally, retransmissions can be more localized thanks to the hierarchy structure, thus eliminating duplicate and unwanted repairs.

In a tree-based approach, mechanisms to build and maintain the hierarchy structure are required. To be more efficient, the built hierarchy has to be congruent with the underlying multicast distribution tree in addition to its ability to adapt to the dynamic nature of a multicast group. DyRAM (for *Dynamic Replier-based Active reliable Multicast*, our reliable multicast protocol that will be presented in this chapter) is one protocol that satisfies both congruence and adaptability requirements. In DyRAM, intermediate nodes (routers) in the multicast tree are involved in the data recovery process. On one hand, a router contributes in the election of "repliers" to retransmit lost packets, thus building an implicit recovery tree that is congruent with the physical multicast tree. On the other hand, a replier is elected on a per packet-basis thus allowing for an adaptability to dynamic changes in the multicast group topology. Since our proposal is router-assisted, additional design goals are required. Mainly, we have been careful about reducing the amount of buffering requirements at the network elements and minimizing the overhead at the routers by implementing light-weight functionalities.

The rest of this chapter is organized as follows. Section 5.2 gives an overview of the DyRAM protocol while the details are provided in section 5.3. Afterwards section 5.4 shows how the RTTs are estimated in a scalable way in order to set the different timeouts of the protocol. Flow Control at the end-points and memory usage at the routers are addressed in section 5.5. In section 5.6, we propose our active architecture which is more suitable to reliable multicast protocols. Section 5.7 shows how DyRAM is implemented in an active execution environment and preliminary results will be presented in measuring the raw processing time at the routers. Finally, section 5.8 concludes. The evaluation of DyRAM using simulations is performed in the next chapter 6.

5.2 DyRAM Overview

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 the routers. DyRAM uses a NACK-based scheme with receiver-based local recovery 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, positive acknowledgements (ACKs) are piggybacked on negative ones (NACKs). In the absence of NACKs, ACKs are also periodically piggybacked on special messages called "Congestion Reports"(CRs). These CRs contain the necessary information for the flow and congestion control addressed respectively in section 5.5 and chapter 7.

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 in addition to load-balancing features.
- the early loss detection of packet losses and the emission of the corresponding NACKs. This is very helpful in providing a low recovery delay.

5.3 DyRAM Details

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. We assume that feedback messages from both receivers and routers follow the reverse path of the data packets. This assumption is essential for taking advantage of active services. This behavior can be implemented (as already outlined) by using a dedicated routing service for feedback packets where every node in the multicast tree is aware of its parent node in the multicast tree. Symmetric routing protocols such as MOSPF

[93] and HBH [28] allow for an easy implementation of this desirable behavior.

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 5.4.

5.3.1 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 them (see figure 5.1). 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 5.5). 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.

A NACK or a CR packet includes 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[1] experienced by the subtree from which the CR is received. These two fields are used by the DyRAM congestion avoidance algorithm. In this section, only the active services related to reliability are described. RTT estimation and flow control related services are considered in sections 5.4 and 5.5. Congestion control is addressed in chapter 7.

[1]The RTT variation experienced by a link in a multicast tree is defined as the difference between the two last consecutive RTT measures at the worst receiver located downstream from this link. How this RTT variation is estimated is addressed in chapter 7.

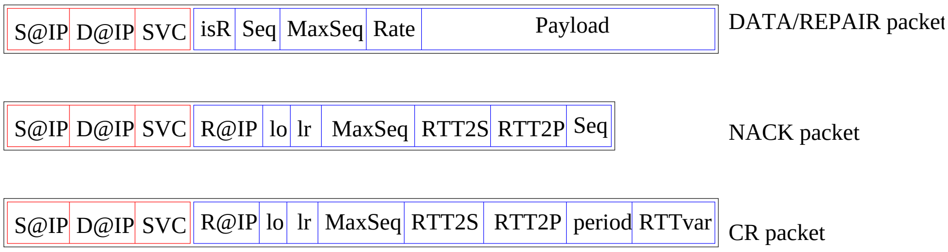


Figure 5.1: Packets structure.

5.3.2 Router's Soft State

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. An 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 identity of links from which a NACK has been received.

DyRAM also enables each 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 (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 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 5.3.3), we keep within a router a link state structure (LS) which contains for every downstream link the following[2]:

- $lastOrdered$ is the sequence number of the last data packet received in order by this link (i.e. by all the receivers behind this link).
- $lastReceived$ is the sequence number of the last received data packet by this link.
- $RTTvar$ is the last RTT variation experienced by this link.

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

- *RTT* is the last experienced RTT by this link. If there are multiple receivers downstream this link, we consider the RTT of the closest one.
- *weight* is a value assigned to this link which is proportional to its capacity to be a replier. How this field is set is addressed in section 5.3.3.

5.3.3 DyRAM Active Services

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 5.2, 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, are provided in chapter 7.

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 then the router will look for a corresponding NS structure and will send the repair on all the links that appear in the subcast list.

Replier Election

The *"replier election"* is a key functionality which is specific to DyRAM. It allows for local recovery 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 updates the corresponding LS structure when necessary. If the received NACK is valid, an NST is initialized and an 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 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 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 are 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.

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 LMS [101]. The replier link in LMS 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 providing 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 a given link, we use the following:

$$weight = \kappa\ (-RTTvar) + (1-\kappa)\ \frac{1}{RTT \times (lr - lo + 1)} \tag{5.1}$$

where κ is a weighting parameter to be chosen in $]0,1[$. $RTTvar$ varies inversely with the available bandwidth on the considered link. The RTT gives an idea on the potential replier distance to the active router. As for $(lr - lo + 1)$, it can be seen as an estimation of the loss rate experienced by the receiver located downstream this link.

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 [32] stating that 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 sends 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 will refrain from sending 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.

5.3.4 DyRAM Main Algorithms

This section presents in a synthetic manner the algorithms performed by the active routers in the multicast tree on a data and a NACK packet. RTT estimation, flow control and congestion avoidance related algorithms are not considered here.

The Data Packet Service

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

---

update the *trackList* of the session (TL)
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

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

5.4 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 in a scalable manner and secondly, present how the different timeouts are set.

5.4.1 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 5.2, one path of a multicast tree that connects the source (labeled by 0) to one of the group receivers (with 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 reception of an HB by a router triggers the emission of an HB message to its parent in the multicast tree in addition to responding with an HB_RESP

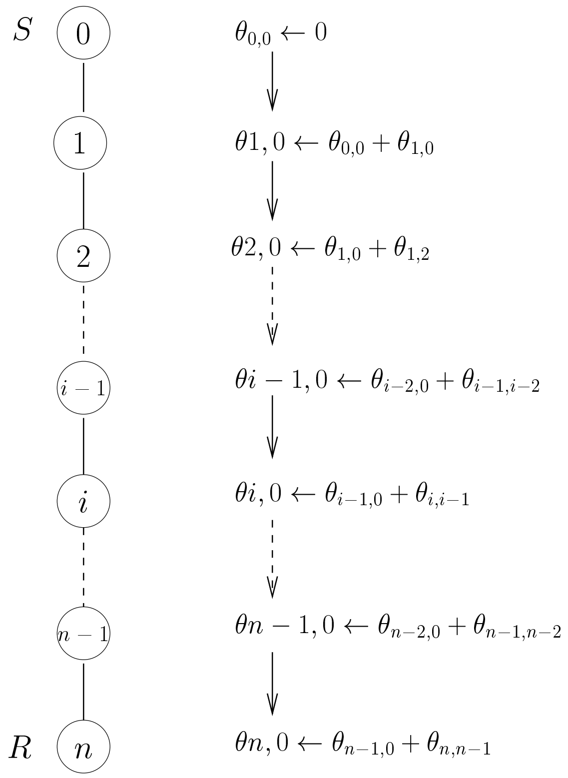


Figure 5.2: RTT estimation

message.

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 an HB message timestamped with the emission time. The parent node, On the reception of the HB message, responds with an 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:

$$\forall i > 0 : \theta_{0,i} = \theta_{0,i-1} + \theta_{i-1,i} \tag{5.2}$$

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 (5.2) 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 5.2). Afterwards the use of the HB and HB_RESP messages will be sufficient to estimate the different RTTs.

5.4.2 Timeouts Setting

If all routers at the edge of the core network are active, our protocol performs well with respect to different metrics as will be seen in the next chapter. 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 the routers are active and have sufficient memory to create an 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[3]. 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 a 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 transmission rate of the source. The current rate is known by each receiver thanks to a dedicated field in the data packet header (section 5.3.1). This field is also used by the congestion avoidance algorithm associated to DyRAM (see chapter 7).

As described in section 5.3.3, 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 NACKs from the downstream links. 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 chapter 7). $\max(RTT_{down})$ ($\min(RTT_{down})$) is the maximum (minimum) RTT between this active router and any receiver in its subtree.

[3] An EWMA filter can be used.

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 furthest 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 whose 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 $[0.1, 0.2]$.

5.5 Flow Control and Memory Usage

5.5.1 Flow Control

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 in the CR to be sent upstream to the source, the minimum $MaxSeq$ value among those reported by the different CRs received from downstream. In this way, the source ends up by receiving the minimum $MaxSeq$ reported by the different receivers. The source will send with its current transmission rate, data packets with sequence number not greater than this minimum $MaxSeq$ value.

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 the 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 given $MaxSeq$ field, would update its flow control window accordingly. 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 $MaxSeq$.

5.5.2 Memory Usage at the Routers

One of our design goals was to unload the routers from caching data packets to allow for 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.

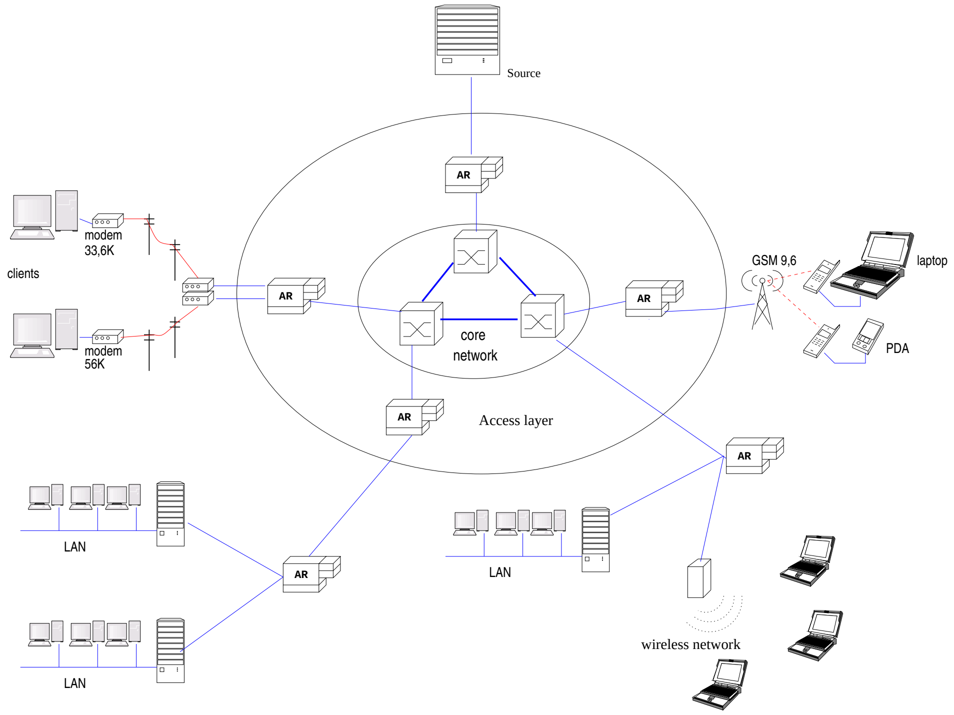


Figure 5.3: An example of an active-based reliable multicast architecture

In fact, a router maintains only one TL structure per multicast session and one LS structure per downstream link. For the NS structure, we create an NS structure per loss which is immediately 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 an 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 an NS structure until the reception of the corresponding repair is not expensive.

To prevent the fact that an NS structure has a non finite life time, this NS is removed if it corresponds to a lost repair. 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.

5.6 An Active-based Reliable Multicast Architecture

We consider the IP multicast model introduced by Deering [32]. The source sends messages to a multicast address subscribed to by all the receivers. The sender does not need to know the receivers identities. A receiver can join or leave a multicast session as it wishes by simply sending a *join* or a *leave* message via IGMP (Internet Group Management Protocol). The routers has to make their best effort to deliver the traffic from the sender to each receiver. The packets will be duplicated in the network as needed, thus reducing bandwidth consumption compared to the unicast approach. Our reliable protocol DyRAM works on top of UDP and will guarantee that all the sent packets are received by all the receivers of the multicast group.

Our architecture (see figure 5.3) is based on a virtual topology of active routers which are able to perform customized computations (services) on the messages flowing through them. 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.

All the active routers are assumed to perform at least two active services, the NACK suppression service and the subcast functionality. The active router which is located at the sender's side just before the core network is called the *source router*. As have been shown in chapter 4, the loss detection service is only beneficial if the loss detection capable-router is close enough to the source. Consequently 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 perform the replier election service. This is essential to avoid electing as a replier, a receiver which is located very far from the affected subtree.

For the realization of our active approach, we propose the *programmable switch approach* which maintains the existing packet format. Programs are injected separately from the processing of messages. Initially, and using a reliable channel, the source injects the required service by sending them to the multicast address. Afterwards, the source begins the multicast of the data packets. When an active router receives a packet, it first looks for the appropriate service deduced from the packet header. Consequently a recognized packet as a DyRAM one will be processed by the DyRAM active services that process in a customized way the different packet types.

To dynamically handle the multicast group topology changes, active routers have to be able to add or remove the concerned services. An active router that leaves the multicast session has simply to remove the associated services. However when an active router joins a multicast tree, it has to easily download the required services. This can be achieved by requesting the services from the closest active router which has already installed them. If there is no such active router then the services need to be sent from the source. The required services are always transmitted via a reliable channel. These operations are usually handled by the used active execution environment.

5.7 Implementation and Experimental Measurements

Given the choice of active networks to implement DyRAM, it is crucial to determine how costly the corresponding active services are. In this section, we report our experience in prototyping DyRAM on top of the TAMANOIR execution environment [54]. 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.

5.7.1 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 from both 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 *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 another TAN in the neighborhood. Downloading services from a service broker is also possible.

5.7.2 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 IP addresses of receivers or active routers directly connected to it.

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 5.6. Moreover such a router will forward immediately a NACK to the source instead of wasting time in a replier election procedure. 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 whose the expiration triggers the emission of the HB_RESP_S message (described in section 5.4), could be initialized. Motivations behind waiting for a given amount

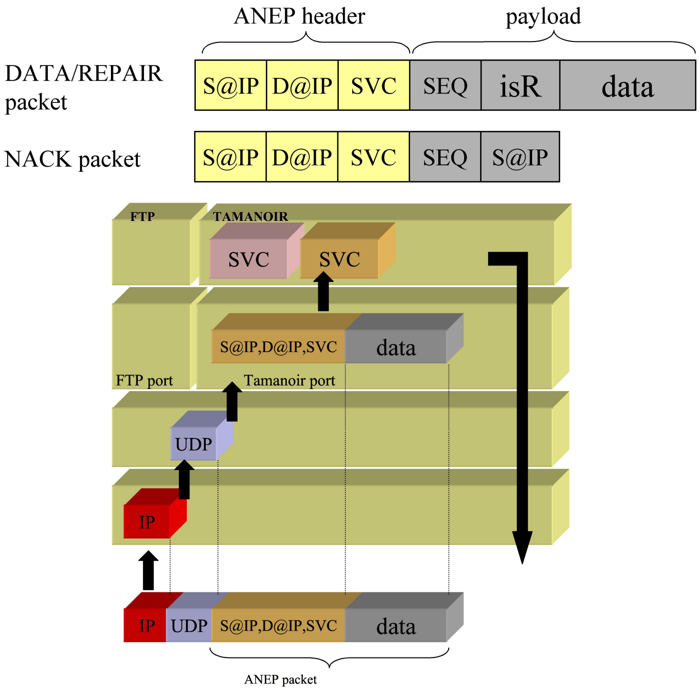


Figure 5.4: Packet format and TAMANOIR architecture.

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 can 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.

5.7.3 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 5.4 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 for simplicity.

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.

5.7.4 Test-bed and Scenarii

Experimentations were performed with a first prototype using a test-bed that consists in a set of receivers and two PC-based routers (see table 5.1 for the characteristics of the used machines) 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

Table 5.1: 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 5.5), the source `ike` multicasts packets to `stan` and `resamd` through two 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 5.5 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 5.6), 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 receivers list in order to determine a replier for the lost data packet.

5.7.5 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 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.

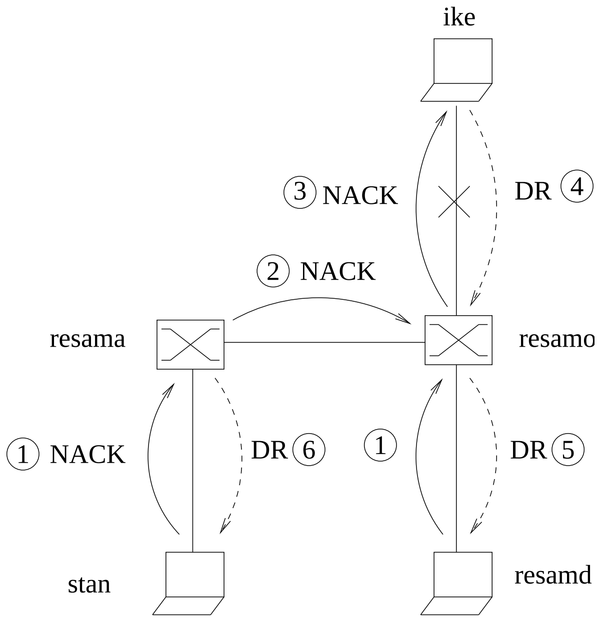


Figure 5.5: Topology 1: DP, NACK and DR services.

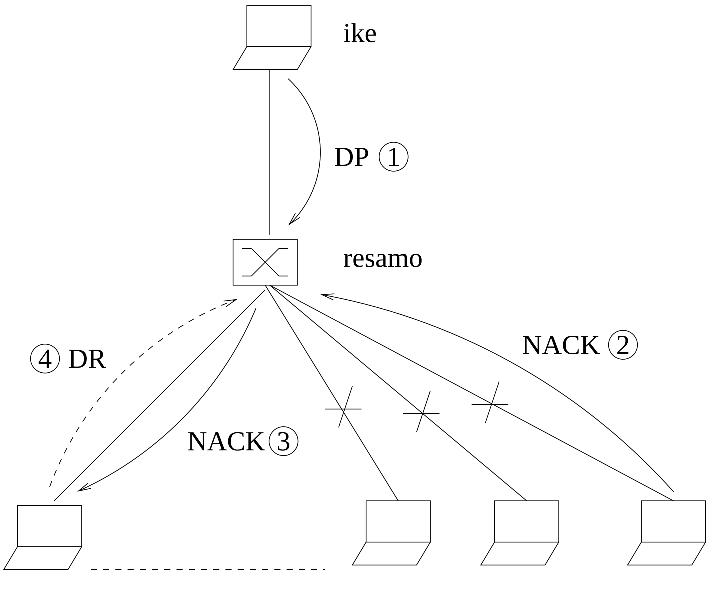


Figure 5.6: Topology 2: replier election

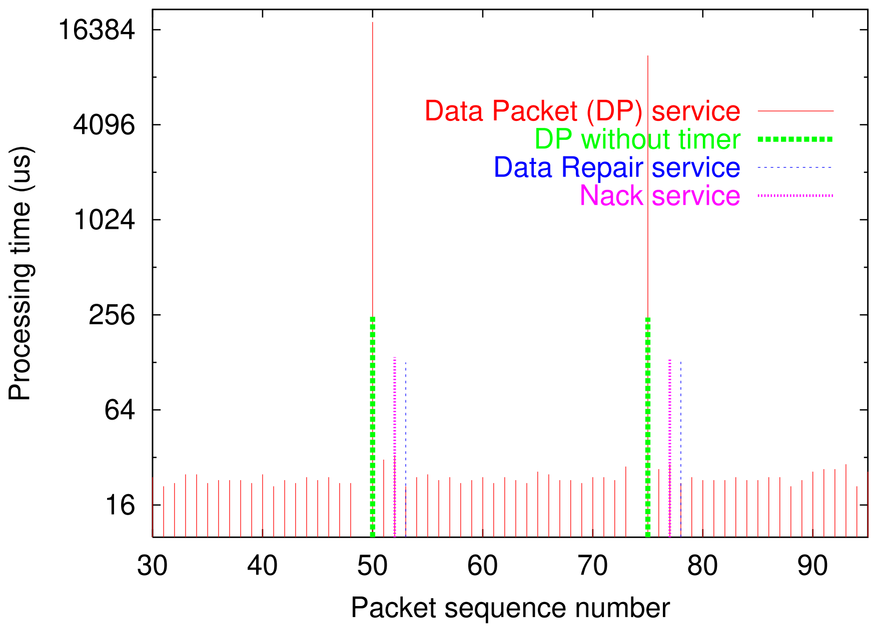


Figure 5.7: Cost of packet services

Preliminary results on our test-bed are illustrated in figures 5.7 and 5.8. Figure 5.7 shows the processing time in microseconds 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 5.7 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 (for these packets, the x-axis indicates the current data packet sequence number when they are processed in order to respect the chronological order). Varying the message size from 1KByte to 32KByte does not change the results.

Figure 5.8 shows the processing time to determine the replier dynamically . 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 on the election process: finding a replier after 4 trials amongst 25 receivers adds $30\mu s$ to the case where the replier is

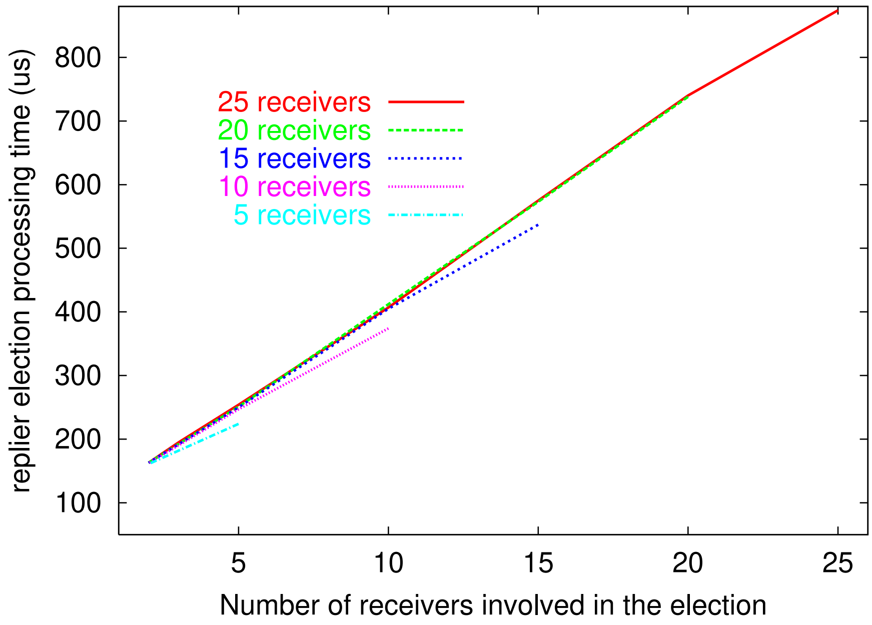


Figure 5.8: Cost of replier election.

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 the overall protocol perspective.

5.8 Conclusion

In this chapter we have reported on our experience in the design 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 data packets caching facilities. A link is elected as a replier on a per-packet basis to perform the retransmission of a lost packet. An estimation of a link's capacity is performed in order to have a fairer replier election with a nice load balancing feature among the links located downstream from a router.

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 chapter 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 to hundreds of microseconds on a Pentium II 400MHz PC-based router. Finally, it is worth noting that a real-world demonstration of DyRAM has been conducted on VTHD[4] in the context of e-toile project[5].

In this chapter, we focused on the description of the reliability mechanism adopted in DyRAM and its first implementation. An extensive evaluation using simulations is provided in the next chapter. Knowing that packets are mainly lost due to congested routers, a congestion control protocol has also been designed to strengthen our reliability mechanism (this latter will be presented in chapter 7).

[4]http://www.vthd.org
[5]http://www.urec.cnrs.fr/etoile

Chapter 6

Performance Evaluation of DyRAM

6.1 Introduction

In this chapter, we investigate in details the evaluation of the protocol DyRAM proposed in chapter 5. In order to evaluate the performances of DyRAM, two simulation models have been implemented using PARSEC (developed at UCLA [5]) and ns (network simulator [46]). The PARSEC model was the first implemented and has been used for a preliminary evaluation of DyRAM. A model for another active reliable multicast protocol ARM [74] has also been implemented in PARSEC for a comparative study with DyRAM. Recall that ARM is similar to DyRAM since it also uses the active networking technology to achieve reliability. However, the two protocols mainly differ in how the local error recovery is implemented. While ARM performs cache of data at the routers for potential retransmissions, DyRAM performs a receiver-based local recovery.

Latter on, another model of DyRAM has been realized with ns which has gained wide acceptance among the network community. The reason behind the move to ns is mainly to be able to evaluate our congestion avoidance algorithm (described in chapter 7). Particularly, ns allowed us to evaluate the fairness toward TCP (as will be seen in chapter 7) since many versions of TCP are available. For the purpose of the performance evaluation of DyRAM (only the reliability mechanism without any congestion control), the ns model has allowed for a comparison study with another reliable multicast protocol SRM [52]. SRM implements the local recovery from the receivers using a timer-based approach and is described in chapter 2. It is worth mentioning that other reliable multicast protocols are also implemented using ns, however we were not able to perform comparisons with them. This is due to the fact that they either are not or no longer available on the Internet like AER [64] or do not integrate all the functionalities like PGM [118] whose the implemented model does not yet integrate the recovery from the receivers.

In this chapter, other protocols are also analyzed for a comparative study with DyRAM. One protocol is noted S, a native reliable multicast protocol which does not benefit from active processing at the routers. In S, there is no local recovery, all the retransmissions are performed by the source and additionally there is no feedback suppression at the routers. We also consider protocols A and D of chapter 4 which are two active-based reliable multicast protocols that benefit from the NACK suppression and the subcast services. Additionally,

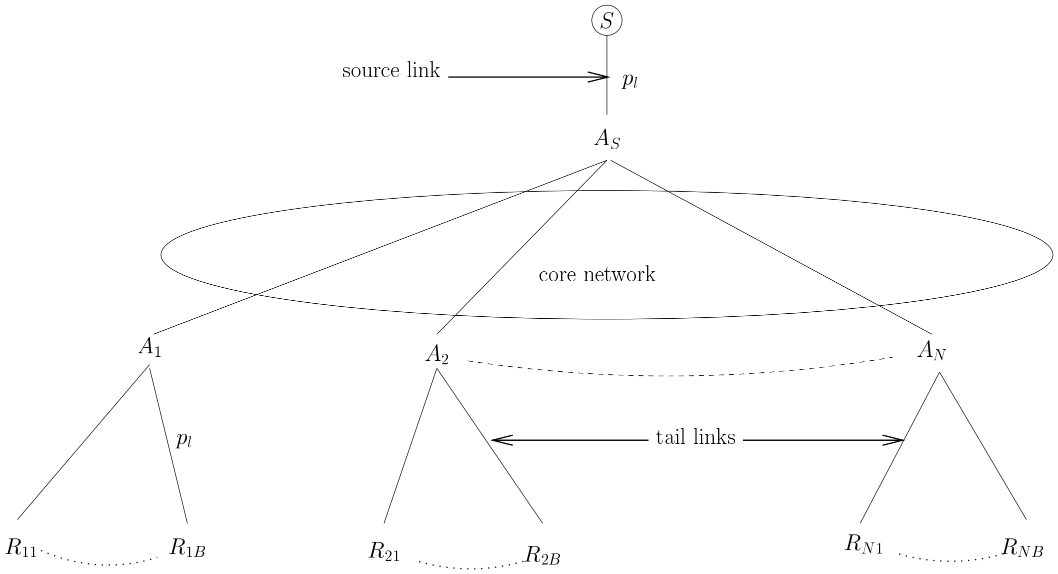


Figure 6.1: Network model.

D benefits from the loss detection service at the routers. Furthermore, we will designate by DyRAM, our basic DyRAM protocol without the loss detection at the routers, and by $DyRAM^+$ the enhancement of DyRAM with the loss detection capability. In all the simulations, the replier election is performed with respect to a ring order without any optimization.

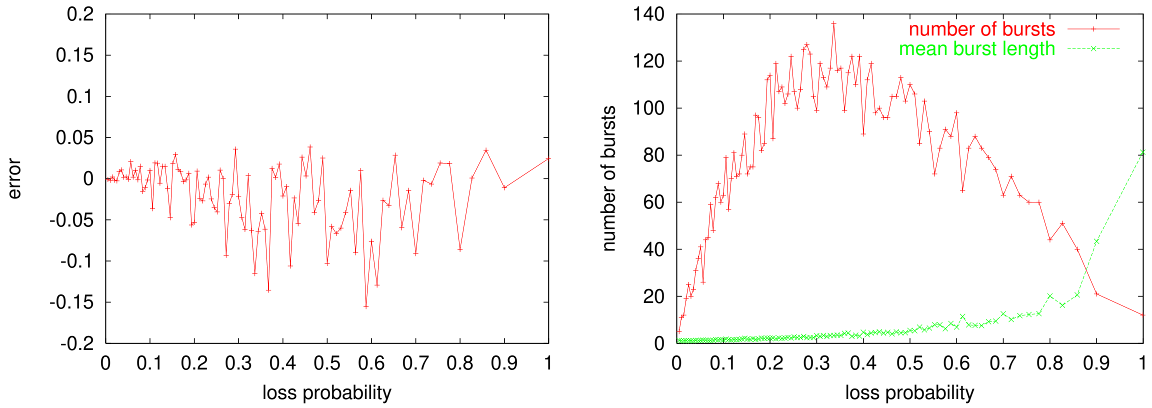
The remainder of this chapter is organized as follows. First, the PARSEC simulation results are presented as preliminary ones in section 6.2. Mainly, DyRAM is compared to ARM and the main active services are evaluated. Afterwards a deeper analysis is conducted in section 6.3 where DyRAM is compared to protocols S and SRM. Many aspects are dealt with such as the impact of the routers density, the load balancing at the receivers and the number of the receivers subscribed to the multicast session.

6.2 Preliminary Evaluation

This first set of simulations are performed using the model realized in PARSEC. We first present the network and loss model we adopted. Then, we define a set of metrics before presenting the preliminary results. At the beginning, simulations are performed on a basic DyRAM protocol without the loss detection service. Firstly, we will show the benefit of performing the local recovery from the receivers. Then a comparison between our approach (no cache in routers) and ARM[74] will be presented. We will also consider the case where DyRAM also benefits from some cache amount in routers in the multicast tree. Finally, we introduce the loss detection service in our evaluation.

6.2.1 Network Model

In order to perform this first set of simulations, we consider a packet switching network composed of a fast core network and several slower edge access networks. We have one source



(a) (b)

Figure 6.2: The temporal correlation.

that multicasts data packets to R receivers distributed among N local groups of equal size $B = R/N$. A local group is composed of the set of receivers $\{R_{i1}, R_{i2}, \ldots R_{iB}\}$ located downstream an active router A_i, $i = 1..N$ (figure 6.1).

We consider that the routers located at the edge of the core network, are active, so they will be able to execute the DyRAM related services[1]. The router located at the source side (A_S) is called the *source router* and the others located at the receivers side are designated by *tail routers*. All the active routers are assumed to perform at least two active services, the NACK suppression service and the subcast functionality. The replier election is only performed by the tail routers while the loss detection is only performed by the source router according to our proposed architecture 5.6.

We assume that there are l_b backbone links between the source and every tail router A_i. 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 overhead at the routers, we assume that both NACKs and data packets are processed in 32 time units. These values are derived from measures in [74]. In this simulation model, we have taken into consideration the fact that the repairs may also be lost.

6.2.2 Loss Model

in order to simulate losses in our network model, we considered 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 to be reliable as mentioned previously. For the other links (the source link and the tail links), the loss probability is noted p_l which gives an end-to-end loss rate perceived by a receiver $p = 1 - (1 - p_l)^2$. The losses at the tail links are assumed to be mutually independent.

[1]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.

To take into consideration the temporal correlation of losses, we use the Markov chain model proposed in [132]. Let M be the transition matrix $M(i,j) = P[j/i]$ 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 [132], 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:

$$M_\lambda = \frac{1}{\lambda} \, . \, M$$

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

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

where $a = 0.8068$ and $b = 0.1768$.

To evaluate our proposed loss model, we plot in figure 6.2a, the error (y-axis) incured when trying to achieve a given loss rate (x-axis). We can see that with this loss model, we obtain in most cases the targeted loss rate. In the Internet, losses tend to appear in bursts whose length increases with the loss rate. Figure 6.2b plots the number of bursts and their mean length as a function of the targeted loss rate. We can see that the mean burst length increases with loss rate. This is a reproduction of losses due to congestion in the network.

6.2.3 Metrics

For this set of simulations, we have defined the following 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.

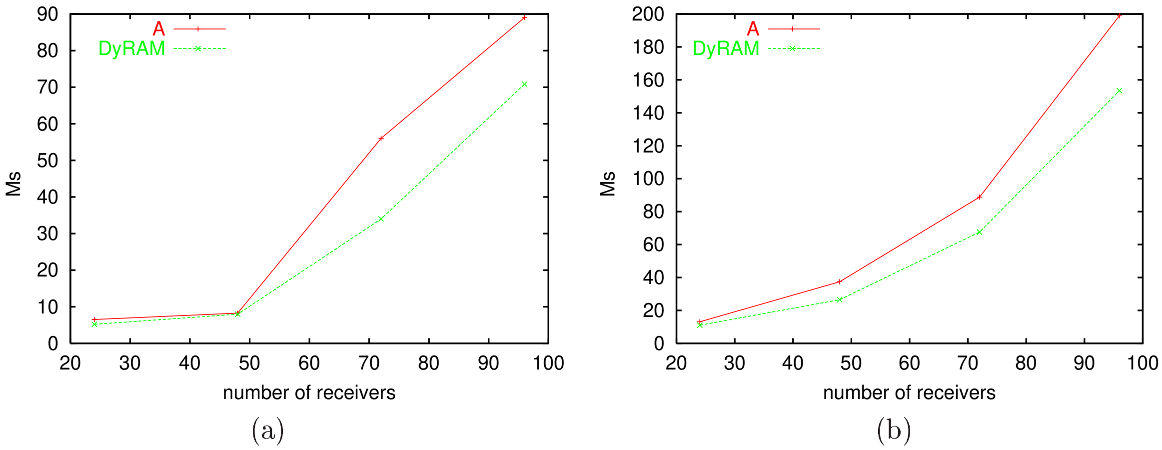


Figure 6.3: Load at the source in DyRAM and A (a) $p = 0.05$, (b) $p = 0.25$.

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 verified that the three metrics defined above behave in the same manner and are dependent on each other. This is why, we will not show all the curves we obtained.

6.2.4 Local Recovery from the Receivers

In order to evaluate the impact of the recovery from the receivers instead of doing that from the source, figure 6.3 plots the number of retransmissions (M_S) from the source in DyRAM and A as a function of the number of the receivers for 5% and 25% loss rates. These results have been obtained from simulations of a multicast session with 4 receivers per local group where the number is varied to obtain different numbers for the session participants. The curves show that the local recovery allows for less load at the source as the number of the 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 recovery 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 at each time. Figure 6.4 compares the per-link consumed bandwidth ratio of DyRAM to another approach without any local recovery facilities (A) 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 6.4 shows that with only 6 receivers per group and local recovery we achieve a gain of 80%.

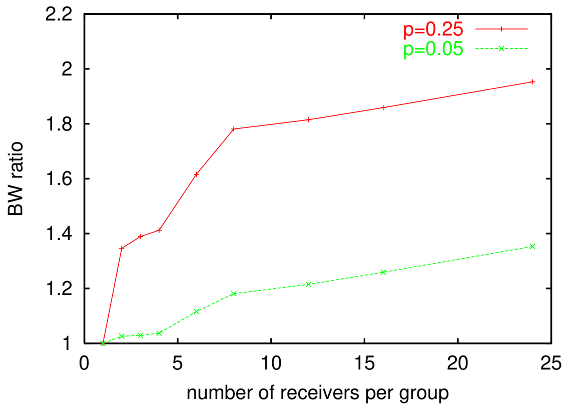


Figure 6.4: Consumed bandwidth ratio in DyRAM with respect to A as a function of the group size.

6.2.5 Receiver-based vs. Router-based local recovery

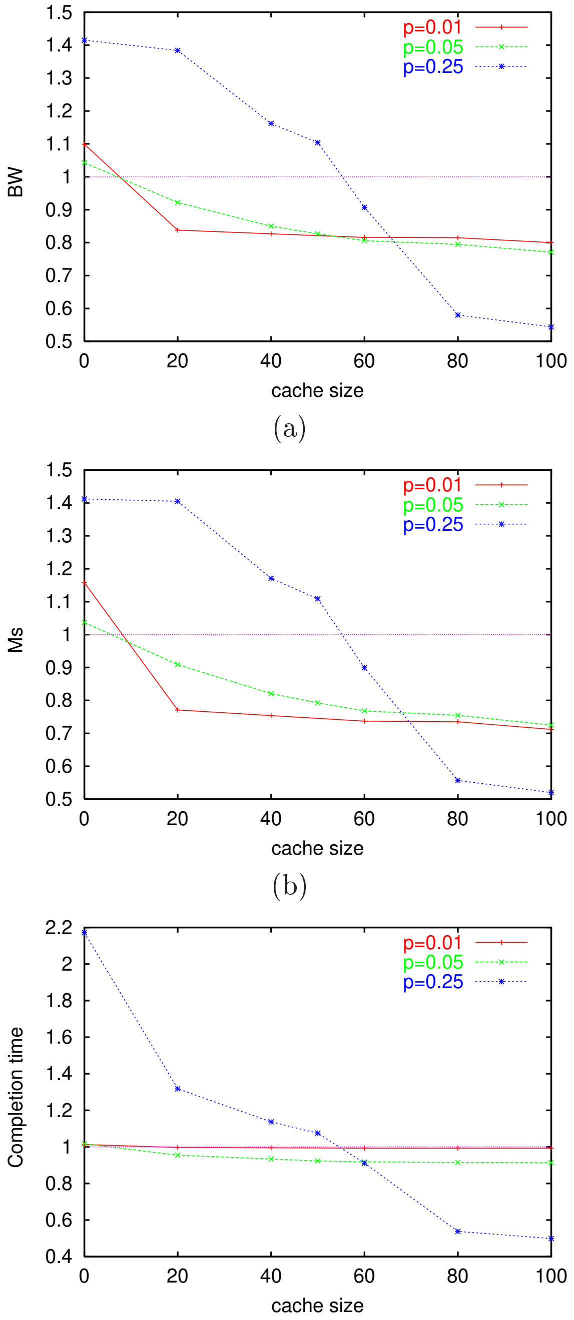
In this section, we compare our approach to ARM. Figure 6.5 shows for different loss rates, the ratio of the consumed bandwidth (BW) (load at the source M_S and the completion time) in ARM to DyRAM. We took into consideration the percentage of the available cache at the ARM routers[2]. This is due to the fact that caching means at a router are limited. We can see that ARM performs better than DyRAM 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 DyRAM still has better performances than ARM when the latter can use more than 50% of cache.

6.2.6 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. The purpose of that is to see if DyRAM can outperforms ARM in the case of low loss rates with a minimum amount of cache at the routers.

Figure 6.6 plots for different loss rates, the bandwidth gain achieved by ARM and DyRAM when the cache size is varied, over a simple active protocol A where all the retransmissions are performed by the source. 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.

[2]For instance, a percentage of 10% means that the available cache is of 10% of the size of the file to be multicast



(a)

(b)

(c)

Figure 6.5: DyRAM vs. ARM (a) Consumed bandwidth, (b) Load at the source, (c) Completion time.

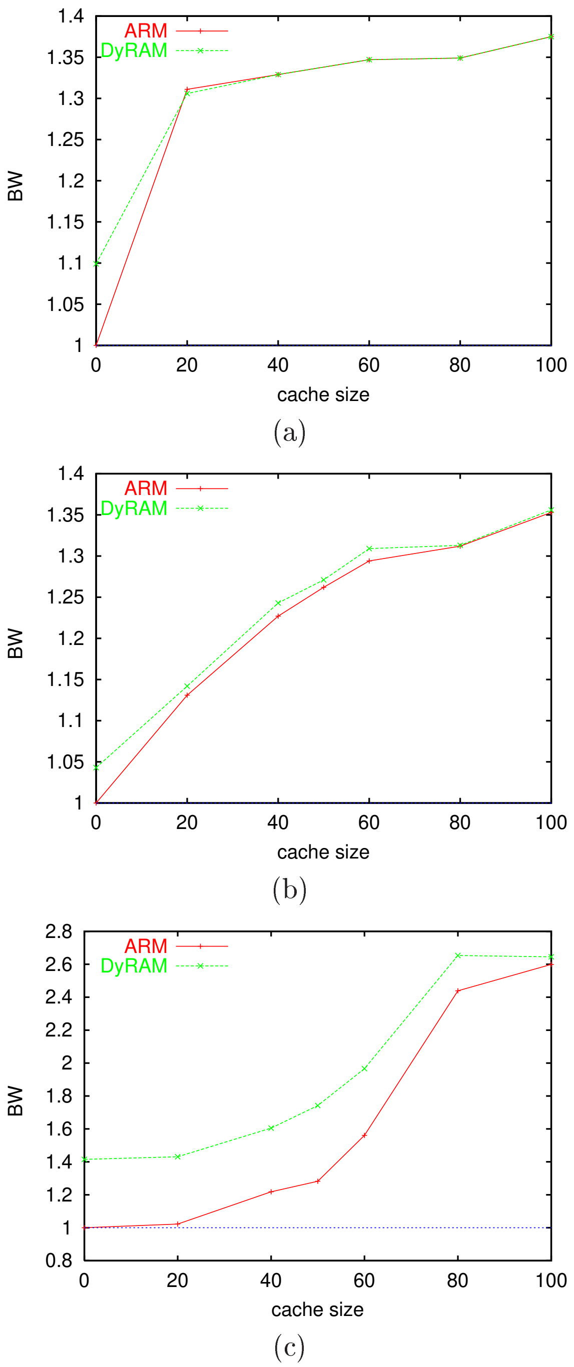


Figure 6.6: Gain obtained by DyRAM and ARM when both benefit from cache in the routers (a) $p = 0.01$ (b) $p = 0.05$, (c) $p = 0.25$.

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.

6.2.7 Adding the Loss Detection Service to DyRAM

Here we will consider the loss detection service in our simulations. A set of simulations are performed to show how a loss detection service could decrease the delay of recovery in DyRAM. Figure 6.7 plots the recovery delay (normalized to the RTT) for the four protocols A, D, DyRAM and $DyRAM^+$ 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 6.7a that DyRAM is 10 times faster than A for a loss rate of 5%. Now, when the loss detection service 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 decreases 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 6.7a-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.

6.3 Further Analysis

In this section, ns is used to evaluate and analyze DyRAM. The network model used in ns is similar but different from the PARSEC one in some points. This network model is described in section 6.3.1 where the different scenarii and metrics used are introduced. Then a set of simulation results are presented. First, DyRAM is compared to a non active-based reliable multicast protocol S showing the benefit of the active routers contribution as well as the impact of their density in the multicast tree. Afterwards, DyRAM is compared to SRM where we mainly consider the load at the different nodes in the multicast tree. Large-scale simulations are also conducted to evaluate the scalability of our approach.

6.3.1 Network Model and Metrics

The network model adopted here is similar to the previous one (figure 6.1) with some differences. The number of the receivers per each group is chosen randomly following a uniform distribution of parameters B_{min} and B_{max}. Furthermore, the bandwidth and the delays associated to each link are also chosen randomly. In such a way, the receivers of a local group are not of equal distance from their active router and have different capacities. In this network model, a router can be configured as active or passive. This is interesting when considering

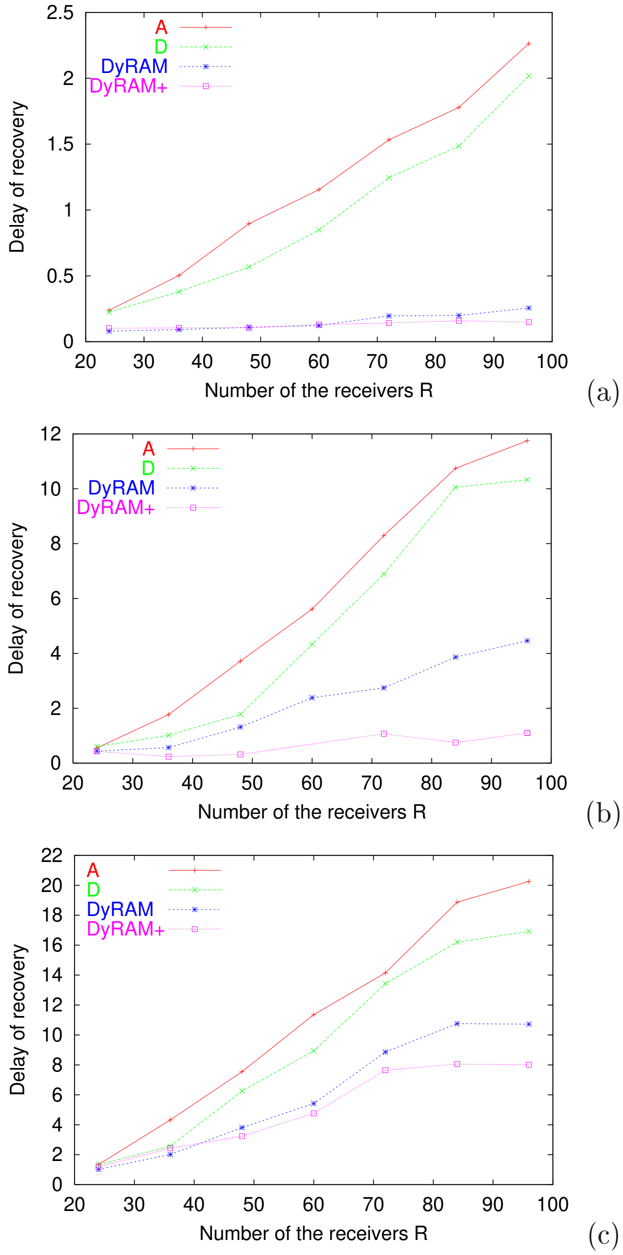


Figure 6.7: The recovery delay with (a) $p = 0.05$, (b) $p = 0.25$ and (c) $p = 0.50$.

the impact of active routers density on the performances of DyRAM.

In order to generate losses at the different lossy links, a loss module is assigned to every tail and source link with a loss rate of p_l. We consider, three different patterns of losses. The first pattern consists in correlated losses introduced at the source link. The second pattern produces losses only at the tail link. The third pattern combines the two previous patterns where losses occur at both the source link and the tail links. The transmission rate of the source is fixed appropriately so losses due to congestion are avoided. The congestion avoidance algorithm associated to DyRAM is not considered here. It is described and evaluated in the next chapter.

All the simulations unless stated otherwise, are run for 200 seconds with a multicast file of an unlimited size. Losses are introduced at 10s and are stopped at 110s. B_{min} and B_{max} are set respectively to 1 and 20. All the routers are configured to operate actively and to execute the different active services. According to our proposed architecture (section 5.6 of the previous chapter), all the routers perform NACK suppression and CR aggregation. In addition, the source router performs the loss detection service while the tail ones perform the replier election service.

The metrics used in this second set of simulations are the following:

Repair Latency

The repair latency for a lost packet is the time elapsed between the emission of a request and the reception of the first repair normalized to the RTT between the affected receiver and the source. In our results, a mean of the repair latency is computed among all the losses experienced by all the receivers.

Load at the Different Nodes

For each type of node (source, receiver or router), the amount of input and output bytes per unit of time is recorded during the simulation. This gives an idea on the load experienced by each node in the network model. This is useful for instance, to have an idea on the source load and the contribution of the receivers in the recovery process.

Repair Implosion

The repair implosion is quantified by computing a mean among the duplicate repairs per packet obtained by dividing the number of repairs received by all the receivers on the number of NACKs sent for this data packet.

6.3.2 Benefit of Active Routers

The purpose of this section is to show quantitatively the benefit of the existence of active routers in DyRAM. To do so, simulations are conducted on DyRAM and S during 200 seconds for different loss scenarii.

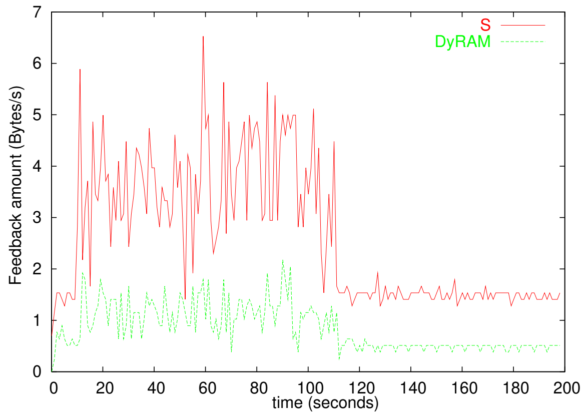


Figure 6.8: Feedback at the source in S and DyRAM, $p_l = 0.10$ with losses at the source link.

Feedback Suppression

Figure 6.8 plots for a loss rate of 10% per link, the input load (corresponding to the feedback messages) at the source where the losses occur only at the source link. In this case, there is no local recovery since a loss affects all the subscribed receivers. We can see that the source in S receives more feedback than in DyRAM. In the absence of losses (before 10s and after 110s). This is due to the fact that the CRs sent periodically by the receivers in S are not aggregated by the routers before reaching the source. When losses occur (between 10s and 110s), we note that the amount of feedback received by the source increases in both S and DyRAM due to the transmission of NACKs by the affected receivers. We can deduce from figure 6.8, that the source receives less NACKs in DyRAM compared to S. This reflects the NACK suppression performed by DyRAM at the active routers before they reach the source. In summary, the source is unloaded in DyRAM from processing multiple feedback sent by the receivers. For instance, the amount of feedback is reduced by a factor of approximately 4 and 3 in respectively the presence and the absence of losses.

Local Recovery

Figure 6.9 shows the input load at the source when losses occur at the tail links. This figure shows the benefit of the loss recovery from the receivers instead of the source. Since all the losses are experienced at the tail links, most of the losses in DyRAM can be recovered locally. Indeed, the source in DyRAM as we can see in figure 6.9, is unloaded during all the simulation. However, in S the source is overwhelmed by feedback messages that mainly consist in requests from the receivers affected by losses. These feedback are not suppressed at the routers as in DyRAM. We can see that the source examines a load multiplied by a factor of 90 compared to the case of DyRAM.

Figure 6.10 shows the amount of feedback the source receives when losses occur at both the source and tail links. Once again, the source is unloaded in DyRAM thanks to the active services such as CR aggregation, NACK suppression and replier election. Furthermore, we can notice that all the losses that occurred are recovered earlier in DyRAM compared to S. Losses are recovered at 115s in DyRAM instead of 180s in protocol S.

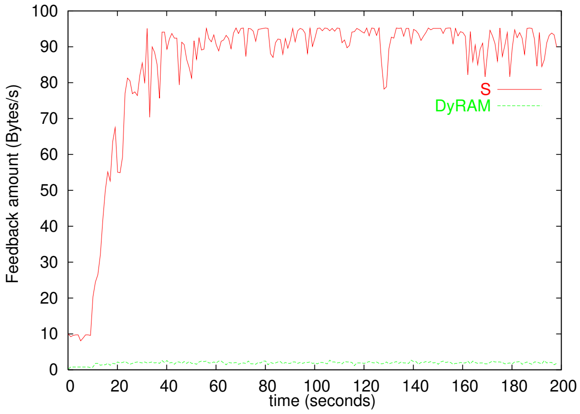


Figure 6.9: Feedback at the source in S and DyRAM, $p_l = 0.10$ with losses at the tail links.

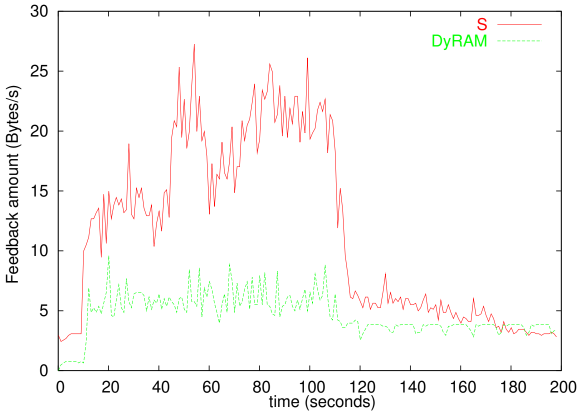


Figure 6.10: Feedback at the source in S and DyRAM, $p_l = 0.10$ with losses at all the links.

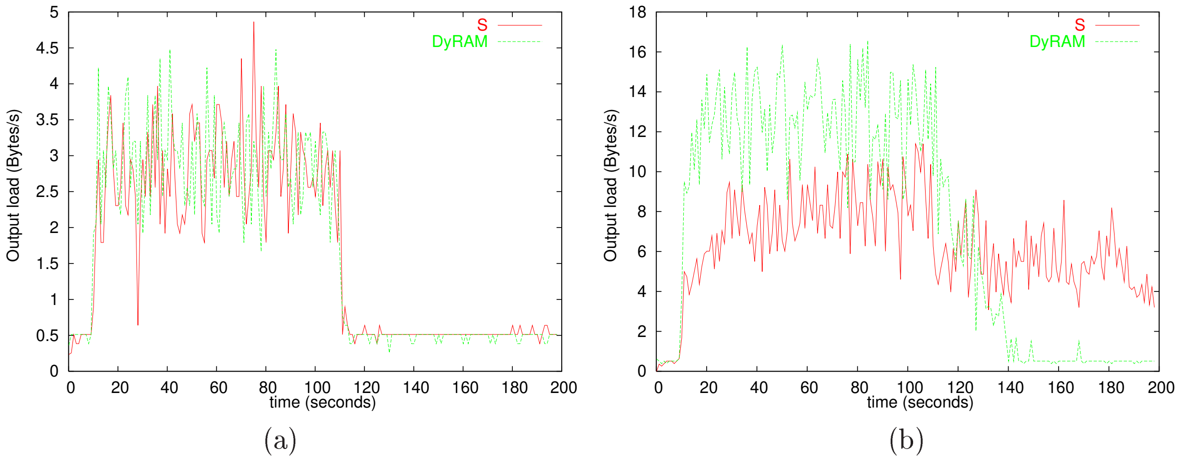


Figure 6.11: Output load at a receiver in S and DyRAM, $p_l = 0.25$ with losses at (a) the source link, (b) all the links.

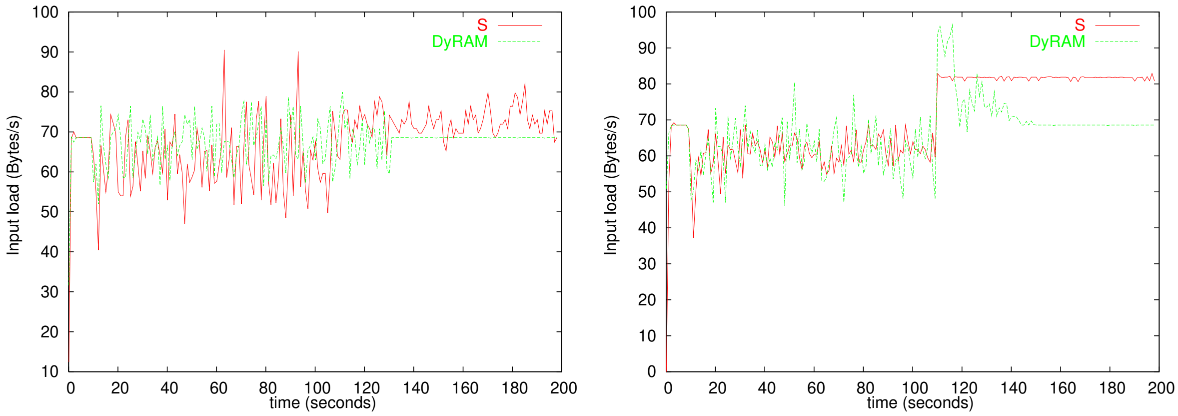
Load at the Receivers

Figures 6.11 and 6.12 plot respectively, the evolution of the output and input load at a receiver in DyRAM and S for a link loss rate of 25%. When losses occur at the source link, we can see in figure 6.11a that the two protocols are equivalent in terms of the amount of feedback sent by the receivers. In such a case there is no local recovery from the receivers which in both S and DyRAM, have similar behavior when sending CRs and NACKs to the source. When losses occur at all the links, we can see in figure 6.11b that a receiver in DyRAM sends out more bytes in the period of losses. This additional traffic from a receiver in DyRAM consists in the repairs sent by this receiver. However, this additional traffic is no longer present at the receivers side. Thanks to the local recovery, losses are rapidly recovered. In S, we can observe that all the losses are not recovered even by the end of the simulation.

With respect to figure 6.12a, we can see that once again, S and DyRAM have similar behavior in terms of the input load at the receivers when the losses occur at the source link. However, when losses occur also at the tail links as shown in figure 6.12b, a receiver in S has no longer the same behavior as in DyRAM. We can see that after the loss period, there is more load at the receivers in DyRAM since they continue receiving repairs and NACKs in addition to original data packets without losses. This will no longer be the case and losses are recovered more quickly in DyRAM than in S. In this latter, we can observe that a receiver continues receiving repairs until the end of the simulation.

6.3.3 Active Routers Density

In this section, we consider the density of the active routers (those located at the receivers' side) on the performance of DyRAM. In order to be able to have the largest multicast group, the number of data packets sent by the source is limited to 200 packets instead of the unlimited number for the previous simulations. Losses occur during all the simulation time and the simulation ends when all the receivers have received all the 200 packets sent by the source. The simulations are run for different values of $N = 1, 2, 3, 4, 8, 20, 40$. The number of the



(a) (b)

Figure 6.12: Input load at a receiver in S and DyRAM, $p_l = 0.25$ with losses at (a) the source link, (b) all the links.

receivers per router is chosen randomly from a uniform distribution of parameters 1 and 20. A number of simulations is performed and a mean is computed for experiences with the same number of receivers.

Figure 6.13 plots for different loss rates and scenarii, the repair latency of lost data packets for different densities (0, 50 and 100%) of active routers. First, for low loss rates say 1% as depicted in figures 6.13a1-a2, we note that when the number of the receivers is small, there is no noticeable difference between an active (with 50 or even 100% of active routers) and a traditional approach in terms of repair latency. When the number of the receivers increases, we note the increasing benefit of the existence of active routers in the multicast tree. For instance, for more than 400 receivers, we can examine a repair latency reduced with a factor of 1.8 when 50% of the routers are active compared to the passive approach (protocol S). When all the routers are active, the repair latency is improved with a factor 2.5 and 4.25 compared to the case of 50% of active routers and the passive approach respectively. When the loss rate increases (figures 6.13b-c) and especially in the presence of a large number of receivers, we can easily note the benefit of increasing the number of active routers in a reliable multicast session. We observe a repair latency reduced by a factor 2 to 5 when a passive approach (S) is replaced by a fully active approach (DyRAM with 100% of active routers).

6.3.4 Comparing DyRAM to SRM

DyRAM implements local recovery from the receivers instead of overwhelming the source by multiple retransmissions. SRM[52] is one of the first protocols that moved the responsibility of loss recovery to the receivers. This section is an attempt to compare our approach DyRAM to SRM mainly, in terms of the load at both the routers and the receivers. We mainly consider the load balancing at the receivers and the impact of the number of the receivers on the performances. In what follows, we only consider losses at the tail links since the purpose is to evaluate the local recovery from the receivers instead of the source.

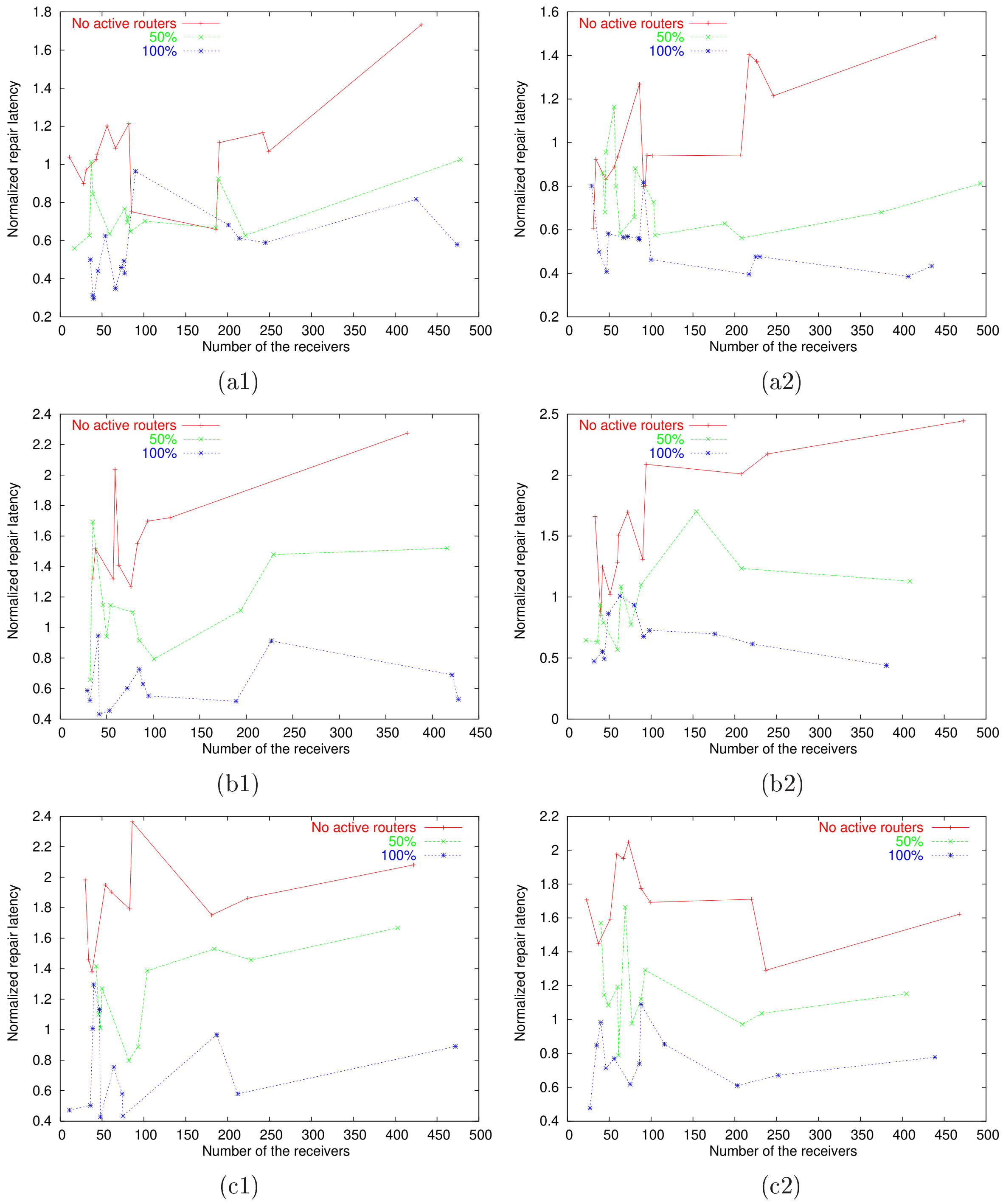


Figure 6.13: Influence of the active routers density in DyRAM on the repair latency, (a) $p_l = 0.01$, (b) $p_l = 0.05$, (c) $p_l = 0.1$. Losses at the tail links (left side) and losses at all the links (right side).

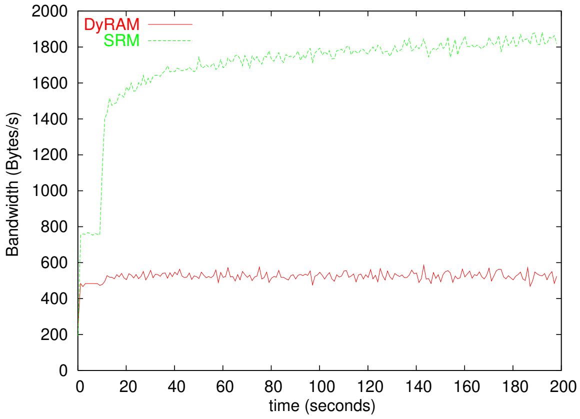


Figure 6.14: Consumed bandwidth by a tail router in SRM and DyRAM, $p_l = 0.1$.

Load at the tail routers

Figure 6.14 plots the bandwidth consumed by a tail router in DyRAM and SRM when the loss rate per link is 0.1. We can see that in the absence of losses (before 10s), the amount of consumed bandwidth by a tail router in SRM and DyRAM is almost constant. We observe that SRM has more load at the tail routers than DyRAM. This is due to the amount of signaling messages generated within SRM. When losses occur, a tail router becomes very loaded in SRM compared to DyRAM. In SRM, NACKs and repairs are sent on the multicast address to all the participants and hence received by all the routers of the multicast tree. In DyRAM, a NACK is sent toward the source and is intercepted by the routers located in its path. These routers process NACKs they receive in order to suppress duplicate ones. Moreover, in DyRAM, a repair from the source or any receiver is sent only on the affected links of the multicast tree thanks to the subcast mechanism. In SRM, the additional load at the routers is due to the reception of the different NACKs sent by all the affected receivers and the multiple receptions of repairs for the same loss.

Load at the receivers

Figure 6.15 shows the input load at a receiver for a loss rate of 25% per link. The input load of a receiver is the amount of data per second it receives during the multicast session. Mainly, a receiver receives repairs in addition to original transmissions. Due to local recovery from the receivers, a receiver can receive requests (NACKs) from other receivers in the multicast session. We notice in figure 6.15, that in the absence of losses, a receiver in DyRAM would have similar input load as in SRM. However when losses occur, an SRM receiver is more loaded than a DyRAM one, except for a short period that follows the losses interval as can be observed in figure 6.15. This shows the earlier reception of repairs by a receiver in DyRAM while it continues the reception of original transmissions without losses. An SRM receiver receives more requests compared to DyRAM because of duplicate NACKs received from the other members of the group. The load at a receiver in SRM is not reduced even at the end of the simulation. This means that losses are not completely recovered.

Figure 6.16 shows the output load in DyRAM and SRM at a receiver for a loss rate of

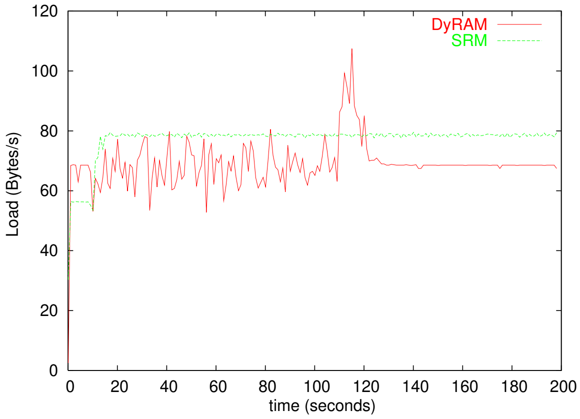


Figure 6.15: The input load at a receiver in SRM and DyRAM, $p_l = 0.25$.

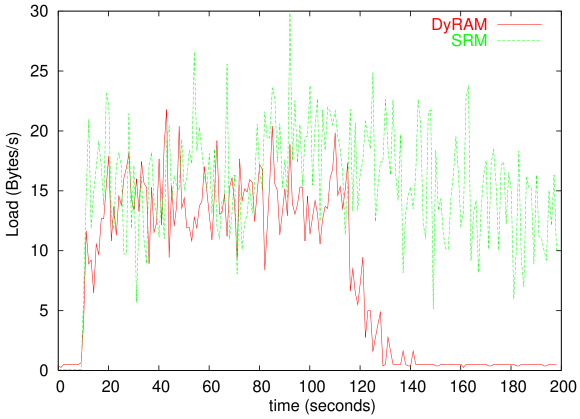


Figure 6.16: The output load at a receiver in SRM and DyRAM , $p_l = 0.25$.

25% per link. The output load is the amount of bytes sent by a receiver in the two considered protocols. A receiver sends feedback messages (mainly NACKs) and to implement local recovery, can send repairs for NACKs it receives. Once again, we notice that in the absence of losses, a receiver in DyRAM would have similar input load as in SRM. During the lossy period, the two protocols are equivalent in terms of the amount of bytes sent by a receiver. Each receiver sends requests for missing data packets in addition to transmitting repairs when requested by other receivers. We can also observe that losses are recovered much earlier in DyRAM. In SRM, a receiver continues to be overloaded by sending requests for packets it has not received yet in addition to retransmitting repairs for duplicate NACKs received from the other members of the group.

Load Balancing at the Receivers

In DyRAM, a per-packet replier election is performed by the routers. This is a nice feature, since a load balancing among the receivers of the same local group, is possible. Here, we aim to show how the retransmission load is distributed among the receivers in DyRAM. Figure 6.17 plots for DyRAM (left side) and SRM (right side), the input, output and overall load at three different receivers for a loss rate $p_l = 0.01$. First, we note that there is more load on the receivers in the lossy interval (10s-110s). In terms of load balancing at the receivers, we observe that in both of the two protocols, the different receivers have the same amount of data coming in or going out. This shows that for a low loss rate, the two compared protocols (SRM and DyRAM) are equivalent in terms of load balancing among the receivers.

To see the impact of higher loss rates on the load balancing in the two protocols, figure 6.18 plots for a loss rate of 25% per link instead of 1%, the input, output and overall load at three different receivers. We observe that in DyRAM (left side), the different loads are the same at the three receivers. However, we observe in SRM (right side) that the three considered receivers present different loads. As we can see in figure 6.18a2, there is one node that receives more NACKs (about 12.5% of additional NACKs) and thus retransmits more repairs (about 100% of additional repairs) than the other nodes as depicted in figure 6.18b2.

Scalability Evaluation

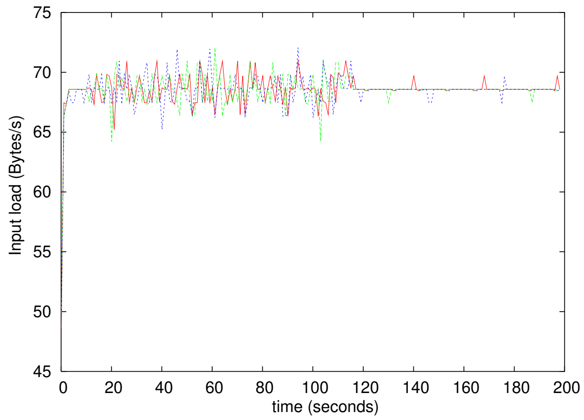
Here, we are interested in evaluating the scalability of DyRAM to a large number of receivers. Simulation scenarii performed on DyRAM and SRM are the same as those described in section 6.3.3 where simulations are run until a 200-packet file is correctly received by all the receivers. Remember that losses are generated during all the simulation time.

Figure 6.19 plots for different loss rates, the repair latency achieved by the two protocols as a function of the number of the subscribed receivers to the multicast session. We observe that DyRAM achieves a repair latency less than 1, thus less than the RTT to the source. This still being true even when increasing the number of the receivers which demonstrates the scalability properties of DyRAM. However, the repair latency in SRM is at least about twice the RTT to the source and increases with the number of the receivers to achieve ten times the RTT to the source when the number of the receivers exceeds 200.

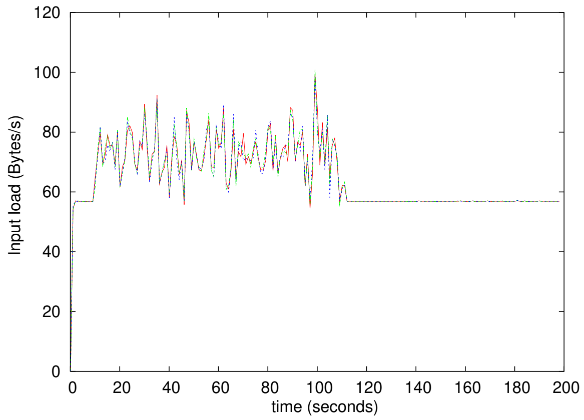
Figure 6.20 plots for different loss rates, the repair implosion experienced by the two protocols when increasing the number of the receivers. We can observe that the repair implosion is always greater than 1 in SRM while it is less than 1 for DyRAM. More interesting for low losses (say 1% and 5%), the repair implosion decreases in DyRAM when the number of the receivers increases. This is due to the fact that more receivers are present per local group and chances to recover losses locally are bigger since loss rates are sufficiently low.

6.4 Conclusion

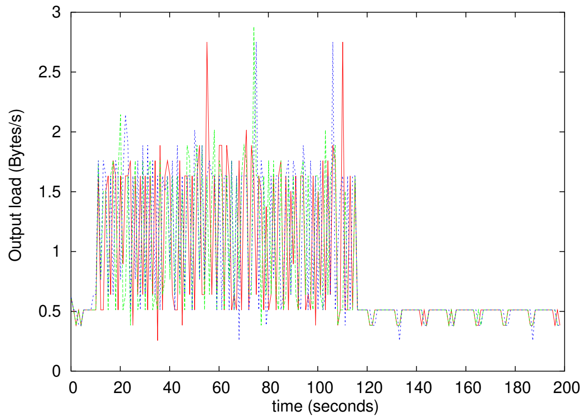
In this chapter, we presented an extensive simulation-based evaluation of DyRAM where only the reliability mechanisms are considered. The associated congestion control protocol is dealt with in the next chapter. Two models have been implemented using PARSEC and ns. In both of the two models, we tried to reflect the real characteristics of the Internet. We proposed a loss model where both spatial and temporal correlation are considered. Different loss scenarii are considered in the simulation experiences mainly in the ns model. Other reliable multicast protocols like ARM have been also implemented in order to be compared to DyRAM.



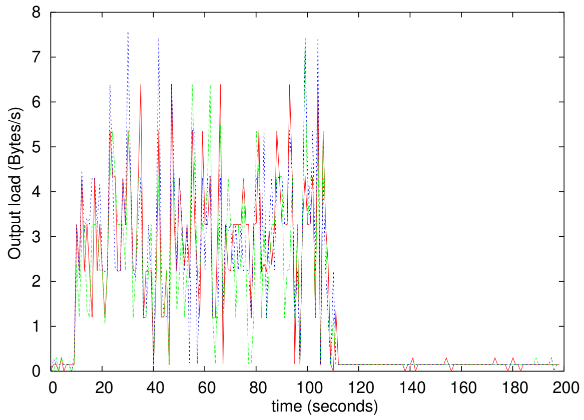
(a1)



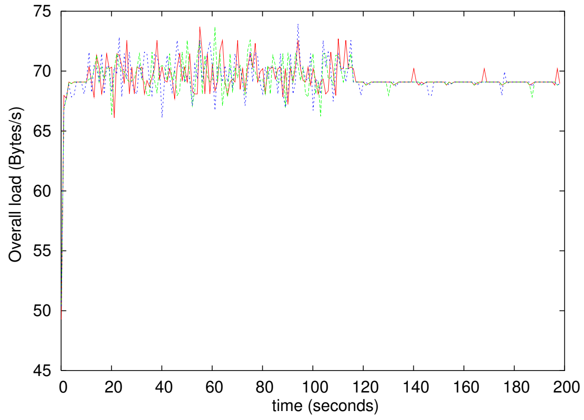
(a2)



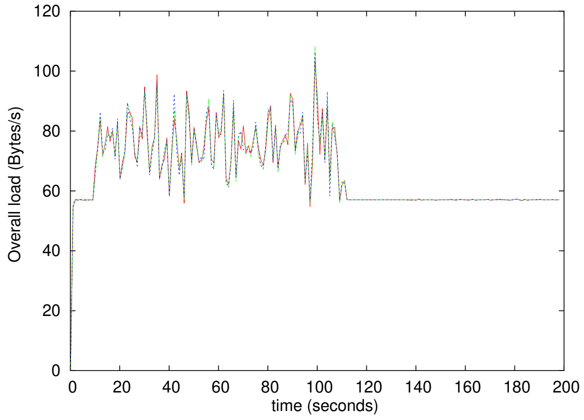
(b1)



(b2)



(c1)



(c2)

Figure 6.17: Load Balancing in DyRAM (left) and SRM (right), $p_l = 0.01$ (a) input, (b) output and (c) overall load.

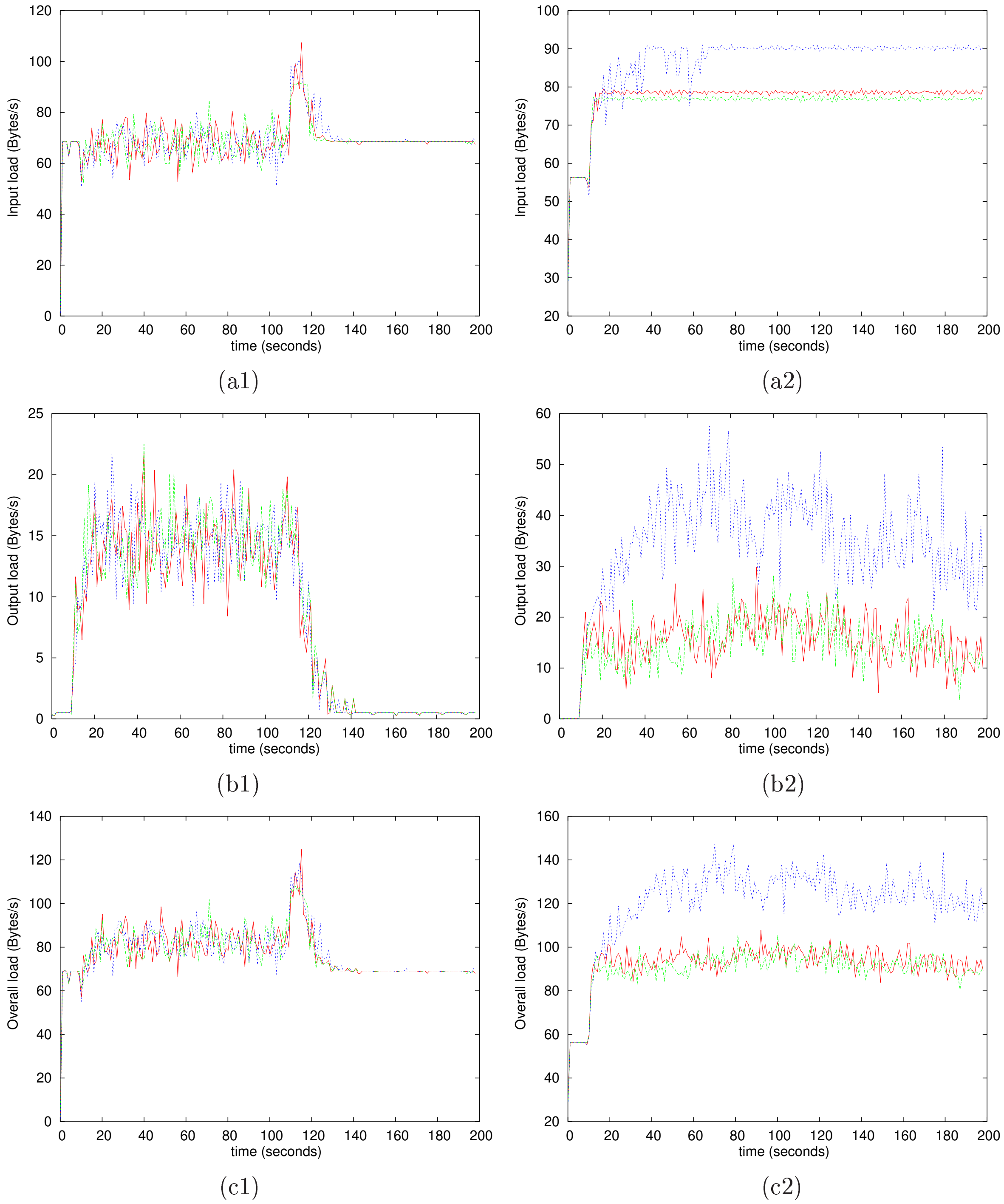


Figure 6.18: Load Balancing in DyRAM (left) and SRM (right), $p_l = 0.25$ (a) input, (b) output and (c) overall load.

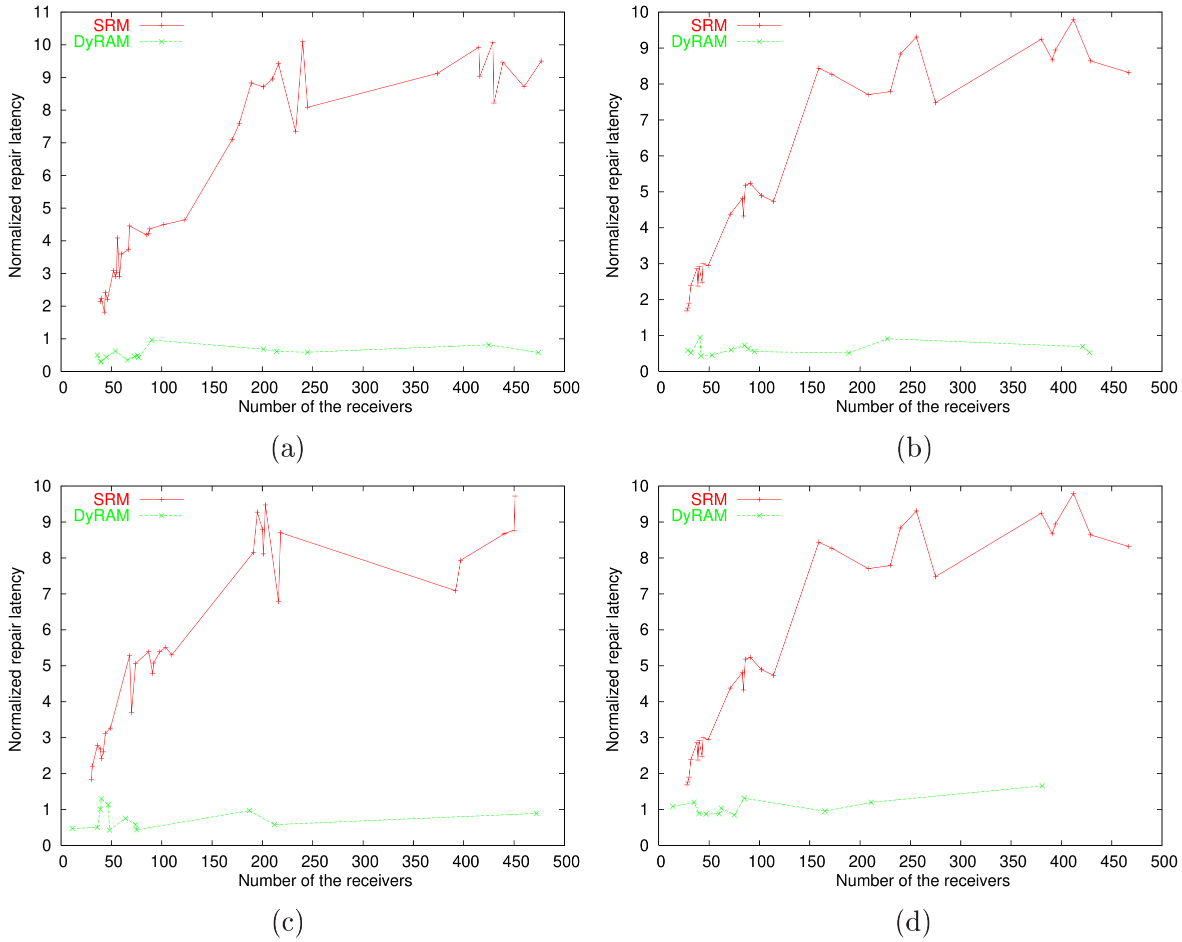


Figure 6.19: Repair latency in SRM and DyRAM with losses at the tail links (a) $p_l = 0.01$, (b) $p_l = 0.05$, (c) $p_l = 0.1$ and (d) $p_l = 0.25$.

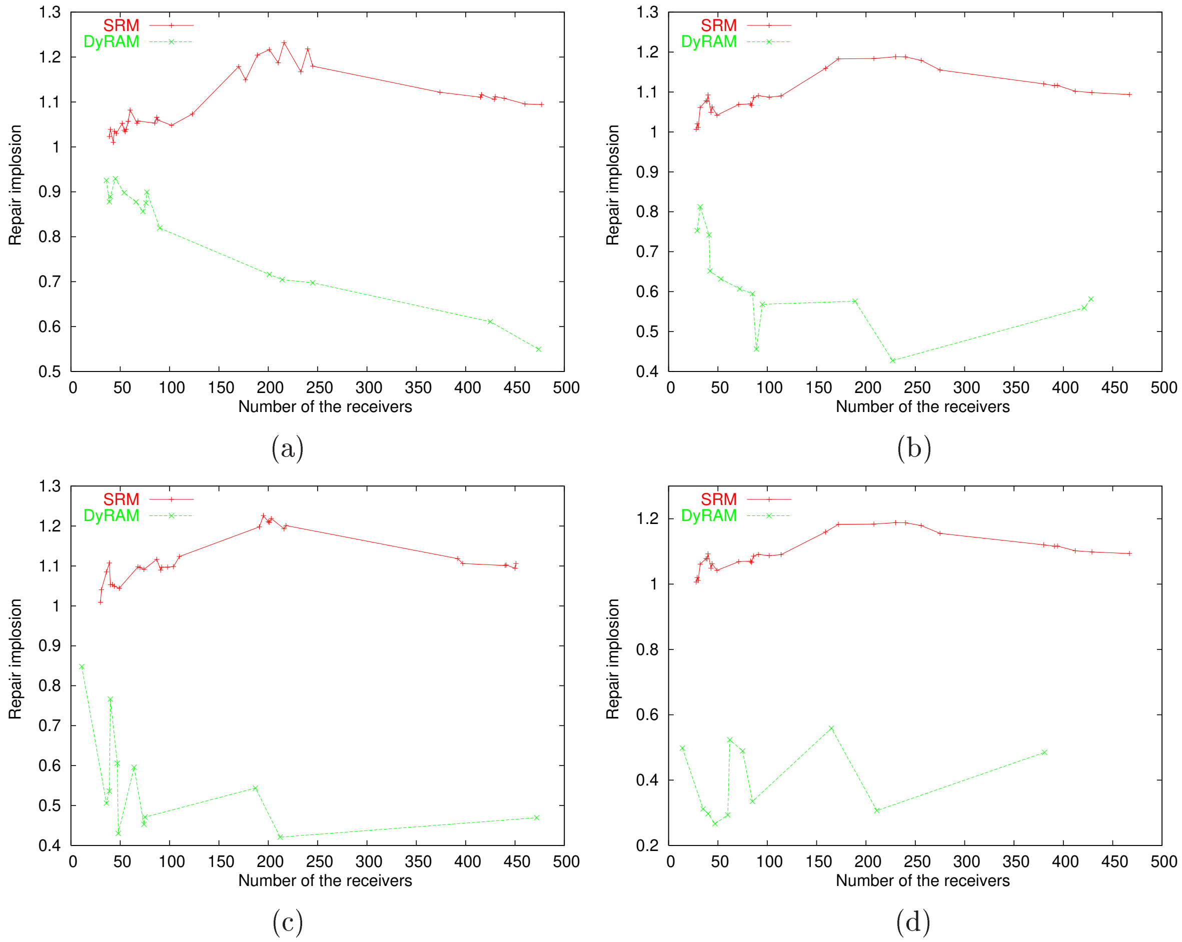


Figure 6.20: Repair implosion in SRM and DyRAM with losses at the tail links (a) $p_l = 0.01$, (b) $p_l = 0.05$, (c) $p_l = 0.1$ and (d) $p_l = 0.25$.

In this evaluation study, we evaluated the different active services proposed in DyRAM such as local recovery from the receivers, feedback suppression/aggregation and the loss detection service. To do so, DyRAM has been compared to several protocols such as A, D, ARM and SRM. We found that DyRAM allows for less load in terms of consumed bandwidth on the different multicast nodes and that the different services allows for reducing the repair latency and implosion. The impact of varying the percentage of active routers in the multicast tree has also been considered. We showed that increasing the density of active routers allows for better performances. The comparative study conducted between DyRAM and SRM has also allowed for showing the load balancing feature of our approach. Finally, simulations with the large number of receivers have been conducted in order to evaluate the scalability of DyRAM. Results have shown that DyRAM is scalable since it allows for a repair latency which is less than 1 even when increasing the number of the receivers.

Chapter 7

AMCA: an Active-based Multicast Congestion Avoidance Algorithm

7.1 Introduction

In the two previous chapters, a reliability mechanism called DyRAM has been presented and extensively evaluated. DyRAM insures reliability through recovery mechanisms where retransmissions are performed for lost data packets. However, it appears that packet losses in the Internet are mainly due to congestion situations in the routers where packets are dropped as soon as their queues are full. The source transmission rate has to be regulated according to the bandwidth available in the multicast tree. This is what a congestion avoidance algorithm is able to do. In addition to the classical properties like responsiveness and stability, a multicast congestion algorithm has to be scalable and fair with TCP and other multicast flows. These requirements are not easy to achieve due to the co-existence of multiple receivers with different end-to-end paths that could share some branches in the multicast tree. A scalable mechanism to probe for available bandwidth in a multicast tree has to be provided.

In this chapter, we propose a new active-based multicast congestion avoidance algorithm (AMCA) that tries to meet the above requirements (responsiveness, stability, scalability and fairness). In this study, AMCA has been integrated into the DyRAM framework but can be easily adopted to any other active reliable multicast protocol. AMCA is a congestion avoidance mechanism rather than a control mechanism because the sender reacts before a loss occurs since bandwidth is probed through RTT variation measurements. These latters provide an estimation of the number of the queued packets during a given period of time, used as the congestion measure. To avoid the problem of rerouting paths which could give erroneous estimations of the RTTs, we adopt a hop-by-hop RTT estimation and bandwidth probing. Using the active networking technology for practical implementation, a per-section dialogue is performed between adjacent nodes including the routers. A section is defined as the set of point-to-point links and traditional routers that connect two active routers or an active router to a terminal node (a receiver or the source). In AMCA, an RTT variation is measured for every section. The different RTT variations are aggregated appropriately so the source ends up by receiving the overall RTT variation experienced by the worst end-to-end path of the multicast tree.

Like RMANP [4] and NCA [64], we benefit from the physical multicast tree to perform congestion feedback aggregation and thus overcome the complexity inherent to building and maintaining a logical tree structure [27, 106]. The main difference between AMCA (rate-based) and NCA (window-based) resides in the usage of different congestion measures. Whereas AMCA uses mainly the RTT variation, NCA uses both the loss rate (p) and the RTT to determine the worst receiver (by computing the quantity $RTT\sqrt{p}$). However the loss rate is very difficult to be estimated on a short period while a sufficiently long period would make the congestion control unresponsive. Moreover, when there are no losses, all the receivers, according to their adopted measure, are considered to have the same capacity. Receivers are distinguished among them only and only if at least one receiver experiences a loss. This is also the case of RMANP which detects congestion on packet losses. In our case, the RTT variation measure in AMCA is able to predict congestion situations even in the absence of losses. Unlike RMANP, AMCA do not perform any cache of data packets in the active routers. We believe that an active router has to support multiple multicast sessions and caching means are not always available.

In what follows, we first show in section 7.2, how an RTT variation measure can be used to probe for available bandwidth in a connection. Afterwards, we present our mechanism adopted to estimate the RTT variations in section 7.3. How feedback messages are suppressed and aggregated along the multicast tree is dealt with in section 7.4. Then, we show in section 7.5, how the source reacts to feedback it receives so the multicast tree is prevented from congestion. In section 7.6, simulation results are presented before concluding.

7.2 Probing Extra Bandwidth of a Connection

In this section, we consider the case of a connection with one queue between a sender and one receiver (figure 7.1). Let r_b be the bottleneck rate of the connection and r the transmission rate of the sender in bit/s. An intuitive observation is that while the transmission rate is less than the bottleneck rate, the reception rate is equal to the emission rate. The reception rate could not exceed the connection bottleneck rate r_b even if the emission rate is increased. When the transmission rate exceeds the bottleneck rate, a queue of packets will build up through the connection. Let $q(t)$ be this queue size in packets at time t. Let Δq be the positive or negative variation in the queue length during a given time period T (figure 7.1):

$$\Delta q = q(t+1) - q(t)$$

Noting by S the packets size, the queue builds up when $r > r_b$ during T with:

$$\Delta q = (r - r_b)\,\frac{T}{S} \tag{7.1}$$

To estimate Δq we can employ a mechanism based on round trip time variations since a growing up (or the opposite case) queue produces an increased (or a decreased) RTT. The RTT variation $\Delta\tau = RTT(t+1) - RTT(t)$ can be expressed in terms of Δq as follows:

$$\Delta\tau = \Delta q\,\frac{S}{r_b} \tag{7.2}$$

which gives:

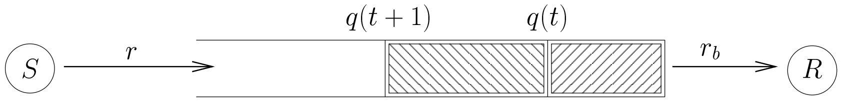


Figure 7.1: Queue length variation.

$$r_b = \Delta q \, \frac{S}{\Delta\tau}$$

Substituting Δq by its expression from (7.1), we find:

$$r_b = r \, \frac{T}{T + \Delta\tau} \tag{7.3}$$

This gives an expression of the bottleneck rate and is used by our congestion avoidance algorithm to adjust the rate so the transmission rate will be the closest to the bottleneck rate. From (7.1) and (7.2) we derive the next equation which gives an expression of Δq as a function of r, T, S and $\Delta\tau$:

$$\Delta q = \frac{r\,T}{S} \, \frac{\Delta\tau}{\Delta\tau + T} \tag{7.4}$$

We can easily obtain accurate values for r, T and S. However for $\Delta\tau$ we have to provide an estimation mechanism. To do so in such a simple connection, the source could have an estimation of the RTT variation just by exploiting feedback information. For instance if we suppose that the receiver sends an ACK per packet received then the source can use the difference between the two last experienced RTTs. The T period in this case is S/r_b.

The problem when using such a method to probe available bandwidth appears when we have a unicast (or multicast) session with multiple links. The path between the sender and the receiver can change and this would give erroneous results. [91] pointed out that TCP Vegas suffers from similar situations since rerouting a path may change the propagation delay of the connection which could result in a substantial inappropriate increase/decrease in throughput. To overcome this problem, we propose to use a per-section bandwidth probing approach. The main concern is to be able to estimate the RTT variations in a per-section fashion and then how these RTT variations are aggregated before giving them to the source. To do so, we have to establish communications between every node and its neighbor in a connection.

7.3 Congestion Feedback

In AMCA, every receiver sends a CR on the reception of a given number of packets rather than sending a CR every a constant period, so the congestion situation is probed proportionally to the emission rate. The resulting period is $T = NS/r$ which represents the expected time to receive N data packets by a receiver where r is the emission rate of the source. In the case where no data packet is received, a NACK is sent instead of a CR to indicate that a packet is lost by a receiver. With $T = NS/r$, from (7.4) the queue size variation can be given by:

$$\Delta q = N \, \frac{\Delta\tau}{\Delta\tau + T} \tag{7.5}$$

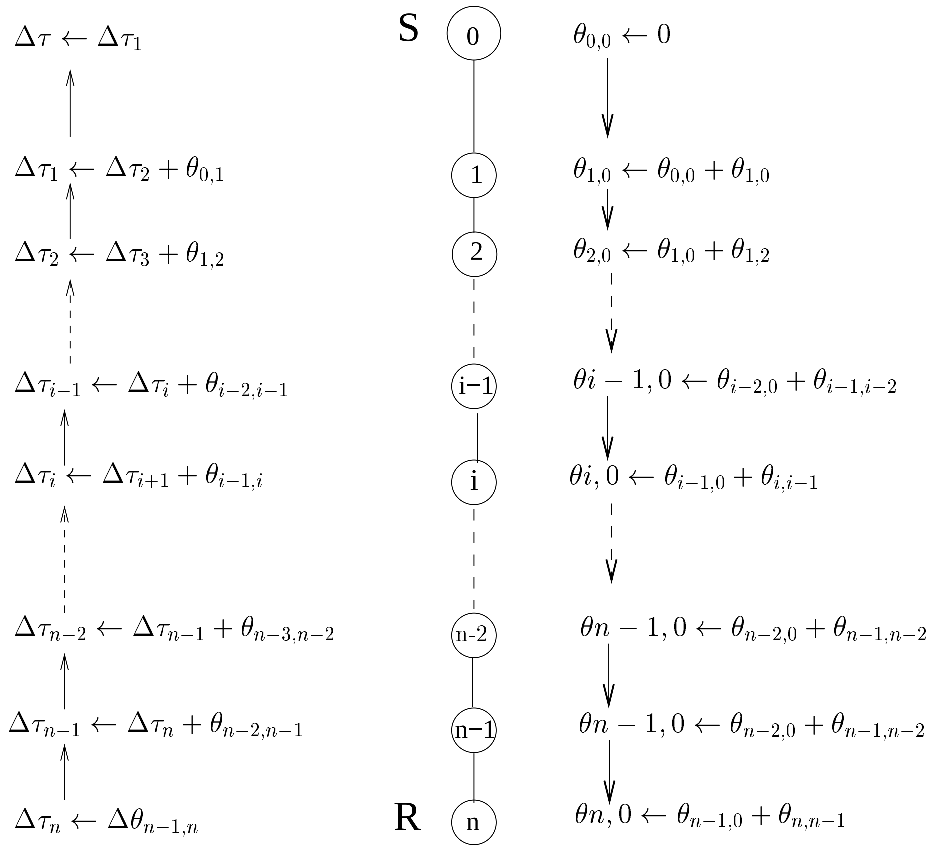


Figure 7.2: RTT variation estimation and CRs aggregation in a linear path of the multicast tree.

We introduce Δq_p defined as the queue size variation per packet:

$$\Delta q_p = \frac{\Delta q}{N} = \frac{\Delta\tau}{\Delta\tau + T} \tag{7.6}$$

The periodicity of sending the CRs varies because it depends on the current reception rate. To make the source aware of the current period, a receiver reports this latter in the *period* field of a CR (figure 5.1).

In what follows, we note by $\Delta\theta_{i,j}$ the experienced RTT variation between the ith and the jth node. For the ith link that connects the $(i-1)$th and the ith node, the RTT variation is $\Delta\theta_{i-1,i}$. Additionally we note by $\Delta\tau_i$ the RTT variation of the $(i-1)$th intermediate node and the receiver. The overall experienced RTT variation between the source and the receiver is $\Delta\tau = \Delta\tau_1$. Initially, a receiver (left side of figure 7.2) computes the RTT variation $\Delta\theta_{n-1,n}$ of its upstream link (number n) using the last two RTT measures to its parent with the mechanism described in section 5.4 (right side of figure 7.2). Afterwards the receiver includes the computed RTT variation in the CR to be sent to the source. The receiver's parent (i.e. node $n-1$) adds the RTT variation reported by the receiver and its own RTT variation to its upstream node to obtain the RTT variation of its parent ($(n-2)$th node) to the receiver:

$$\Delta\tau_{n-1} = \Delta\tau_n + \Delta\theta_{n-2,n-1}$$

Node $(n-1)$ will then report $\Delta\tau_{n-1}$ in the $RTTvar$ field of the CR. In this way, a parent node i, on receipt of a CR from node $(i+1)$ adds its RTT variation to node $(i-1)$ to the received one before forwarding the CR with the newly computed RTT variation:

$$\Delta\tau_i = \Delta\tau_{i+1} + \Delta\theta_{i-1,i}$$

The sender will end up by receiving the RTT variation experienced on the whole linear connection $\Delta\tau$. This latter can be used in equation (7.3) or (7.6) to have respectively a lower bound for the bottleneck rate r_b or the estimated queue length variation per packet Δq_p during the previous period.

7.4 Feedback Suppression/Aggregation

To achieve scalability, a mechanism of aggregation/suppression is performed. We use the physical multicast tree to hierarchically aggregating feedback at the intermediate nodes (routers). A suppression mechanism is performed on NACKs (by DyRAM for instance) thus allowing just one NACK per loss to be sent upstream. Since we do not suppose that all the routers are active, more than one NACK for the same loss could be received by the source. To avoid reacting to duplicate NACKs, the source reacts to the first one and ignores the subsequent ones for a given period of time which depends on the multicast tree structure. Details about the NACK suppression have been provided in chapter 5.

The CRs are aggregated at the intermediate nodes. 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 aggregation of the RTT variations reported by the CRs from downstream in the case of a multi-path connection, we illustrate our solution with the simple two-level tree depicted in figure 7.3. We consider two receivers R_1 and R_2 connected to the source S via one active router A. Receivers R_1 and R_2 send CRs to the active router with their respective RTT variations $\Delta\tau_1$ and $\Delta\tau_2$. Once these children CRs are received, the active router A sends its CR to the source with $\Delta\tau$ computed using:

$$\Delta\tau = \Delta\tau_{up} + Max(\Delta\tau_1, \Delta\tau_2)$$

where $\Delta\tau_{up}$ is the RTT variation of the source link (A, S). This latter is extracted from the HB_RESP message sent by the source. Using this method in aggregating the RTT variation, the source ends up by receiving the RTT variation experienced by the worst end-to-end path of the multicast tree.

7.5 Rate Regulation

In AMCA, the regulation parameter is the rate. A minimum and a maximum rates r_{min} and r_{max} are set by the application. Any receiver that could not support the minimum transmission rate has to leave the multicast session. Initially, the source starts to send data packets with a rate equal to the minimum rate r_{min}. Then it tries to increase its rate if no congestion indication is received without exceeding the maximum rate r_{max}. The source uses the

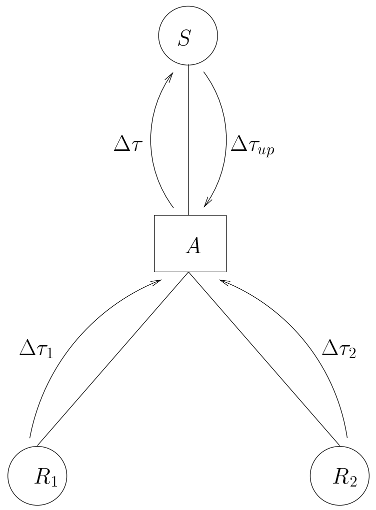


Figure 7.3: CRs aggregation in a multicast tree

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 a phase similar to the slow start phase of TCP, the source would increase its rate by 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 by S/RTT_{max} where RTT_{max} is the maximum experienced RTT among the receivers, is equivalent to adding 1 to the congestion window for the largest end-to-end connection of the multicast tree. This behavior makes our congestion avoidance algorithm fair with TCP from the beginning of the multicast session.

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

$$\Delta q_a = \max(0, n_s - n_a) \tag{7.7}$$

where n_s and n_a are respectively the number of sent and acked packets during the previous period. Since a CR is sent to the source every N packets received during a period T, we have:

$$n_s = \frac{T}{S} r = N$$

The number of acked packets is computed using:

$$n_a = lo_{i+1} - lo_i$$

where lo_{i+1} and lo_i are respectively the values of the lo field of the newly received and previously received CR. Finally:

$$\Delta q_a = \max(0, N - (lo_{i+1} - lo_i)) \tag{7.8}$$

The source, on the reception of a CR, extracts the current period (T) and the RTT variation ($\Delta\tau$). Afterwards, in addition to the queue size variation per packet Δq_p, the number of data packets not acked yet Δq_a is also computed using (7.8). The aim is to maintain Δq_a in $[\alpha, \beta]$, where α and β are similar to those of TCP-Vegas.

In the slow start phase, the source continues increasing its rate every time it receives a CR that indicates $\Delta q_p < \epsilon$ and $\Delta q_a < \beta$. The source enters the congestion avoidance phase when it receives a NACK or a CR with $\Delta q_p > \epsilon$ or $\Delta q_a > \beta$. On the receipt of a NACK, the source reduces the rate by half. Subsequent NACKs are ignored during a period estimated by the difference between the current RTT to the source of the furthest and the closest receiver in the multicast tree ($RTT_{max} - RTT_{min}$). To avoid decreasing dramatically the rate because of isolated losses, we check the lo and lr fields of the NACKs and reduce the rate only if $lr - lo \geq 3$, otherwise we just retransmit the corresponding repair.

During the congestion avoidance phase, on the reception of a CR with an RTT variation $\Delta\tau$ and a period T, the source updates its rate depending on the current values of both Δq_p and Δq_a. When these two measures do not indicate congestion ($\Delta q_p \in [0, \epsilon]$ and $\Delta q_a < \beta$), the rate is multiplied by γ_{inc} chosen slightly greater than 1. In the other cases, the rate is multiplied by $\gamma = T/(T + \Delta\tau)$ which is derived from (7.3) and allows the rate to converge to the bottleneck rate. When the emission rate is greater than the bottleneck rate, a queue will build up in the bottleneck link. This will be translated by a positive RTT variation $\Delta\tau$ and thus $\gamma < 1$. Multiplying by γ will decrease the transmission rate and this is the appropriate action from the source. In the reverse situation where the emission rate is less than the available bandwidth, $\Delta\tau$ is negative and multiplying by $\gamma > 1$ will increase the transmission rate. In addition to the proposed mechanisms, a severe congestion is detected if no CR, HB or NACK is received during a relatively long period which is set to twice the current period ($2T$). In this case the rate is dropped to its minimum value r_{min}.

Since DyRAM implements a local recovery mechanism, we have cared about the local recovery problem. Many reliable protocols perform local recovery which leads to the coexistence of multiple data sources. This is why not only the original source has to be congestion controlled, but retransmitting entities (receivers, routers or servers) should also be controlled. Additionally, a feedback suppression is implicitly or explicitly performed which would make the source unresponsive to congestion variations. Suppose that behind a congested link, one or more receivers have sent NACKs that will be received by a replier instead of the source. The replier will retransmit the missed packets and the source is never aware about the congestion and would continue to send with the same rate and even with a higher rate. As a result the congestion situation is aggravated. This is the so called *local recovery problem* [92].

In our scheme, we propose to make the source aware about any NACK that has been forwarded downstream to an elected replier. An active router when aggregating the CRs, includes NACK information mainly in the lo and lr fields of the aggregated CRs. In this way, the source is always aware about losses that occur in the multicast tree even in the presence of local recoveries. The retransmission rate from the repliers is also controlled to do not

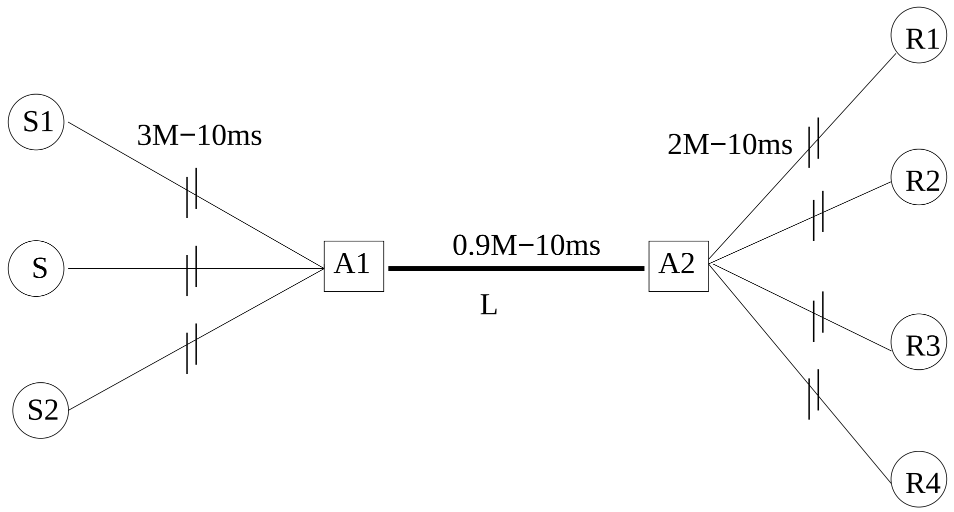


Figure 7.4: The used topology

aggravate a congestion situation until the source adapts its rate. When an elected replier has to retransmit more than one packet, then it sends repairs with a rate equal to the current rate of the source divided by $S(lr - lo)$ which is proportional to the congestion degree. Finally, it is worth mentioning that the DyRAM loss detection service at the routers could be very helpful for the congestion control. A NACK is sent to the source as soon as a loss is detected and the source would reduce its rate rapidly which could enhance the responsiveness of the proposed congestion avoidance algorithm.

7.6 Simulation Results

A simulation model of AMCA with DyRAM has been implemented using ns-2.1b8 (network simulator [46]). For all the simulations the data packet size is set to 1024 bytes and the NACK/CR packet size is set to 32 bytes. The minimum and maximum rates are set respectively to 1 and 10000 packets per second. While ϵ is set to 0.2, α and β are respectively set to 1 and 3. For the rate regulation, γ_{inc} is set to 1.05. A receiver sends a CR every $N = 32$ data packets received unless said otherwise. For all the simulations, we used the simple topology of figure 7.4 with four receivers separated by two routers from the source. The bottleneck link L has a bandwidth of $0.9Mbps$. S is the DyRAM source that multicasts data to the receivers $R1$ and $R2$. $S1$ and $S2$ are used as sources for TCP (Vegas) flows to study the fairness of DyRAM with TCP. $R3$ and $R4$ are the TCP sinks of $S1$ and $S2$ respectively.

7.6.1 Convergence of the Algorithm

A first set of experiments is conducted with just one multicast flow. The source $S1$ multicasts data using DyRAM to the receivers of figure 7.4. The purpose of this first set of experiments is to show how our congestion algorithm probes and makes use of the available bandwidth. Figure 7.5 shows the achieved throughput by the receivers for different values for N. We can see that the algorithm converges to the bottleneck available bandwidth of $900Kbps$ and then makes use of the available bandwidth during all the simulation time. It is worth showing that there is a tradeoff between the responsiveness (quick reaction) and the stability (in terms of oscillations) depending on the CRs periodicity. We can see that the smaller N is, the more responsive the algorithm is. However the oscillations are smoother when we increase N.

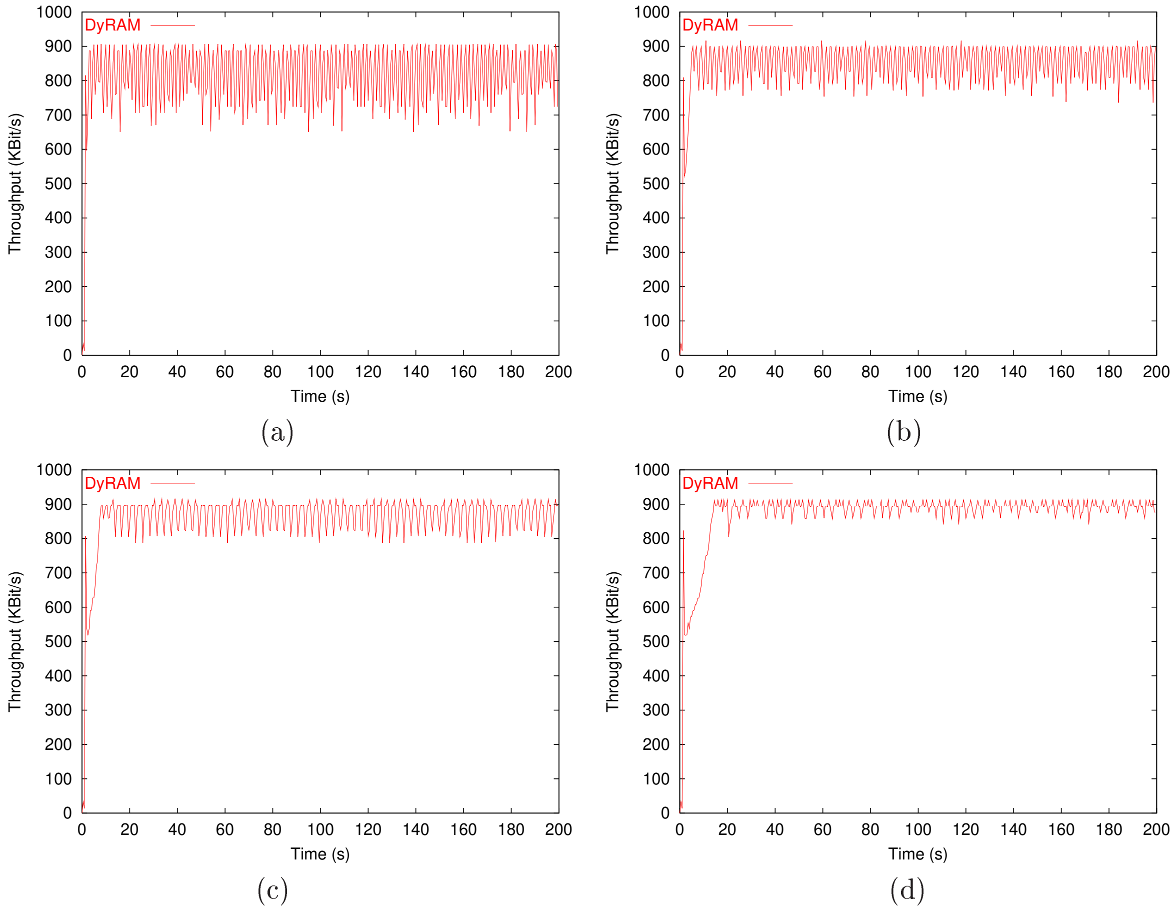


Figure 7.5: The tradeoff between responsiveness and stability (a) 16, (b) 32, (c) 64, (d) 128

7.6.2 Fairness with TCP

Since most of the Internet traffic is transported by TCP, the fairness to TCP flows is an important issue. To evaluate the fairness of our scheme, experiments have been conducted on the topology depicted in figure 7.4. One experiment is performed with one DyRAM flow competing with two TCP flows. The DyRAM source is S and the 2 TCP flows are sent by $S1$ and $S2$ to the receivers $R3$ and $R4$ respectively. The simulation is run for 200 seconds. All of the three flows are started at the same time. Figure 7.6 shows the achieved throughput by the different flows. We can see that the bottleneck link of $0.9M$ is almost equally shared among the three flows.

To show the dynamic behavior of our congestion algorithm, we have run a simulation with one DyRAM multicast flow in the background and one TCP flow started at 20s and stopped at 100s. Figure 7.7(a) shows the achieved throughput by DyRAM and TCP. We can see that DyRAM makes use of all the available bandwidth while the TCP flow is not started yet. In

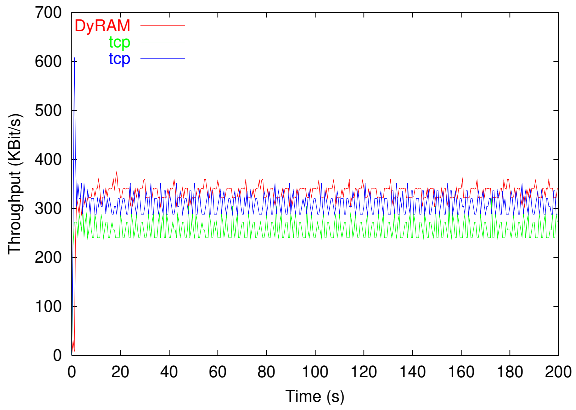
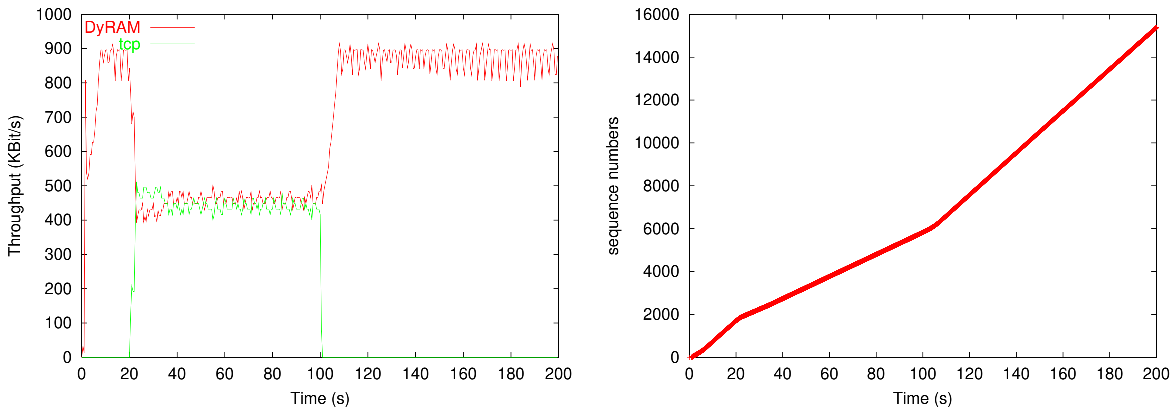


Figure 7.6: Fairness with TCP

the presence of the two flows, the available bandwidth is shared equally among them. When the TCP flow is stopped, DyRAM is able again to make use of the available bandwidth. We note that the required time by DyRAM to react to both the existence and the absence of the TCP flow, is very small. This shows the responsiveness of our proposed algorithm. Figure 7.7(b) shows the sequence numbers of data packets received by the DyRAM receivers. We can observe that DyRAM dynamically adjusts its rate and the slope angle of the sequence number line is divided by 2 in the presence of the TCP flow. Another interesting observation is that there is no packet loss during all the simulation time.

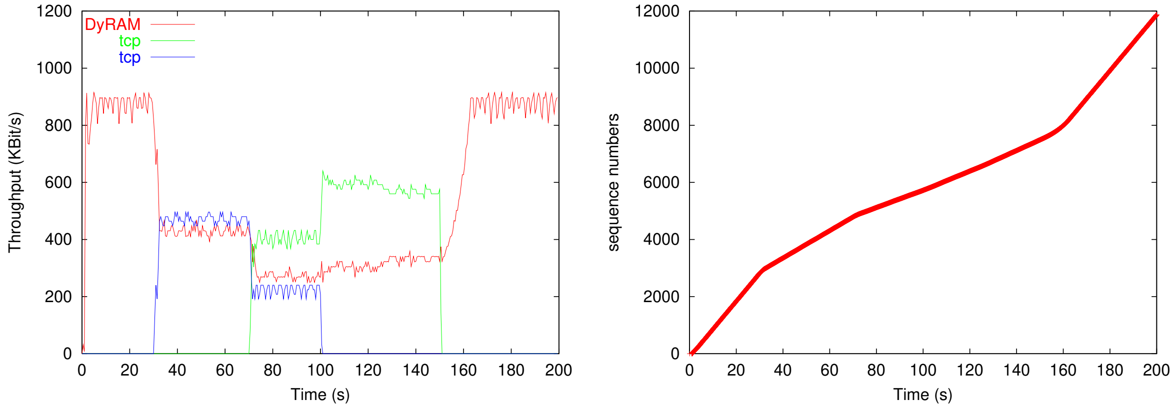


(a) (b)

Figure 7.7: Dynamic behavior of DyRAM with one TCP flow

Figure 7.8 shows the results of a simulation run with two TCP flows and one DyRAM multicast flow. DyRAM is run from 0 to 200s and the two TCP flows are run respectively from 30 to 100s and 70 to 150s. We can note the same observations as for the previous experiment. DyRAM equally shares the available bandwidth with two TCP flows and always react quickly to the network dynamics. DyRAM is always fair and never aggressive with TCP.

Another simulation has been performed with the TCP flow in the background and the



(a) (b)

Figure 7.8: Dynamic behavior of DyRAM with two TCP flows

DyRAM flow run from 20 to approximately 118s (the required time to send 5000 data packets). Figure 7.9 shows the achieved throughput by the two flows. It is clear that once more DyRAM is proved to be TCP-friendly and that it shares resources equally with any TCP flow.

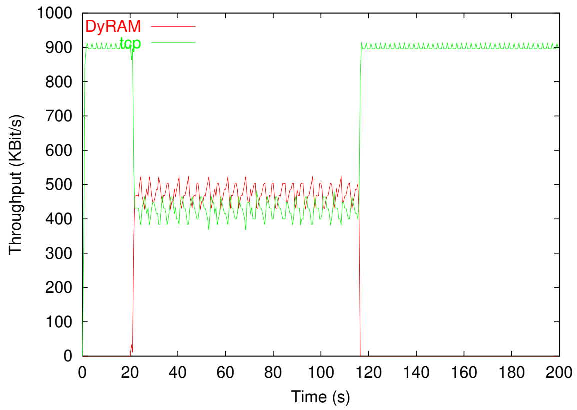


Figure 7.9: Fairness with one TCP flow in the background

7.6.3 Multiple Bottleneck Links

To see the behavior of our congestion avoidance algorithm with TCP flows in the presence of multiple bottleneck links, we have performed simulations with the same topology of figure 7.4 with some lower capacity links. The simulations are performed with 2 TCP flows and one DyRAM flow. The TCP flows are from S1 and S2 to R3 and R4 respectively. S is the DyRAM source that multicasts to the receivers R1 and R2. We set the link bandwidth of receivers R2 (one of the DyRAM flow receivers) and R4 to $0.2Mbps$ instead of $2Mbps$. Figure 7.10(a) shows that DyRAM achieves a throughput that corresponds to the minimum available on the whole multicast tree ($0.2Mbps$). The TCP flow to R4 uses the remaining bandwidth of the link L. Figure 7.10(b) shows the achieved throughput by the different flows where the bandwidth of the TCP receivers (R3 and R4) links is set to $0.2Mbps$. We can see that the 2 TCP flows use the available $0.2Mbps$ bandwidth for each of them and that DyRAM makes use of the remaining bandwidth on the bottleneck link.

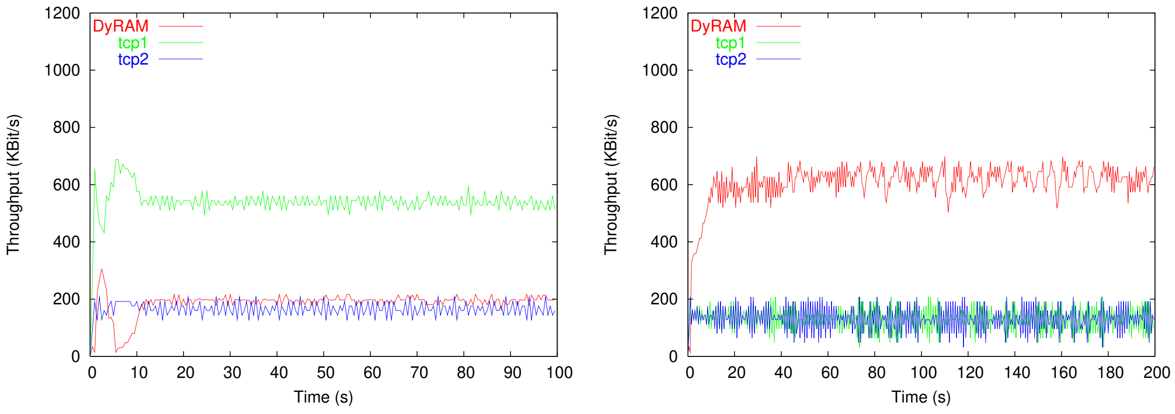


Figure 7.10: Multiple bottleneck links : (a) R1,R4(2M), R2,R3(0.2M) and (b) R1,R2(2M), R3,R4(0.2M)

7.7 Conclusion

In this chapter, we have presented AMCA an active-based congestion avoidance algorithm for multicast communications. The proposed scheme is validated with DyRAM, an active reliable multicast protocol. Nevertheless AMCA can be used by any other active reliable multicast protocol and particularly in the case of unicast where the problem of rerouting path that TCP Vegas suffers from, can be avoided. Our congestion algorithm relies on the active networking technology to perform a per-section dialogue to probe for available bandwidth along a multicast tree. This is done by using the RTT variations experienced by every section in the multicast tree. The RTT variations are then appropriately aggregated by intermediate nodes before they are given to the source.

A set of simulations have been conducted using ns-2.1b8 to study the behavior of our congestion avoidance algorithm. It has been shown that AMCA converges, makes use of the available bandwidth and reacts rapidly to the dynamic changes experienced by the multicast tree. In addition, simulations have shown that losses are very few showing the conservative property of AMCA which reacts to congestion before a loss occurs. Finally, our algorithm is proved to be fair with TCP.

AMCA is a single-rate approach and the emission rate is adapted according to the worst receiver in the multicast group. Transmitting with a rate which matches the slowest receiver will limit the throughput of other receivers and thus their satisfaction. In a multicast session, a multi-rate mechanism can improve the receivers' satisfaction. In the next chapter, AMCA will be extended to support multiple transmission rates thus allowing for more heterogeneity in the multicast group.

Chapter 8

Dealing with Heterogeneity in a Reliable Multicast Protocol

8.1 Introduction

The multicast congestion avoidance algorithm AMCA proposed in chapter 7 is single-rate where heterogeneity is accommodated by adjusting its rate in response to the most congested path in the multicast tree. This latter is dynamically determined based on round trip times (RTT) variation estimated in a per-hop manner. Transmitting with a rate which matches the slowest receiver, limits the throughput of other receivers and thus their satisfaction. In a multicast session, a multi-rate mechanism can improve the receivers' satisfaction. In fact, receivers with different needs can be served at a rate closer to their needs rather than having to match the speed of the slowest receiver. In a multi-rate session, the multicast source can transmit at different rates either through a hierarchical scheme (layering) [60, 80, 123] or a replicated scheme (destination set grouping, DSG [62]).

In layered multicast, each receiver controls the rate at which it receives data, usually by using multiple multicast groups. The receivers join and leave groups depending on their path congestion state so the amount of data being received is always appropriate. In a replicated scheme, the source splits the receivers into subgroups of similar capacities. Afterwards, it sends a separate flow to every subgroup with the appropriate rate. Layering schemes provide more economical bandwidth usage than DSG schemes, however layering is more complicated and requires efficient hierarchical encoding/decoding algorithms and synchronization among different layers.

A multi-rate multicast improves the *inter-receiver fairness* (or equivalently, the *intra-session fairness*) since each receiver can be served at a rate which is the closest to its isolated rate. However, fairness toward other unicast and multicast sessions known as *inter-session fairness* is required and has to be satisfied. Many works [8, 122, 61, 11, 115] have addressed the concept of fairness and provided some definitions. Earlier fairness definitions [122, 61, 11] did not consider multi-rate schemes. More recently, Rubenstein et al. [115] studied fairness in the context of a multi-rate multicast. They identified four desirable properties of a *max-min fair allocation*. The main fairness property referred to as "*Fully-Utilized-Receiver-Fairness*" states that in a max-min fair allocation:

- either a receiver gets its highest utility (or equivalently, its isolated rate),
- or there exists at least one fully utilized link over which bandwidth is equally allocated to all this link's downstream receivers.

Rubenstein et al. [115] have shown that this property and the three others[1] could not be satisfied by a single-rate scheme because some receivers could not have their highest utility since they are limited by the worst receiver in the multicast tree. This would not be necessary the consequence of the existence of a fully utilized link in their path neither a fair share of such a link with other sessions. In such a case, other flows could be aggressive with a single-rate multicast. On the other hand, they have shown that a layered approach is more max-min fair than single-rate multicast. However, they did not consider replication-based multi-rate schemes in their study.

In a DSG approach [62] (referred to as *source-based replication scheme*), data replication could be a source of unfairness. Allocated bandwidth to a source-based replication scheme is almost the sum of all the rates of the replicated streams at the links near the source. This could make this replication scheme aggressive with other flows. More generally, we state that a multi-rate scheme could be a source of unfairness if it exists at least one link on which the consumed bandwidth is greater than the isolated rate of the fastest receiver among those located downstream this link. Jiang et al. [62] proposed an allocation algorithm that satisfies a constraint known as *the aggregated allocation connection* (ACC) which states that the total bandwidth consumed on any link by an ACC can not exceed the maximum isolated rate of a downstream receiver in the multicast tree. This algorithm was designed for ATM networks and appears to be difficult to be adopted in an Internet-like network.

A layered approach however, satisfies the ACC constraint since there is no data replication and that the cumulative rate of the different layers would not exceed the isolated rate of the fastest receiver. However, from a practical point of view, layered approaches fail to be totally max-min fair. The adaptation granularity is at the layer level since a receiver can not subscribe to a fraction of a layer. To achieve a fine-grained adaptation, a layered scheme needs to have an infinite number of layers. An other way to achieve fine-grained adaptation in a layered approach consists in having the layers' rates change over the time depending on the receivers requirements [72]. In their replicated scheme, Jiang et al. [63] proposed a two-group scheme where the highest subgroup transmission rate is adapted in response to feedback from the receivers that belong to.

In order to avoid the complexity inherent to a layered scheme especially for the case of a fully reliable multicast, we propose to use a replicated scheme[2]. In order to be fair with other flows while improving the receivers' satisfaction, we propose in this chapter, a new replicated scheme where data replication is no longer the responsibility of the source. We refer to this new approach as *receiver-based replication scheme* which differs from the source-based replication scheme in the fact that data replication is performed by some designated receivers instead of the source.

[1] these three other properties are omitted here, for more details the interested reader can refer to [115].

[2] Unlike widely believed that layering is better than replication, experiments [70] showed that layered multicast is not necessarily superior to replication schemes in many circumstances.

More precisely, our solution consists in a distributed approach where some receivers (*replicators*) contribute in the replication of the data flow with an appropriate rate to other receivers of lower capacity. In this way, a *regulation tree* is built with the source as the root, the replicators as intermediate nodes and the remaining receivers as final nodes. In order to minimize bandwidth consumption due to data replication, the regulation tree is built in respect to the physical multicast tree. This is achieved by involving the routers in the regulation tree construction procedure. Every router performs a partition of its downstream links into subgroups and chooses a replicator for every subgroup formed.

In addition to scalability, building the replication tree with respect to the multicast tree, allows for the avoidance of the coexistence of multiple flows on the same link since receivers that belong to the same subtree are more likely to belong to the same subgroup. In a source-based replication, receivers far from each other could belong to the same subgroup which could result in data replication on some common links mainly near the source. Furthermore, our approach allows for a fine-grained rate adaptation (congestion control, see section 8.5) since independently of the source and other replicators, each replicator adjusts its rate according to the feedback it receives from its associated subgroup.

In order to the rates of the replicated flows to be adapted to the receivers population subscribed to a multicast session, first a new partitioning algorithm which can be executed at either the source or the routers is presented. This algorithm has the ability of performing an on-the-fly partitioning of the receivers depending on feedback it receives. Mainly, this algorithm does not require a prior estimation of the receivers' capacities. Some proposals suggest the use of RTT [53], loss rate [63] measurements or both of them [13] in order to estimate the capacity of the different receivers. Jiang et al. proposed in [63], a special (two-group) DSG protocol to be deployed in the Internet. They proposed a mechanism based on the experienced losses by a receiver to estimate its isolated rate. This mechanism can be used in loss tolerant applications; however the aim of our work is to be able to perform a partitioning in the context of fully reliable multicast applications where we privilege a conservative approach where losses are to be avoided as much as possible.

Our partitioning algorithm is based on the receivers RTT variations instead of their isolated rates. In this context we propose another formulation of the utility function using the RTT variations instead of the isolated rates. Our algorithm approximates and in many cases, achieves the optimal solution without using complex computations. For instance, in [3], a computation is performed on every candidate solution before choosing the one that maximizes the receivers satisfaction. The dynamic programming algorithm proposed in [135, 53, 13] requires less computation effort but still be complex.

In section 8.2, we present our partitioning algorithm which is first executed by the source. An analysis of this algorithm in addition to some simulation results will be provided. Afterwards in section 8.3, the partitioning algorithm is generalized to be executed at the routers to achieve more fairness and scalability. How a regulation tree can be built by the routers and the receivers in a distributed way to allow for a receiver-based replication of data is also dealt with. An analysis of this new approach is provided in section 8.3 with some simulation results. In section 8.4, the impact of our receiver-based replication scheme on network fairness towards other unicast flows is addressed. Finally, the proposed receiver-based replication

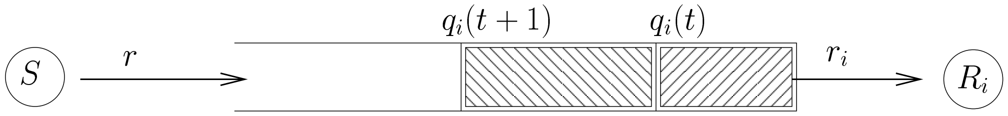


Figure 8.1: Queue length variation.

scheme is applied to AMCA. Details of the ns implementation as well as the corresponding simulation results are addressed in section 8.5.

8.2 An RTT-based Partitioning Algorithm

In this section, we develop our partitioning approach based on the RTT variations instead of of the isolated rates. First, a new formulation of the utility function is proposed. Afterwards, our RTT-based partitioning algorithm is presented and some of its properties are studied. Finally, some simulation results are provided.

8.2.1 Utility Function

In a multicast session, the satisfaction of a receiver R_i can be quantified using a utility function (or a receiver fairness measure) that maps the reception rate of the receiver to a fairness value normalized to the range $[0.0, 1.0]$. Authors in [61, 63] proposed the following fairness function:

$$U_i(r) = \frac{\min(r_i, r)}{\max(r_i, r)} \tag{8.1}$$

where r_i and r are respectively the R_i's isolated rate and the source transmission rate. The above utility function has a maximum value of 1 if the reception rate equals the receiver isolated rate ($U_i(r_i) = 1$) which corresponds to the maximal R_i's satisfaction. U_i has the property to be non-decreasing in the range $[0, r_i]$ and non-increasing in the range $[r_i, \infty]$. However, this utility function assumes that the isolated rates of the different receivers are known. We propose another expression of the utility function using the RTT variation experienced by the receiver as a response to a given transmission rate. The RTT variation is a measure which can be easily obtained in the Internet using a ping-like mechanism. Recall that the RTT variation noted $\Delta\tau$, experienced by a receiver is the difference between two consecutive RTT measurements.

For the purpose of redefining the utility function, we consider the case of one receiver with r_i as the isolated rate. Let r be the transmission rate of the sender in bit/s. When the transmission rate exceeds the bottleneck rate, a queue of packets will build up within the connection. We suppose that the receiver sends periodically (every T seconds) a probing packet towards the source in order to estimate its RTT to the source. Let $q_i(t)$ be this queue size in packets at time t. Let Δq_i be the positive or negative variation in the queue length during a given time period T upon the reception of the subsequent probing packet (figure 7.1). We have:

$$\Delta q_i = q_i(t+1) - q_i(t)$$

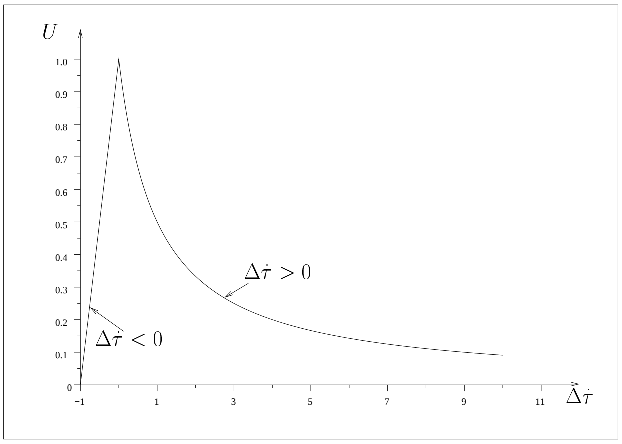


Figure 8.2: The inter-receiver fairness as a function of the relative RTT variation.

Noting by S the packets size, the queue builds up when $r > r_i$ during T with:

$$\Delta q_i = (r - r_i)\ \frac{T}{S}$$

The RTT variation $\Delta\tau_i$ experienced by the connection during the last T period can be expressed in terms of Δq_i as follows:

$$\Delta\tau_i = \Delta q_i\ \frac{S}{r_i}$$

which gives:

$$\Delta\tau_i = \frac{r - r_i}{r_i}\ T$$

Hence:

$$\Delta\dot{\tau_i} = \frac{r - r_i}{r_i}$$

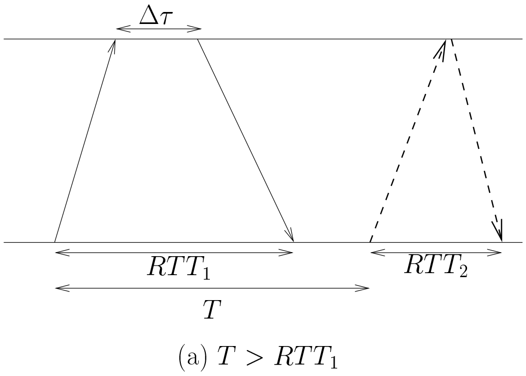
where $\Delta\dot{\tau_i} = \Delta\tau_i/T$ can be seen as the *relative RTT variation* experienced by receiver R_i. We can easily obtain:

$$\frac{r}{r_i} = 1 + \Delta\dot{\tau_i} \tag{8.2}$$

Using equation (8.2), the individual receiver utility expression (8.1) can be written as a function of the relative RTT variation as follows (figure 8.2):

$$U_i(r) = \begin{cases} \dfrac{1}{1 + \Delta\dot{\tau_i}} & \text{if } \Delta\dot{\tau_i} \geq 0 \\ 1 + \Delta\dot{\tau_i} & \text{if } \Delta\dot{\tau_i} \in [-1, 0[\end{cases} \tag{8.3}$$

The receiver that experiences a positive RTT variation could experience losses since its reception rate is greater than its isolated rate. In the case of a negative RTT variation, the receiver will be unsatisfied since it has more bandwidth resources. We note that the utility



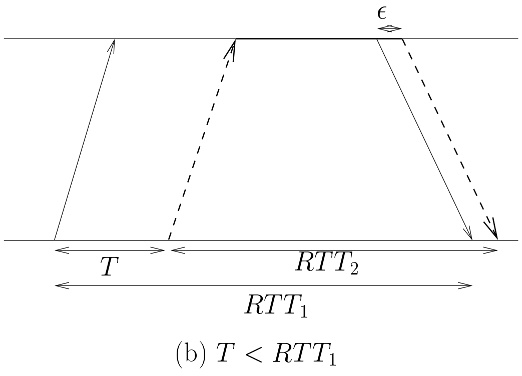


Figure 8.3: Lower bound for $\Delta\tau$.

function is not defined for $\Delta\dot{\tau}_i < -1$ which corresponds to a negative reception rate ($r < 0$).

Proof: Assume that $\Delta\dot{\tau}_i < 0$ for a receiver R_i (in this case $r < r_i$). We need to show that $\Delta\dot{\tau}_i$ could not be less than -1. Two cases are possible, either the next probing packet is sent after (figure 8.3a) or before (figure 8.3b) the reception of the previous probing packet. Let RTT_1 and RTT_2 be the successive RTT variations computed at the reception of the two successive probing packets.

In the first case (figure 8.3a), it is clear that $|\Delta\tau|$ could never be greater than the probing period T. That is $T > |\Delta\tau|$ which gives $\Delta\dot{\tau}_i > -1$.

In the second case (figure 8.3b), it is clear that the second probing packet can not arrive before the first one, that is:

$$\exists\epsilon > 0,\ |\Delta\tau| = RTT_2 - RTT_1 = T - \epsilon \Longrightarrow |\Delta\tau| < T$$

Since $\Delta\dot{\tau}_i < 0$ then we certainly have $\Delta\dot{\tau}_i > -1$. ■

In a similar way to [61], we define the utility function of a single-rate multicast session as the weighted sum of the individual utility values of receivers in the session:

$$U(r) = \sum_{i=1}^{n} \alpha_i U_i(r) \tag{8.4}$$

subject to $\sum \alpha_i = 1$ and $\alpha_i \in [0, 1], i = 1, \ldots, n$, where n is the number of the receivers in the multicast session. A multi-rate multicast session consists of one or more subgroups split from an original multicast session. The session utility function in this case is defined as the summation of the utility values obtained by all multicast subgroups, using the single-rate utility measure in each subgroup. More specifically, if a multicast session of receivers $\{R_1, R_2, \ldots, R_N\}$ is split into K subgroups $\{G_0, G_1, \ldots, G_{K-1}\}$ with the transmission rates $g_0, g_1, \ldots, g_{K-1}$, then this multi-rate multicast session is given by:

$$U(g_0, g_1, \ldots, g_{K-1}) = \sum_{k=0}^{K-1} \sum_{i=1}^{n_k} \alpha_{ik} U_{ik}(g_k) \tag{8.5}$$

subject to $\sum_{i,k} \alpha_{ik} = 1$ and $\alpha_{ik} \in [0,1]$. We have $\sum_k n_k = N$ where n_k is the number of the receivers in subgroup G_k. $U_{ik}(g_k)$ and α_{ik} are respectively the utility function and the weight associated to the ith receiver of the kth subgroup. In a fully reliable multicast, the transmission rate g_k for the receivers in subgroup G_k has to match the minimum rate of the subgroup isolated rates, i.e. $\forall k, \quad g_k = \min_{i \in G_k} r_i$

8.2.2 Partitioning Algorithm

Given a set of receivers $\{R_1, R_2, \ldots, R_N\}$ with isolated rates $r_1 < r_2 < \ldots < r_N$, the problem consists in splitting this set of receivers into K subgroups (K is less than a maximum number G) to make a partition $P = \{P_0, P_1, \ldots, P_{K-1}\}$ of the receivers so that the overall session utility is maximized. We aim to determine the optimal solution or at least an approximated one (without prior knowledge of the number of subgroups) such that the global receivers utility is greater than a given threshold. Our partitioning algorithm is based on the RTT variation experienced by every receiver and is executed on-the-fly while the source is increasing its rate. The source begins sending packets at a specific minimum rate and increases its rate as long as it does not receive any feedback indicating a relative RTT variation greater than a given threshold $\dot{\epsilon}$ (algorithm 1). The aim of algorithm 1 is to partition the receivers among subgroups of "similar" capacities. Initially, P_0 is the set of all the receivers. The source periodically checks if any receiver from P_0 experienced a $\Delta\dot{\tau}$ greater than a given threshold b; if so it splits P_0 into two subgroups P_0 and P_1:

$$P_1 = \{R_j \in P_0, \Delta\dot{\tau}_j > a\}$$
$$P_0 = P_0 - P_1$$

Only feedback from P_0 are considered by the source for the purpose of next splittings. The source will continue increasing its rate and would if necessary, split P_0 again into two groups P_0 and P_2. The splitting process will continue until G subgroups have already been built or the P_0 is no more "split-able" (there is only one element in P_0, i.e. $|P_0| = 1$, or members of P_0 have similar capacities).

Algorithm 1 Partitioning Algorithm. Input: P_0, a set of receivers. Output: a partition $\{P_0, P_{K-1}, \ldots, P_2, P_1\}$ where $K \leq G$

Require: $N > 1$ and $a < b < \dot{\epsilon}$
  $\rho \leftarrow \frac{a+1}{b+1}$
  the source rate $r \leftarrow r_{min}$
  $P_0 \leftarrow \{R_j, j = 1, \ldots, N\}$, the set of all the receivers
  $i \leftarrow 1$
  Periodically,
  if $\exists j, R_j \in P_0$ such that $\Delta\dot{\tau}_j > b$ **then**
    $P_i \leftarrow \{R_j \in P_0, \Delta\dot{\tau}_j > a\}$
    $P_0 \leftarrow P_0 - P_i$
    $i \leftarrow i + 1$
  end if
  until $i = G$ or $|P_0| = 1$ or $\forall j < N, R_j \in P_0, \frac{1+\Delta\dot{\tau}_{j+1}}{1+\Delta\dot{\tau}_j} \geq \rho$

Definition 1 (consecutive receivers) *We consider two receivers R_i and R_j as consecutive (in this order) if their corresponding isolated rates r_i and r_j satisfy $r_i < r_j$ and $\forall k$ ($k \neq i$ and $k \neq j$) $: r_k < r_i < r_j$ or $r_i < r_j < r_k$.*

We have the following lemma:

Lemma 1 *Under the precondition $G = \infty$, if $r_j/r_{j+1} < \rho$, then when the algorithm converges, the consecutive receivers R_j and R_{j+1} will not belong to the same subgroup.*

Proof: Let γ be the multiplicative factor by which the source rate is multiplied at every period. Suppose that at a given time the two receivers R_j and R_{j+1}, belong to the same group, this means that none of their RTT variations is greater than the b threshold, that is:

$$\Delta\dot{\tau}_j = \frac{r - r_j}{r_j} < b \qquad \textit{and} \qquad \Delta\dot{\tau}_{j+1} = \frac{r - r_{j+1}}{r_{j+1}} < b$$

At the next period the source rate becomes γr rather than r. Receivers R_j and R_{j+1} will no longer continue to be in the same subgroup if their new relative RTT variations $\dot{\tau}_j{}^+$ and $\dot{\tau}_{j+1}^+$ satisfy:

$$\Delta\dot{\tau}_j{}^+ = \frac{r - r_j}{r_j} > b \qquad \text{and} \qquad \Delta\dot{\tau}_{j+1}^+ = \frac{r - r_{j+1}}{r_{j+1}} < a$$

It follows that

$$(b+1)r_j < \gamma r < (a+1)r_{j+1} \tag{8.6}$$

Hence:

$$\frac{r_j}{r_{j+1}} < \frac{a+1}{b+1} = \rho \tag{8.7}$$

■

Lemma 1 gives the necessary condition for two consecutive receivers to do not belong to the same subgroup at the end of the partitioning algorithm. It follows the next corollary:

Corollary 1 *Under the precondition $G = \infty$, two consecutive receivers R_j and R_{j+1} will belong to the same subgroup if, $r_j/r_{j+1} \geq \rho$ or equivalently if, $(1 + \Delta\dot{\tau}_{j+1})/(1 + \Delta\dot{\tau}_j) \geq \rho$*

Proof: The proof is straightforward using lemma 1. Since the isolated rates are not available, using the relative RTT variation instead and knowing that:

$$\frac{r}{r_j} = 1 + \Delta\dot{\tau}_j \qquad \text{and} \qquad \frac{r}{r_{j+1}} = 1 + \Delta\dot{\tau}_{j+1} \tag{8.8}$$

Dividing the first by the second equation of (8.8) then the sufficient condition so two consecutive receivers R_j and R_{j+1} belong to the same subgroup, can be written:

$$\frac{r_j}{r_{j+1}} = \frac{1 + \Delta\dot{\tau}_{j+1}}{1 + \Delta\dot{\tau}_j} \geq \rho \tag{8.9}$$

■

Algorithm 1 converges if G groups are already constructed or when the remaining receivers (i.e. those in P_0) have so similar capacities such that a split is no more possible. This depends on the a and b parameters as stated by the following theorem:

Theorem 1 *(Convergence Criteria) The partitioning algorithm 1 converges if (i) G subgroups are built, (ii)* $|P_0| = 1$ *or (iii)* $\forall j < N, R_j \in P_0, \dfrac{1+\Delta\dot{\tau}_{j+1}}{1+\Delta\dot{\tau}_j} \geq \rho$

Proof: The first criterion is enforced by the algorithm inputs while the second is obvious. The third criterion is a generalization of corollary 1 for every two consecutive receivers in P_0. ∎

Theorem 2 *(Lower bound guarantee on the utility function) Under the precondition* $G = \infty$*, in a fully reliable multicast session, the overall utility that results from the execution of algorithm 1 has a lower bound expressed as:* $\displaystyle\sum_{k=0}^{K-1} \alpha_{ik} \frac{1-\rho^{n_k}}{1-\rho}$

This last theorem shows that depending on ρ, the execution of the partitioning algorithm guarantees a lower bound on the session utility function independently of the receivers' isolated rates distribution.

Proof: Assuming a fully reliable multicast, the reception rate of the P_k's receivers is $g_k = \min_{R_i \in P_k} r_i = r_{k1}$. The P_k subgroup utility is

$$U_k(g_k) = U_k(r_{k1}) = \alpha_{ik} \sum_{i=1}^{n_k} \frac{r_{k1}}{r_{ki}} \tag{8.10}$$

We have:

$$\forall i, j, \qquad \frac{r_{i+j}}{r_i} = \frac{r_{i+j}}{r_{i+j-1}} \times \frac{r_{i+j-1}}{r_{i+j-2}} \times \ldots \times \frac{r_{i+2}}{r_{i+1}} \times \frac{r_{i+1}}{r_i} \tag{8.11}$$

Since we have shown that every two consecutive receivers R_i and R_{i+1} that belong to the same subgroup satisfy $\frac{r_{i+1}}{r_i} \geq \rho$. then for receivers R_i and R_{i+j} that belong to the same subgroup P_j, using (8.11) we get $\frac{r_{i+j}}{r_i} \geq \rho^j$ and (8.10) becomes:

$$U_k(r_{k1}) \geq \alpha_{ik}\left(1 + \rho + \rho^2 + \ldots + \rho^{n_k-1}\right) = \alpha_{ik} \frac{1-\rho^{n_k}}{1-\rho}$$

Finally, the overall session utility satisfies:

$$U(g_0, g_1, \ldots, g_{K-1}) \geq \sum_{k=0}^{K-1} \alpha_{ik} \frac{1-\rho^{n_k}}{1-\rho} \tag{8.12}$$

∎

We demonstrated here an interesting feature of our partitioning algorithm which consists in assuring a lower bound on the resulting utility value depending on the algorithm parameters. It is quite obvious that if we want to have a higher utility value, we have to choose a higher ρ. It is worth noting that the number of subgroups is proportional to this parameter.

Numerical Results

For numerical results about the lower bound guarantees on the utility function, we consider the particular case where the number of receivers in each subgroup is the same i.e. $\forall k, n_k = N/K$. In this case, inequation (8.12) becomes:

$$U(g_0, g_1, \ldots, g_{K-1}) \geq \frac{K}{N} \frac{1-\rho^{N/K}}{1-\rho} = U_{min} \tag{8.13}$$

where $\forall i, j, \quad \alpha_{i,j} = 1/N$ (N is the number of receivers in the whole multi-rate multicast session).

Figure 8.4 plots the minimum utility value (U_{min}) as a function of the number of the built subgroups for 12 and 48 receivers and for different values of the ρ parameter $(0.8, 0.85, 0.9,$ and $0.95)$. Figure 8.4a shows for $\rho = 0.9$, a minimum utility of 0.8 with just 2 subgroups which corresponds to 80% of the maximum utility achieved with a partition of one receiver per subgroup. For 48 receivers with $\rho = 0.9$, only 4 subgroups are sufficient to achieve a minimum utility of 80% as depicted by figure 8.4b.

Figure 8.5 shows the minimum utility as a function of the number of subgroups for different numbers of the receivers ($\rho = 0.9$). We can see that for 6 receivers the minimum utility is at least 80% which increases with the number of subgroups. For 48 receivers, we need approximately 3 and 10 subgroups to assure a minimum utility of 50% and 80% respectively. Increasing the number of receivers increases the required number of subgroups to achieve a minimum receivers satisfaction.

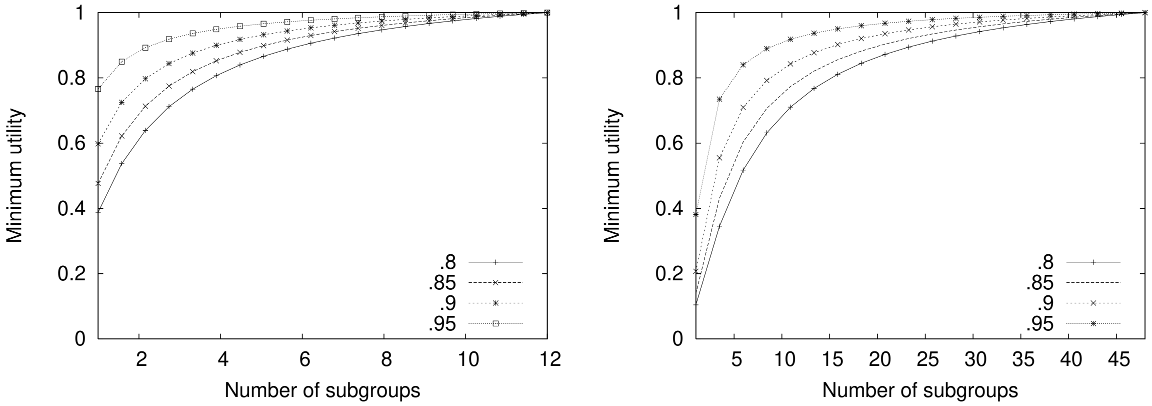
Another observation is that increasing the number of subgroups improves the utility value. Figures 8.6a and 8.6b plot the utility gain as a function of the number of subgroups for 12 and 48 receivers respectively. We can see that for 12 receivers and $\rho = 0.8$, with two subgroups we already have a gain of 160% and with 4 subgroups, the gain is about 210%. We can note that independently of ρ, the gain does not increase significantly when increasing the number of subgroups. For instance (figure 8.6b), with 5 subgroups we can achieve a gain of 2 for $\rho = 0.8$. If we double the number of subgroups from 5 to 10, the gain is improved only by 0.1.

8.2.3 Simulation Results

In order to get an insight into the proposed partitioning algorithm, simulations are performed. For a given number of receivers, the isolated rates are randomly generated following a uniform or an exponential distribution with different parameters. Our partitioning algorithm is then applied on the resulting set of generated rates and this process is repeated for different values of ρ. A large number of simulations are performed for every set of parameters in order to get stable results. For every simulation, the number of subgroups built is recorded with the corresponding utility value.

Comparing with the Optimal Solution

Here, we are concerned with comparing our partitioning algorithm performances to those of an algorithm that produces the optimal solution. The optimal algorithm consists in com-



(a) (b)

Figure 8.4: Lower bound on the session utility (a) 12 receivers and (b) 48 receivers.

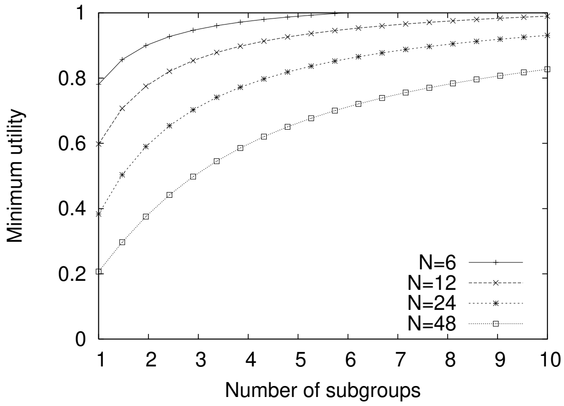
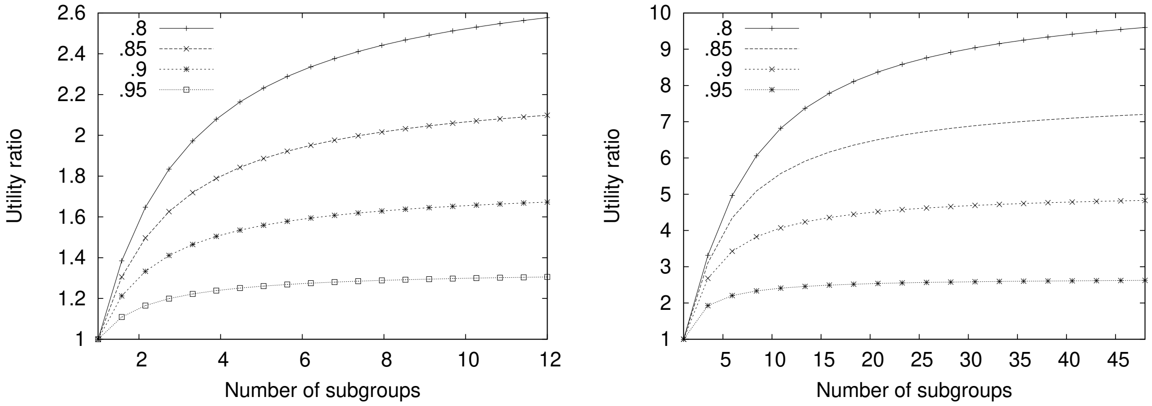


Figure 8.5: Lower bound on the session utility, $\rho = 0.9$, varying the number of the receivers.



(a) (b)

Figure 8.6: Utility gain (a) 12 receivers and (b) 48 receivers.

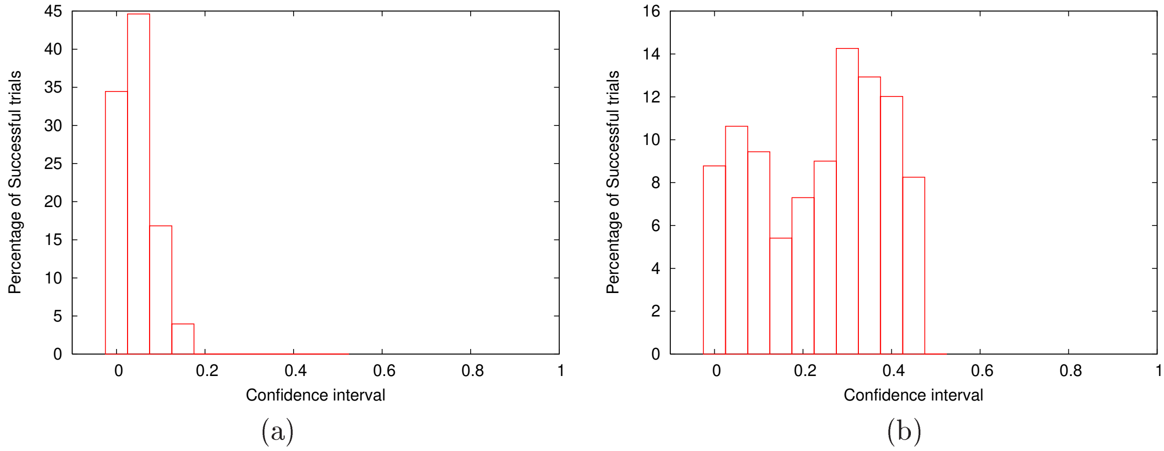


Figure 8.7: Solution optimality. Uniform distribution between (a) 5 and 10, (b) 5 and 55.

puting for each possible partition the corresponding utility value. Then the minimum utility value is saved in order to be compared with the utility achieved by our partitioning algorithm. A large number (about 1500) of simulations have been performed on two sets of rates. The first set consist in rates generated following a uniform distribution between 5 and 10. The second set presents a higher degree of heterogeneity with a uniform distribution between 5 and 55. We have limited ourselves to a partition of only two subgroups.

Figure 8.7a shows that for the first set of trials, 35% of the trials achieved the optimal solution (this corresponds to the box centered in 0). The three other boxes correspond to the percentage of trials for which the obtained utility is 95%, 90% and 85% of the optimal solution. An interesting remark is that all the obtained utility values are at least 85%. The results of increasing the heterogeneity (the second set of simulations) are shown in figure 8.7b. In this case, the partitioning algorithm gives a larger range of utility values. The reason is mainly the fact that our partitioning is performed on-the-fly and setting the maximum number of subgroups to 2 limits the performance of the algorithm. Actually the algorithm could take the decision to split the original set of receivers without having an accurate estimation of the capacity of the remaining receivers. This behavior depends on the ρ parameter which has to be well chosen to reflect the heterogeneity degree.

Figure 8.8 is another way to compare the two sets of simulations for which the cumulative percentage of success is plotted as a function of the decreasing degree of the approximability to the optimal solution. We have at least 35% and 9% of trials with the optimal solution for the first and second sets respectively. All of the trials achieve at least 80% and 55% of the optimal utility value for the first and second set of simulations.

We showed here that our partitioning algorithm achieves or at least approximates the optimal solution in many cases without knowing the receivers capacities. It is worth noting that our algorithm was limited by the number of subgroups set to 2.

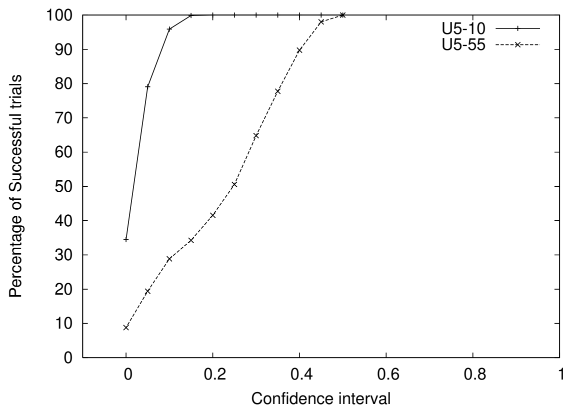


Figure 8.8: Solution optimality: comparison of the two sets of results.

Varying the number of subgroups

All the figures of this section show the different points that correspond to the resulting pairs of utility value (or the utility gain) and the number of subgroups. The utility gain is computed with respect to the case where there is only one subgroup as a reference. The curves of figure 8.9a1 and 8.9a2 show the results of the execution of our algorithm with 12 receivers for which the isolated rates are uniformly distributed between 5 and 10. Figure 8.9a1 confirms the numerical results. The mean utility according to figure 8.4a is 0.4 and 8.9a1 shows a minimum utility value of approximately 0.6. We can also see that with only 2 subgroups, we can achieve a mean utility of 0.9. In figure 8.9a2, we can see the influence of increasing the number of subgroups. We note that with 2 subgroups, we can increase the receivers' satisfaction by 25%. Increasing the number of subgroups beyond 3 does not increase significantly the receivers' satisfaction (a maximum gain of 1.4 instead of 1.3 with 10 subgroups!). We have similar results with 48 receivers as depicted by figures 8.9b1 and 8.9b2.

Figure 8.10 presents the results of having more heterogeneous receivers whose isolated rates are uniformly distributed between 5 and 55. Compared to the first set, with the same number of subgroups, we note that the utility value decreases due to the higher heterogeneity degree. However, we still achieve a utility value of 0.8 with only 3 subgroups for 12 receivers and with 8 subgroups for 48 receivers. Once again, increasing the number of subgroups beyond a given threshold does not provide significant improvements. We can conclude that independently of the receivers heterogeneity degree, we do not need to increase the number of subgroups beyond a given threshold which is proportional to the receivers heterogeneity degree. Figure 8.11 plots the same statistics but with isolated rates distributed exponentially with a mean of 5. The curves show that significant gains are achieved with a reasonable number of subgroups.

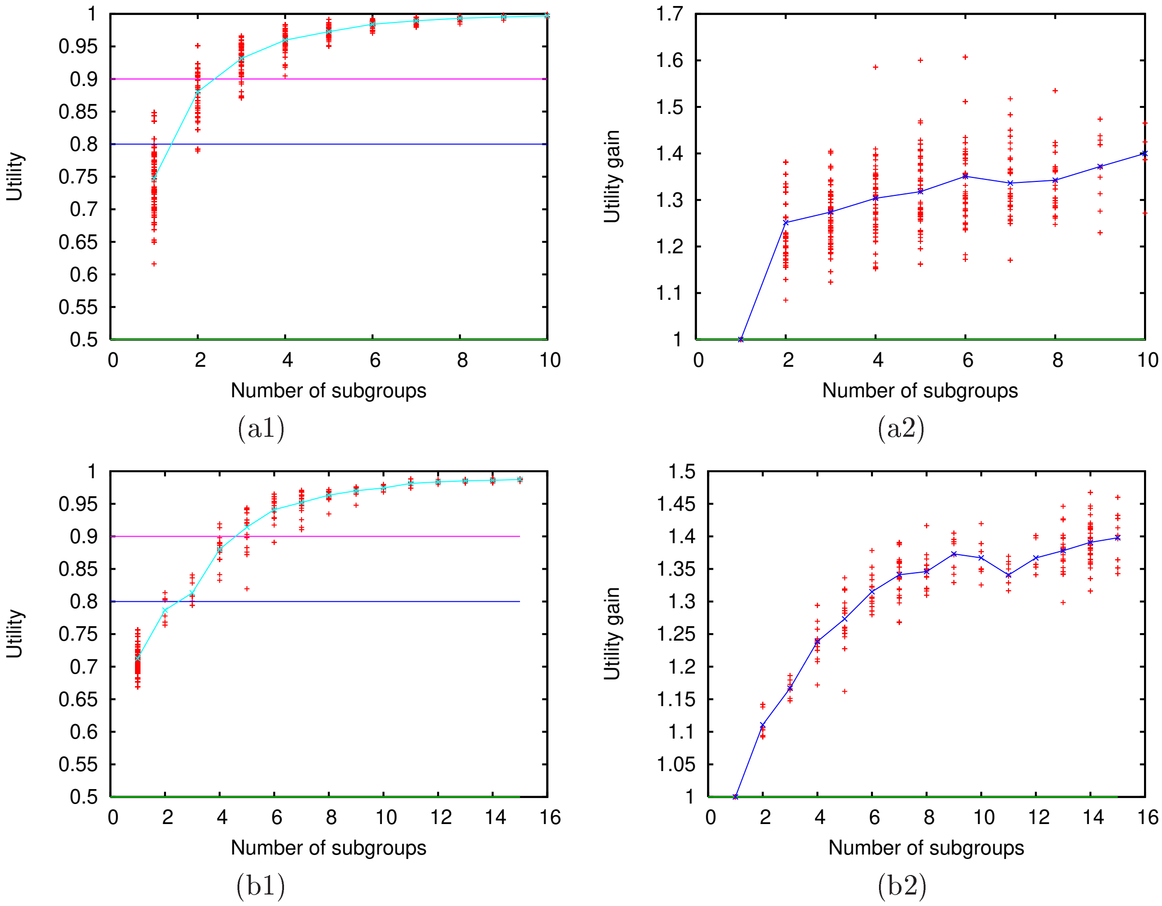


Figure 8.9: Utility and utility gain for receivers' isolated rates distributed uniformly between 5 and 10. (a) 12 receivers and (b) 48 receivers.

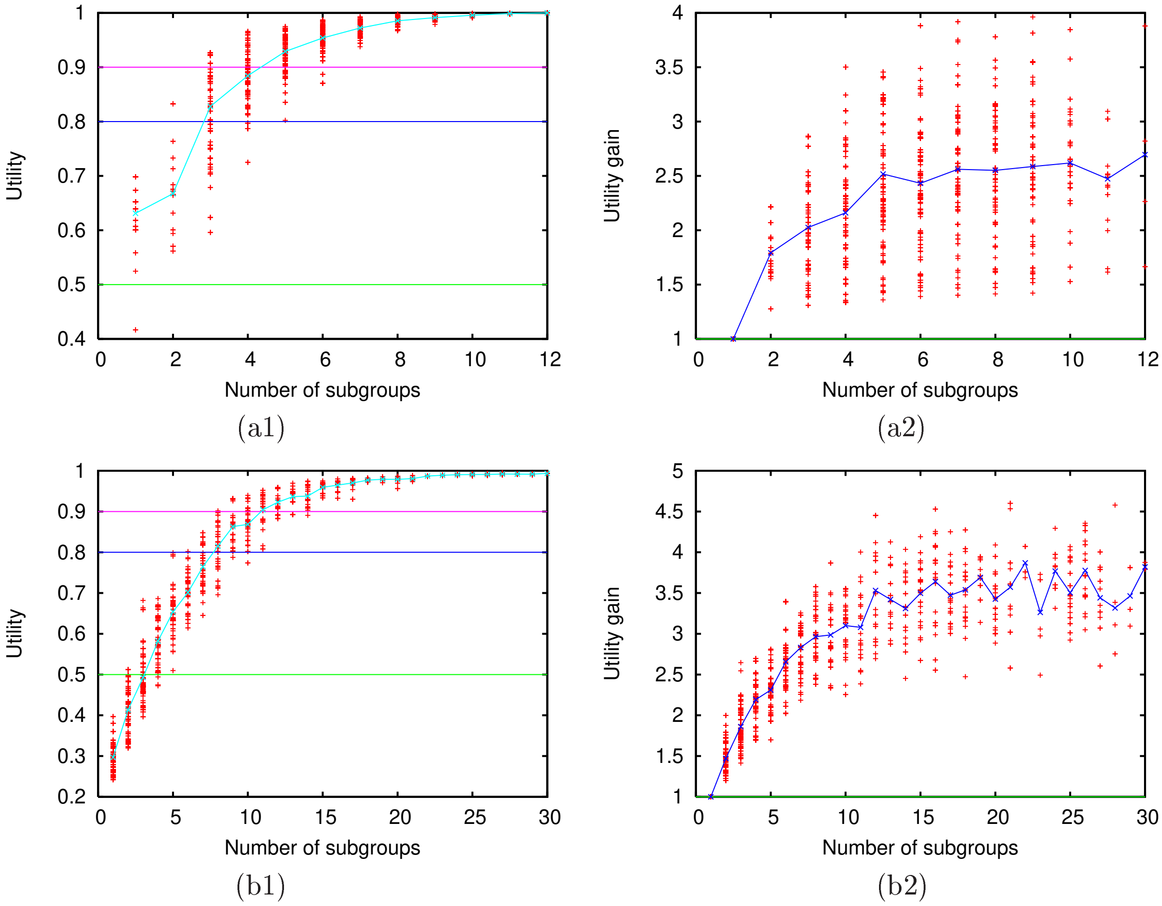


Figure 8.10: Utility and utility gain for receivers' isolated rates distributed uniformly between 5 and 55. (a) 12 receivers and (b) 48 receivers.

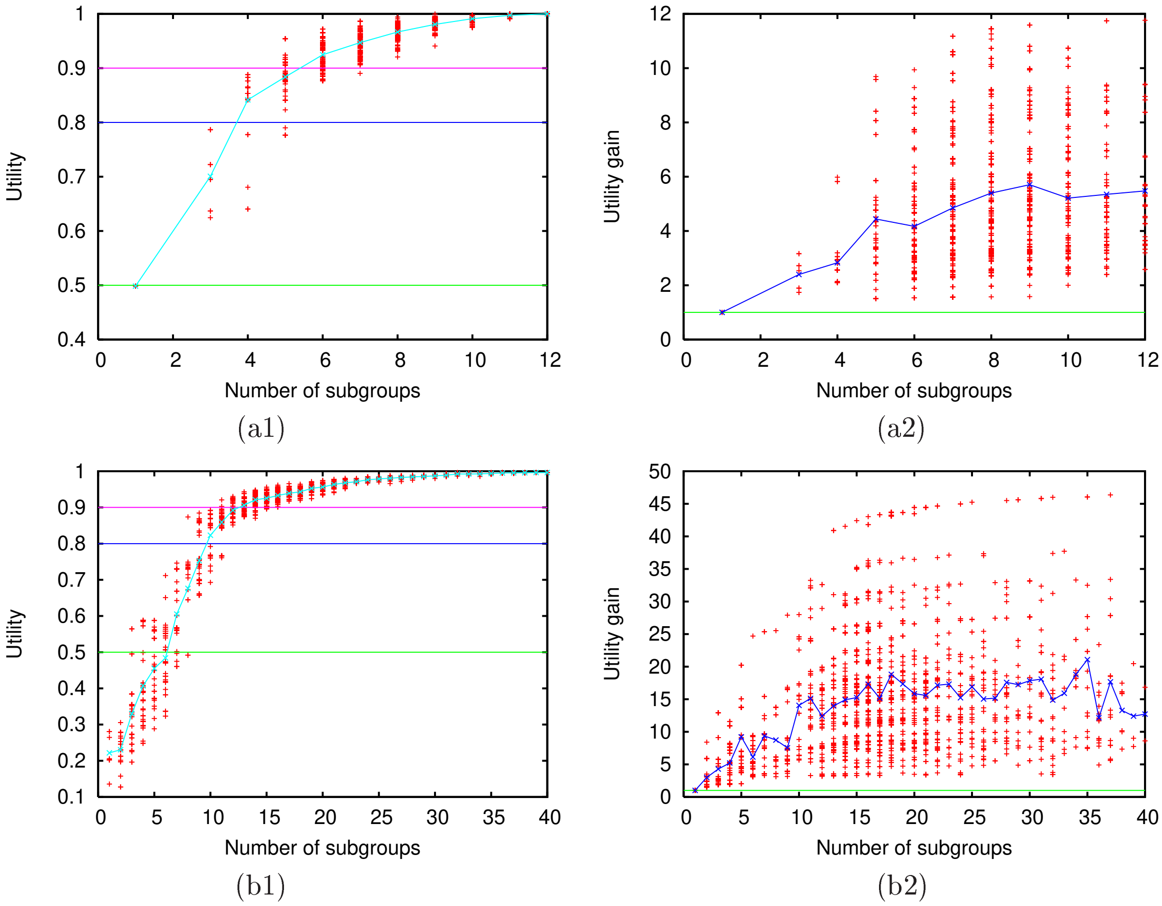


Figure 8.11: Utility and utility gains for an exponential distribution with mean 5. (a) 12 receivers and (b) 48 receivers.

8.3 A Receiver-based Replication Scheme

In the previous section, the RTT-based partitioning algorithm is executed by the source. For the implementation of our multi-rate scheme, we adopt a replicated scheme due to its simplicity. However, the source and its vicinity will be overwhelmed if data replication is done by the source. In order to be more scalable, we propose to imply the receivers in the process of data replication. A receiver that belongs to a subgroup of higher capacity could perform data replication to all the receivers of a lower capacity subgroup. In such a way, a regulation tree will be built where some receivers replicate data with an appropriate rate to other ones with lower capacity.

In what follows, we investigate the possibility of involving the routers in the construction of the regulation tree. This allows for more scalability in addition to build a regulation tree with respect to the physical multicast tree. The partitioning algorithm is no longer executed by the source. In a distributed way, each router performs locally its partitioning algorithm to build a piece of the overall regulation tree. The distributed process of building the regulation tree is provided in the next subsection followed by some illustrative examples. A comparative study of the source-based and receiver-based replication schemes using both analysis and simulations is also performed and some results are presented.

8.3.1 The Regulation Tree Construction

The regulation tree construction is a distributed process where each router performs locally a partitioning of its downstream links into subgroups and designates a replicator for every subgroup formed. Algorithm 2 shows how a router contributes in building such a regulation tree on-the-fly depending on the RTT variation feedback it receives. We do not show the transmission rate adjustment in this algorithm since it is the role of the source and the replicators. A replicator as will be seen, has to adjust its replication rate depending on feedback it receives from its children receivers.

Initially, a router maintains a list P_0 that contains all of its downstream links. Every time, a downstream link (j) reports a relative RTT variation $(\Delta\dot{\tau}_j)$ greater than parameter b, the router creates a new partition P_i with all the links that reported a relative RTT variation greater than parameter a and selects a replicator Rep_i for this subgroup. We note by $Rd(P_i)$ the set of direct receivers located downstream of the P_i links. The function $Best(P_i)$ returns for subgroup P_i, the identity of a receiver from $Rd(P_i)$ which has the highest estimated capacity. A receiver capacity is evaluated using a metric that will be presented in the context of AMCA, in section 8.5.1. When subgroup P_i is split from P_0, then $Best(P_0)$ is elected as its replicator. Since this replicator may have been chosen for P_{i-1} in the previous split, $Best(P_i)$ is definitely elected as the P_{i-1}'s replicator (see algorithm 2).

In our approach, a router forwards the source data packets only on the P_0 links. Every time, a new subgroup P_i is formed and the corresponding replicator Rep_i is selected, the router notifies this latter to start performing data replication. How this notification could be performed is addressed in section 8.5.1. A replicator Rep_i sends its replicated data packets to the receivers of its corresponding subgroup P_i. Feedback messages from subgroup P_i are sent to their corresponding replicator Rep_i while those arriving on the P_0 links are forwarded to

Algorithm 2 Regulation tree construction at a router

Require: $N > 1$ and $a < b$

$P_0 \leftarrow \{l_j, j = 1, \ldots, N\}$, the set of all the links downstream
$i \leftarrow 1$
Periodically,
if $\exists j, l_j \in P_0$ such that $\Delta\dot{\tau}_j > b$ **then**
  $P_i \leftarrow \{l_j \in P_0, \Delta\dot{\tau}_j > a\}$
  $P_0 \leftarrow P_0 - P_i$
  $Rep_i \leftarrow Best(P_0)$
  if $i > 1$ **then**
    $Rep_{i-1} \leftarrow Best(P_i)$
  end if
  $i \leftarrow i + 1$
end if
until no split is possible

the source. Once the regulation tree is constructed, the source (and every replicator) sends data to its children with a rate that matches their minimum isolated rate. Thus allowing for independent fine-grained congestion control at both the source and the replicators. Any single-rate congestion control can be used, and hence we do not rely on a specific congestion control algorithm to implement our approach. Section 8.5 provides details about the extension of AMCA (presented in chapter 7) to support more heterogeneous receivers.

A replicator has to perform caching of data packets to be retransmitted to its children. It removes data packets from its cache as soon as it receives acknowledgments for them. As have been mentioned, every receiver sends feedback to its replicator (or the source for the P_0 receivers). Feedback would contain essentially information about RTT variation and reception status. More details about how this approach can be implemented in the case of AMCA are given in section 8.5

8.3.2 Illustrative Examples

Example 1. "A star topology"

To illustrate our regulation tree construction algorithm, we consider a multicast session with seven subscribed receivers $\{R_0, R_1, R_2, R_3, R_4, R_5, R_6\}$ with respectively the following isolated rates $\{5, 6, 9, 11, 15, 19, 21\}$. All of these receivers are located downstream the same router A (see figure 8.12). If we take $a = 0.01$ and $b = 0.26$ which correspond to $\rho = 0.8$, executing algorithm 2 produces $\{\{R_0, R_1\}, \{R_2, R_3\}, \{R_4\}, \{R_5, R_6\}\}$ as a partition with the regulation tree shown in figure 8.12a. The resulting subgroups are $P_0 = \{R_5, R_6\}$, $P_1 = \{R_0, R_1\}$, $P_2 = \{R_2, R_3\}$, $P_3 = \{R_4\}$ with respectively the reception rates, $g_0 = 19$, $g_1 = 5$, $g_2 = 9$, $g_3 = 15$, which gives a session utility of 0.936 instead of 0.525 if no split is performed ($\forall i, \alpha_i = 1/7$).

The selected replicator for the first split subgroup $P_1 = \{R_0, R_1\}$ is $Best(P_0) = R_6$. In the next split, when the second subgroup P_2 is formed with $Best(P_0) = R_6$ as the replicator then the P_1 replicator (according to the proposed algorithm) has to be changed to $Best(P_2) = R_3$ which is definitely elected as a replicator for P_1. When $P_3 = \{R_4\}$ is split, its chosen replicator

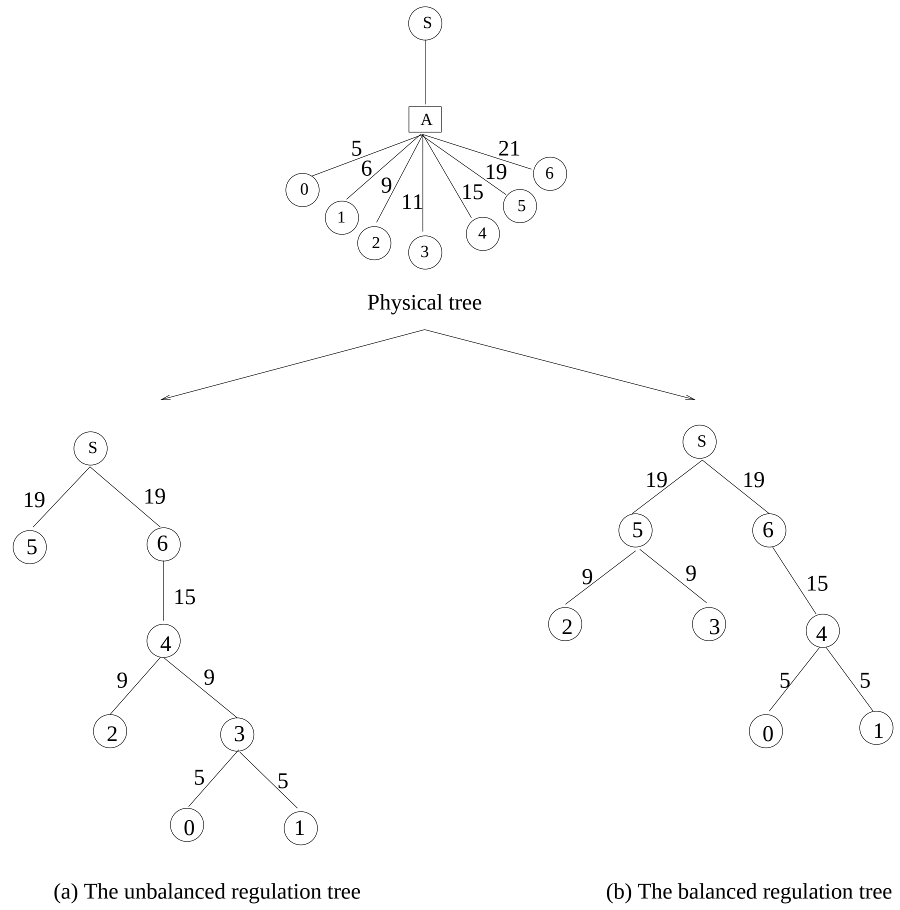


Figure 8.12: Example 1.

is $Best(P_0) = R_6$ and $Best(P_3) = R_4$ is definitely selected as a replicator for P_2 ...etc. We can note that the built tree is not balanced. For instance, receiver R_5 could be the replicator for P_2 as shown in figure 8.12b. To have a more balanced tree, we can use another rule for selecting the replicators. In fact, when partition P_i is formed, a replicator is chosen as follows:

$$Rep_i = Best\left(\bigcup_{k>i,k=0} P_k - \{Rep_k, k > i\} \right) \tag{8.14}$$

A balanced tree has the advantage of introducing less initial delay while a non-balanced tree requires less buffering requirements at the replicators since their reception and replication rates are closer to each other.

Example 2. "A hierarchy of routers"

In order to understand better our distributed partitioning mechanism, we consider one source multicasting to six receivers through three routers A_1, A_2 and A_3 (figure 8.13). A maximum bandwidth is given for every link in the considered multicast tree. The link (A_2,R_4) has a bandwidth of x which for the following will be set either to 10 or 2. The main concern here is the behavior of router A_1 since it has two indirect receivers R_3 and R_4 in addition to two

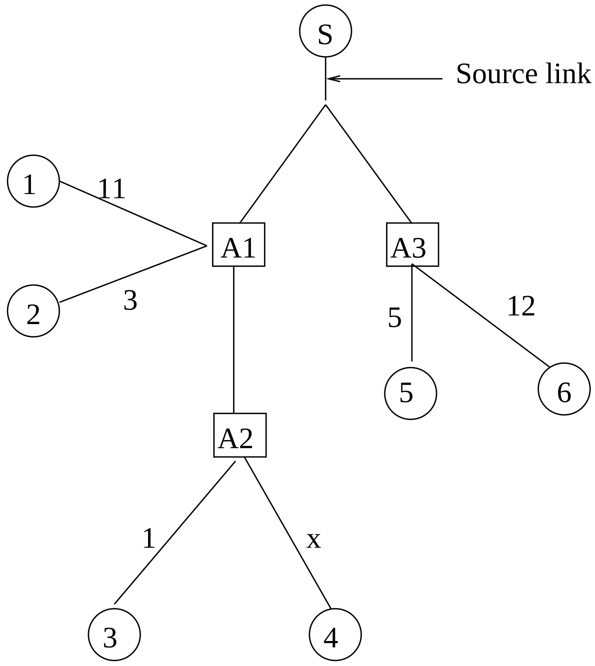


Figure 8.13: Example with a hierarchy of routers

direct ones, R_1 and R_2. In what follows, a link will be designated by its downstream node in the multicast tree. For example the A_1 downstream links that lead to router A_2 and receiver R_1 are respectively noted A_2 and R_1. We also use $P_{i,j}$ to designate the ith partition of router A_j.

Suppose that the maximum number of subgroups that could be built by a router is 2. If we set x to 10, then we get the following local partitions for the routers:

$$\begin{aligned} P_{0,1} = & \{R_1, A_2\},\ P_{1,1} = & \{R_2\} \\ P_{0,2} = & \{R_4\},\ P_{1,2} = & \{R_3\} \\ P_{0,3} = & \{R_6\},\ P_{1,3} = & \{R_5\} \end{aligned}$$

which gives the following overall partition seen by the source (figure 8.14a):

$$P_0 = \{R_1, R_4, R_6\},\ P_1 = \{R_2\},\ P_2 = \{R_3\},\ P_3 = \{R_5\} \tag{8.15}$$

This partition (8.15) gives a utility value, $U_a = (1/1 + 3/3 + 5/5 + 10/10 + 10/11 + 10/12)/6 = 0.957$ instead of 0.301 if no partitioning is performed ($\forall i, \alpha_i = 1/6$). In this latter case the source would transmit data at a rate equal to 1 instead of 10.

If we consider that the partitioning is performed by the source instead of the routers, then the optimal partition with two subgroups is $P_0 = \{R_3, R_2, R_5\}$ and $P_1 = \{R_1, R_4, R_6\}$ which gives a utility value of $U'_a = 0.713 < U_a = 0.957$. In order to get a utility value equal to $U_a = 0.957$, a source-based partitioning requires four subgroups. In this case, the source link will be loaded with $10+1+3+5 = 19$ instead of 10 since the source will send four flows with rates 10, 1, 3 and 5. This gives $19/10 = 1.9$ of additional consumed bandwidth at the source

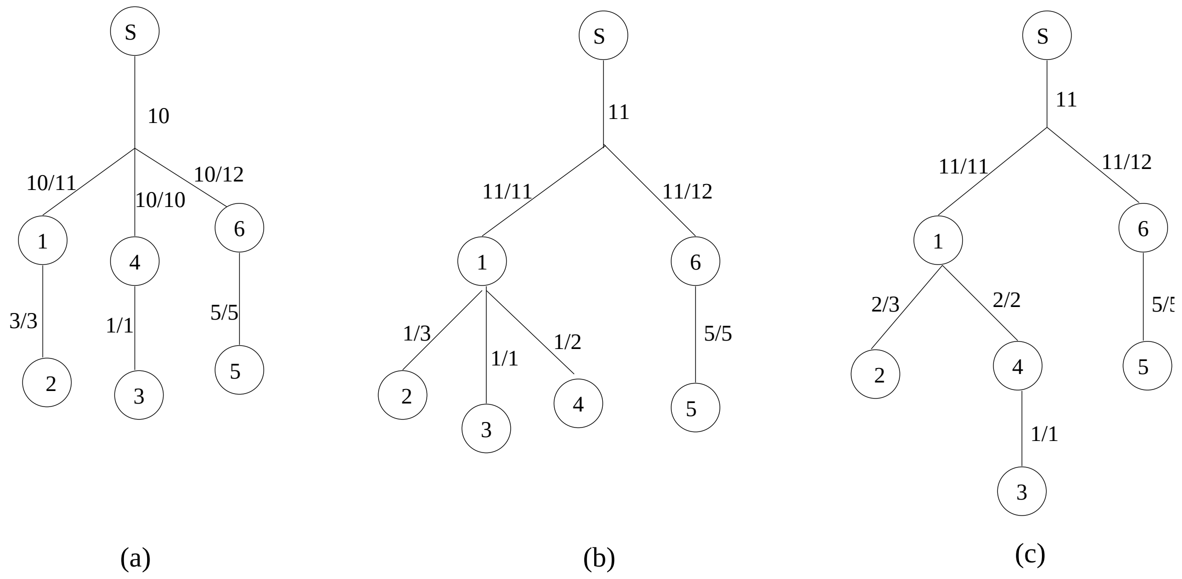


Figure 8.14: (a) $x = 10$, (b) $x = 2$, $\rho < 0.5$ and (c) $x = 2$, $\rho > 0.5$

link.

If x is set to 2, then depending on ρ, we get respectively for $\rho < 0.5$ and $\rho > 0.5$:

$$\begin{aligned} P_{0,1} = &\ \{R_1\},\ P_{1,1} = \{R_2, A_2\} \\ P_{0,2} = &\ \{R_3, R_4\} \\ P_{0,3} = &\ \{R_6\},\ P_{1,3} = \{R_5\} \end{aligned}$$

and

$$\begin{aligned} P_{0,1} = &\ \{R_1\},\ P_{1,1} = \{R_2, A_2\} \\ P_{0,2} = &\ \{R_4\},\ P_{1,2} = \{R_3\} \\ P_{0,3} = &\ \{R_6\},\ P_{1,3} = \{R_5\} \end{aligned}$$

that gives respectively the following overall partitions:

$$P_0 = \{R_1, R_6\}, P_1 = \{R_2, R_3, R_4\}, P_2 = \{R_5\} \tag{8.16}$$

and

$$P_0 = \{R_1, R_6\}, P_1 = \{R_2, R_4\}, P_2 = \{R_3\}, P_3 = \{R_5\} \tag{8.17}$$

When $x = 2$, the optimal source-based partition with two subgroups is $P_0 = \{R_2, R_4, R_3, R_5\}$ and $P_1 = \{R_1, R_6\}$ which gives a utility value of $U_b^{'} = 0.667$ instead of $U_b = 0.792$ or $U_c = 0.931$ for the partitions (8.16) (figure 8.14b) and (8.17) (figure 8.14c) respectively.

Example 3. "Mean load and its distribution" Here we consider the same topology as the previous example. However, we are interested in comparing the two replication schemes in terms of the consumed bandwidth per link in addition to its distribution on the overall

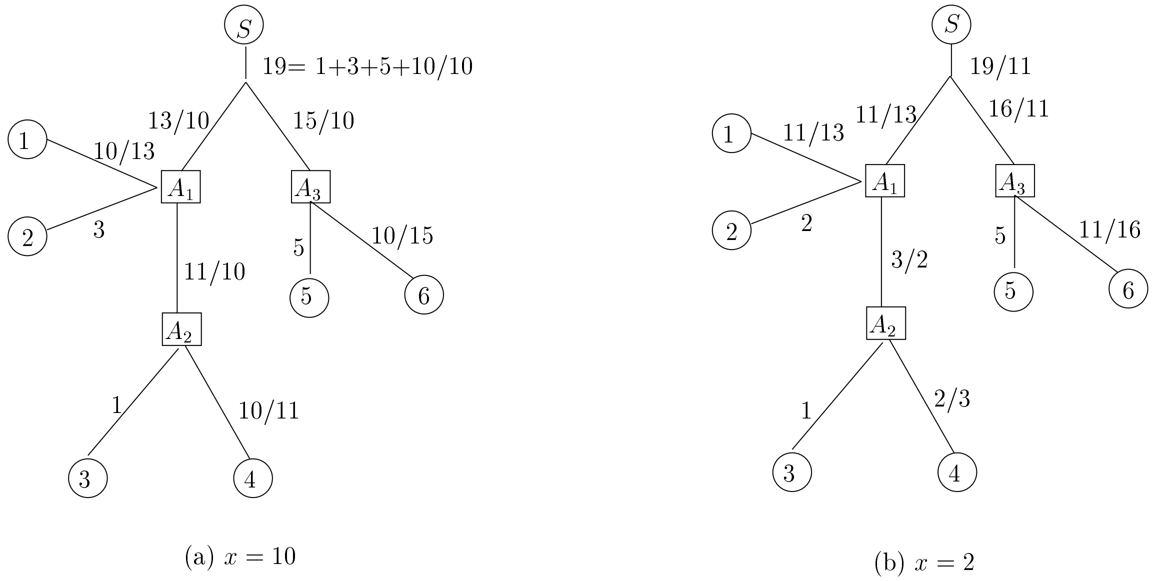


Figure 8.15: Consumed bandwidth per link, (a) $x = 10$, (b) $x = 2$

Policy	Utility value		Mean load		Load distribution	
	$x = 2$	$x = 10$	$x = 2$	$x = 10$	$x = 2$	$x = 10$
Single-rate	0.368	0.301	1.0	1.0	0.0	0.0
Sender-based	0.667	0.713	8.1	9.7	27.01	33.41
Receiver-based	0.931	0.957	7.5	8.8	17.56	26.85

Table 8.1: Example 2. Utility and load distribution

multicast tree. Figure 8.15 shows the consumed bandwidth per link using notation X_S^i/X_R^i where X_S^i and X_R^i are the bandwidth consumed by link i in the source and receiver-based schemes respectively. Table 8.3.2 summarizes the utility, mean load and its distribution among links in the different policies. We note that a receiver-based approach consumes less bandwidth than a source-based one. Moreover, a receiver-based replication allows for more load distribution among the different links. This is very important regarding the impact on fairness towards other flows as will be investigated in section 8.4. We will show analytically in section 8.3.4 in the case of a star topology that a receiver-based replication scheme allows for more load distribution among the multicast tree links.

8.3.3 Lower Bound on the Utility Function

To get an insight into the impact of executing the partitioning algorithm at the routers instead of the source, we derive a lower bound on the utility value in this case. We use U_R and U_S to designate respectively the utility value obtained in a sender and a receiver-based replication. We consider the worst case topology that consists in one level of routers (figure 8.16). One source multicasts data to N receivers through M routers. Downstream each router $A_m, (1 \leq m \leq M)$, there are $B, B > 2$ receivers. Let K $(2 < K < B)$, be the number of

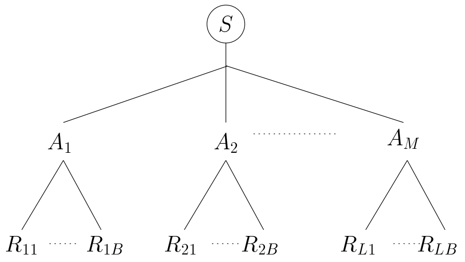


Figure 8.16: Network Model.

subgroups built by every router, then the overall number of subgroups will be $M(K-1)+1$ instead of K in case the partitioning algorithm is executed by the source. It is obvious that a network topology with more hierarchy levels would produce a larger number of subgroups.

In a receiver-based replication, each router A_m contributes to form $(K-1)$ subgroups $(P_{km}, 1 \leq k \leq K-1)$ in addition to a subset of the source subgroup P_{0m}. Without loss of generality, assume that the built subgroups by the routers are of equal sizes (B/K). Based on the minimum utility expression (8.13), the contribution of router A_m in the overall utility value satisfies[3]:

$$U_R(g_1^m, g_2^m, \ldots, g_{K-1}^m) \geq \frac{K-1}{N} \frac{1-\rho^{B/K}}{1-\rho} \tag{8.18}$$

where $\forall i, k \ : \alpha_{ik} = 1/N$. The overall utility value when considering all the routers can then be bounded as follows:

$$U_R(\underline{g}_1, \underline{g}_2, \ldots, \underline{g}_{K-1}, g_0) \geq M \ \frac{K-1}{N} \frac{1-\rho^{B/K}}{1-\rho} + U_R(g_0) \tag{8.19}$$

where $\underline{g}_k = (g_k^m)_{m=1..M,k=1..K}$ and g_k^m is the kth replication rate at the mth router A_m. g_0 is the source transmission rate dictated by receivers in $P_0 = \bigcup_{m=1,M} P_{0m}$ and $U_R(g_0)$ is the utility value contributed by the P_0 subgroup. The P_0 subgroup results from the application of the partitioning algorithm at all the routers. Hence, there is no explicit relation between the isolated rates of the P_0 receivers. Due to that, we are not able to provide an expression or even a lower bound for $U_R(g_0)$. However and knowing that $U_R(g_0) > 0$, we can write the following:

$$U_R(\underline{g}_1^m, \underline{g}_2^m, \ldots, \underline{g}_{K-1}^m, g_0) > M \ \frac{K-1}{N} \frac{1-\rho^{B/K}}{1-\rho} = U_R min \tag{8.20}$$

Recall that the utility value in the sender-based replication case is:

$$U_S(g_0, g_1, \ldots, g_{K-1}) \geq \frac{K}{N} \frac{1-\rho^{N/K}}{1-\rho} = U_S min \tag{8.21}$$

[3]We only consider the $K-1$ subgroups, since the P_{0m} has to be considered only once in the overall utility value

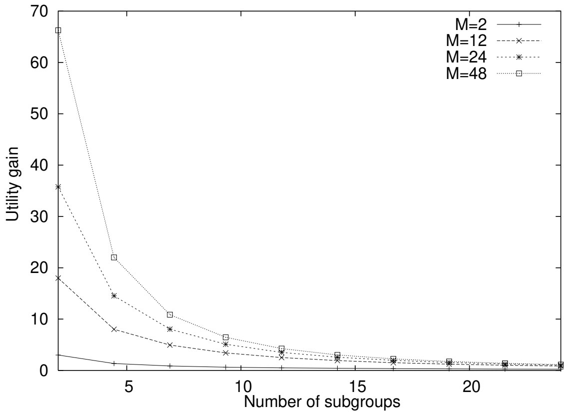


Figure 8.17: Utility gain of a receiver-based replication when increasing the number of subgroups per router, $N = 2304$.

The gain in receivers' utility in the receiver-based replication with respect to a source-based scheme can be expressed as follows:

$$\frac{U_R min}{U_S min} = M \; \frac{K-1}{K} \; \frac{1-\rho^{B/K}}{1-\rho^{N/K}} \tag{8.22}$$

We are mainly interested in the impact of increasing K, the number of the subgroups built, on the gain achieved by a receiver-based replication when the partitioning is performed by the routers. We assume that M and B are of fixed values, then $U_R min/U_S min$ decreases when K increases. For the largest value that K could take (B), we get:

$$\frac{U_R min}{U_S min} = M \; \frac{B-1}{B} \; \frac{1-\rho}{1-\rho^{M}} \tag{8.23}$$

We have

$$lim_{B\to\infty}\frac{U_R min}{U_S min} = M \; \frac{1-\rho}{1-\rho^{M}}$$

where $M \; \frac{1-\rho}{1-\rho^{M}}$ is an increasing function of M. That is the gain of a receiver-based replication increases with the number of routers contributing in the execution of the partitioning algorithm. However, increasing the number of subgroups per router do not improve further the receivers satisfaction compared to a sender-based approach. Figure 8.17 plots for a fixed number of receivers ($N = 2304$), the gain as a function of K for different values of M. We easily observe that independently of the number of the routers, the gain in utility decreases when increasing the number of subgroups per router.

Figure 8.18 plots the gain in utility when the number of receivers increases, with only two subgroups built per router. We can see that the gain in utility due to our receiver-based replication increases with the number of receivers. For instance with 2000 receivers located downstream 48 routers, we can improve the session utility with a factor of 20.

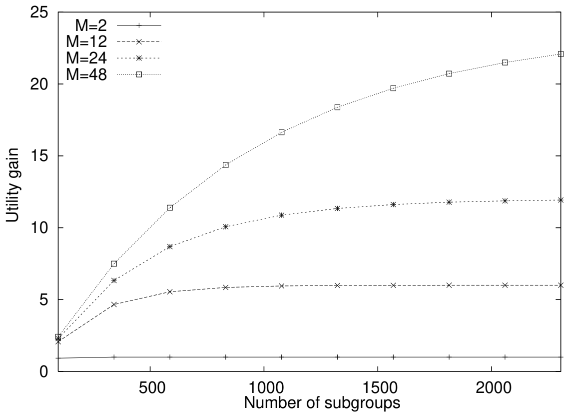


Figure 8.18: Utility gain of a receiver-based replication when increasing the number of the receivers, $K = 2$

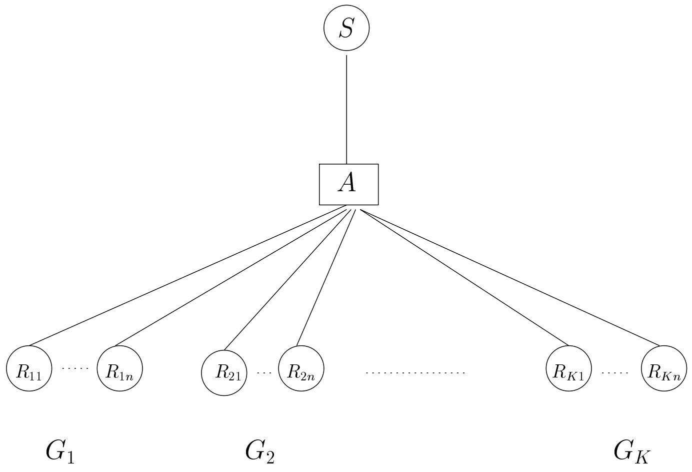


Figure 8.19: Star topology

8.3.4 Load Distribution

The main drawback of a replicated multi-rate scheme is that it wastes bandwidth due to data replication for every subgroup of receivers. We argue that our receiver-based approach allows for bandwidth saving compared to the classical sender-based scheme. In a sender-based replication, the source is responsible for sending multiple data flows to subgroups with members which could be distributed among distant subtrees. This could result in data replication on multiple links of the multicast tree. We have shown in example 3 of section 8.3.2, how a receiver-based replication could be better in terms of consumed bandwidth especially in the presence of a hierarchy of routers. In the case of a star topology (which is one of the worst cases), we will show that the consumed bandwidth is the same. However in terms of load distribution, a receiver-based replication allows for more load balanced bandwidth consumption among the different multicast tree links. This has a great impact on the fairness properties of a receiver-based replication as will be studied in section 8.4.

We consider the star topology of figure 8.19 with one source multicasts data through one router, to Kn receivers distributed on K equisized subgroups $G_1, G_2, \ldots, G_K$ with replication rates of $g_1 < g_2 < \ldots < g_K$. X_S and X_R are two random variables that give the bandwidth consumption per link in a sender and a receiver-based replication scheme respectively. We derive the mean of the two random variables and their variation. This latter permits the evaluation of the load distribution among links in the two different approaches.

In a sender-based scheme, the consumed bandwidth on the source link equals the sum of all the replication rates $\sum_{i=1}^{K} g_i$. Every tail link of subgroup G_i is crossed by a flow of rate g_i. Noting by $L = N + 1 = nK + 1$, the total number of links, the mean consumed bandwidth can be expressed as follows:

$$\begin{aligned} E[X_S] &= \frac{(\sum_{i=1}^{K} g_i + n \sum_{i=1}^{K} g_i)}{L} \\ &= \frac{n+1}{L} \sum_{i=1}^{K} g_i \end{aligned} \tag{8.24}$$

In the case of a receiver-based replication, the consumed bandwidth on the source equals g_K, the highest replication rate. A replicator Rep_i ($i = 1..K-1$) consumes its replication rate g_i in addition to its own reception rate g_{i+1}. The other receivers consume just their respective reception rate. All the n receivers of G_1 (the worst subgroup) receives data with rate g_1. In a subgroup G_i, $i = 2..K$, there is $n-1$ receivers each of them consumes g_i of bandwidth. Following that, we find that the X_R mean value is the same as for X_S:

$$E[X_R] = \frac{n+1}{L} \sum_{i=1}^{K} g_i \tag{8.25}$$

For the mean variation computation, we consider the case where we have at least two subgroups $(K > 1)^4$. We have for X_S:

$$E[X_S^2] = \frac{(\sum_{i=1}^{K} g_i)^2 + n \sum_{i=1}^{K} g_i^2}{L} \tag{8.26}$$

thus, the mean variation can be computed as follows:

$$\begin{aligned} V[X_S] &= E[X_S^2] - E^2[X_S] \\ &= \frac{(\sum_{i=1}^{K} g_i)^2 + n \sum_{i=1}^{K} g_i^2}{L} - \frac{(n+1)^2}{L^2} (\sum_{i=1}^{K} g_i)^2 \\ &= \frac{(L - (1+n)^2)(\sum_{i=1}^{K} g_i)^2 + nL \sum_{i=1}^{K} g_i^2}{L^2} \end{aligned} \tag{8.27}$$

In the case of a receiver-based replication, we have:

[4] For $K = 1$, we have $V[X_R] = V[X_S] = 0$

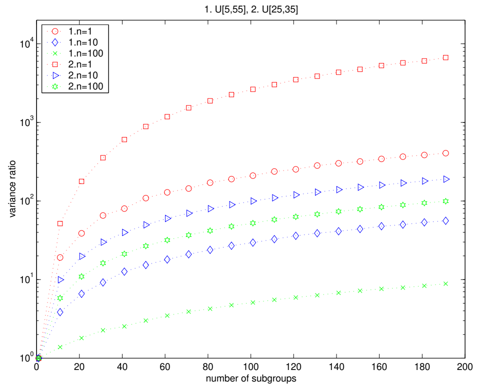


Figure 8.20: $V[X_S]/V[X_R]$ as a function of the number of subgroups (1=U[5,55], 2=U[25,35]).

$$E[X_R^2] = \frac{(n+1)\sum_{i=1}^{K} g_i^2 + 2\sum_{i=1}^{K-1}(g_i g_{i+1})}{L} \tag{8.28}$$

and thus:

$$\begin{aligned} V[X_R] &= E[X_R^2] - E^2[X_R] \\ &= \frac{(n+1)L\sum_{i=1}^{K} g_i^2 + 2L\sum_{i=1}^{K-1}(g_i g_{i+1}) - (n+1)^2(\sum_{i=1}^{K} g_i)^2}{L^2} \end{aligned} \tag{8.29}$$

For the numerical results, figures 8.20 and 8.21 plot the gain in the load distribution ($V[X_S]/V[X_R]$) when a receiver-based replication is adopted instead of a sender-based scheme, and this is done for two different distributions of the receivers' isolated rates. We generate the isolated rates following a uniform distribution of parameters 5,55 and 25,35 giving the same mean but different mean variations for different heterogeneity degrees. Figure 8.20 plots for the two distributions, the gain achieved when varying the number of subgroups. The number of receivers per group is set to 1, 10 and 100. First, we observe that the ratio is always greater than 1 justifying the benefit from the receiver-based replication on better load distribution. Moreover, the gain increases when increasing the number of subgroups. This is due to the fact that in a sender-based replication scheme, the source link will be more and more loaded since it transports all the data flows replicated by the source. Regarding the degree of heterogeneity, it is clear that the benefit from the receiver-based approach is more significant in the presence of more heterogeneous receivers.

Figure 8.21 plots the load distribution ratio as a function of the number of receivers per subgroup for the two rates distributions. The number of subgroups is set to 10 and 20. We can see that the load distribution gain decreases when the number of receivers per subgroup increases. This is quite normal since increasing the number of receivers results in a decreasing number of subgroups.

8.3.5 Replication Cost

In the previous section, we showed that the mean bandwidth consumption in a star topology is the same in the two replication schemes. However, we did not take into consideration the

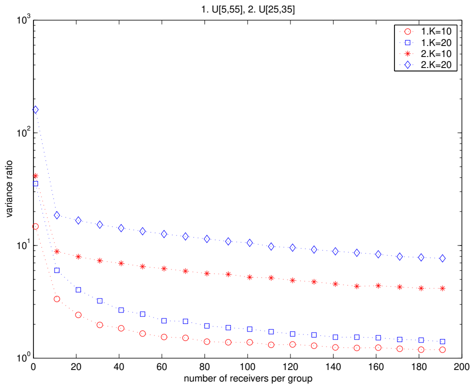


Figure 8.21: $V[X_S]/V[X_R]$ as a function of the number of receivers per subgroup (1=U[5,55], 2=U[25,35]).

cost that can be assigned to the different links of the multicast tree due to data replication. Here we are interested in evaluating the replication cost in terms of the link costs. We always consider the star topology of figure 8.19. We use random variables C_S and C_R to represent the replication cost of the sender-based and the receiver-based schemes respectively. We have:

$$\begin{aligned} L(E[C_S]-E[C_R]) &= c_s \textstyle\sum_{i=1}^{K} g_i - [c_s g_K + \sum_{i=1}^{K-1} c_i g_i] \\ &= c_s(\textstyle\sum_{i=1}^{K} g_i - g_K) - \sum_{i=1}^{K-1} c_i g_i \\ &= \textstyle\sum_{i=1}^{K-1}(c_s - c_i) g_i \end{aligned} \tag{8.30}$$

When the costs assigned to the source (c_s) and tail links ($c_i, 1 \leq i \leq N$) are equal, the replication cost is the same in both the sender and receiver-based approaches. In what follows, we assign to the replier link Rep_i, a cost of $c_i = 1/g_{i+1}$ which is the inverse of the maximum supported rate by this link.

Theorem 3 *(Replication cost) A receiver-based replication scheme is better than a source-based replication scheme in terms of replication cost if* $c_s > c(\underline{g}) = \frac{\sum_{i=1}^{K-1} g_i/g_{i+1}}{\sum_{i=1}^{K-1} g_i}$*. Moreover* $c(\underline{g})$ *has an upper bound* $c_m(\underline{g}) = \frac{K-1}{KE[G]-g_1}$ *and* $\lim_{K\to\infty} c_m(\underline{g}) = \frac{1}{E[G]}$ *where* $E[G]$ *is the mean of the different subgroups transmission rates.*

Based on this theorem, a receiver-based replication is better than a source-based replication scheme when the source link has a cost greater than a given threshold that depends on the distribution of the subgroups rates. This threshold has an upper bound which is the inverse of the subgroups mean rates.

Proof:
Equation 8.30 can be written:

$$\begin{aligned} L(E[C_S]-E[C_R]) &= c_s \textstyle\sum_{i=1}^{K-1} g_i - \sum_{i=1}^{K-1} c_i g_i \\ &= c_s \textstyle\sum_{i=1}^{K-1} g_i - \sum_{i=1}^{K-1} \frac{g_i}{g_{i+1}} \end{aligned} \tag{8.31}$$

It follows that a receiver-based replication is better in terms of replication cost when,

$$c_s > c(\underline{g}) = \frac{\sum_{i=1}^{K-1} g_i/g_{i+1}}{\sum_{i=1}^{K-1} g_i} \tag{8.32}$$

On the other hand, we have:

$$\frac{g_i}{g_{i+1}} < 1 \Rightarrow \sum_{i=1}^{K-1} \frac{g_i}{g_{i+1}} < K - 1$$

and when K is sufficiently large,

$$\sum_{i=1}^{K-1} g_i = K \frac{\sum_{i=1}^{K} g_i}{K} - g_1 = KE[G] - g_1$$

then

$$c(\underline{g}) < \frac{K-1}{KE[G] - g_1} = c_m(\underline{g})$$

$c_m(\underline{g})$ is an increasing function of K and $lim_{K\to\infty} c_m(\underline{g}) = 1/E[G]$. In summary:

$$c_s > \frac{\sum_{i=1}^{K-1} g_i/g_{i+1}}{\sum_{i=1}^{K-1} g_i} < \frac{K-1}{KE[G] - g_1} < \frac{1}{E[G]}$$

The receiver-based replication consumes more bandwidth if the source link has a cost less than all the replicator links. A replicator has to be chosen from direct receivers with the lowest cost.

We have the following corollary:

Corollary 2 *The sufficient condition so a receiver-based replication scheme is better than a source-based replication scheme in terms of replication cost is the following:* $c_s > \dfrac{1}{E[G]}$.

$1/E[G]$ decreases when $E[G]$ increases. A receiver-based replication is better if the replication transmission rate are high.

Figure 8.22 plots the minimum cost of the source link from which a receiver-based approach is better in terms of replication cost in a star topology (figure 8.19). Subgroups' transmission rates are generated randomly following uniform (figure 8.22a) and exponential (8.22b) distributions with different parameters giving different degrees of heterogeneity. We can see that all the curves are around 1 showing a source link cost less than 10. We can also see that when the heterogeneity degree increases, the minimum cost decreases showing that a receiver-based approach is likely to be better in terms of replication cost than a source-based one in very heterogeneous networks which is the case of the current Internet.

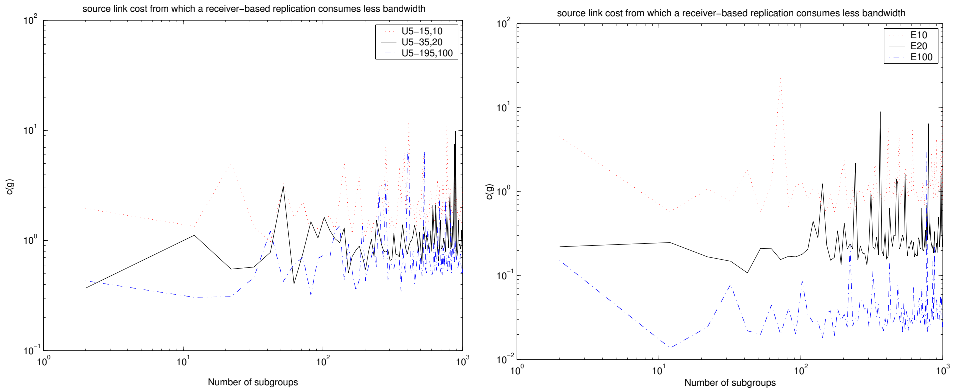


Figure 8.22: Minimum source link cost from which a receiver-based approach is better than a source-based one, (a) uniform and (b) exponential distributions.

8.3.6 Simulation Results

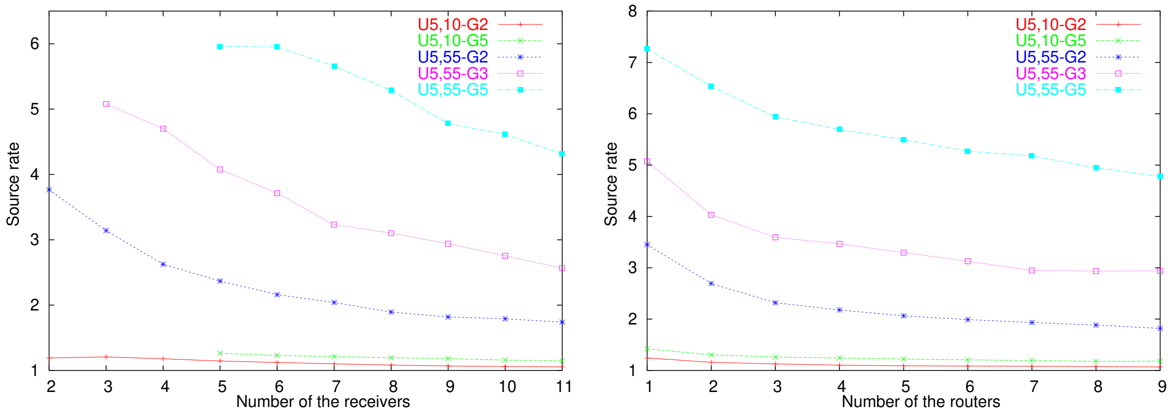
In this section, we evaluate our receiver-based replication scheme using simulations. For that, we use the same network model of section 8.3.3 depicted in figure 8.16. The isolated rates of the receivers are generated following different distribution laws with different parameters, thus allowing for obtaining different heterogeneity degrees. The generated rates are randomly distributed among M sets of size B, each of which is associated to a router. The regulation tree construction algorithm is applied on these rates for different values of $\rho > 0.6$. Each router executes its partitioning algorithm with G as the maximum number of subgroups to be built. Every simulation with a given set of parameters is repeated multiple times to allow for mean estimations of the different metrics of this study.

First we are concerned with the benefit of the receiver-based approach compared to a single-rate approach. After that a comparison study of the sender and receiver-based approaches is performed.

Receiver-based Replication Versus Single-rate

In a single-rate congestion control, the source adapts its rate according to the worst receiver in the multicast group. Partitioning the receivers among subgroups allows for the source to increase its transmission rate. In our receiver-based replication scheme, the source transmits with a rate that matches the slowest receiver in P_0 (containing the fast receivers) instead of the worst receiver in the whole multicast session. In order to show how the source transmission rate is never dictated by the worst receiver, we plot in figures 8.23a and 8.23b, the ratio of the sender rate in the receiver-based replication to the single rate scheme as a function of the number of receivers per router (B) and the number of routers (M) respectively. This is done for different rate distributions and different G parameters.

We can see that independently of the number of routers and their receivers, the rate ratio is always greater than 1. For instance, figure 8.23a shows that the source rate can be multiplied by 2.5 in the presence of 11 receivers per router with just a maximum number of subgroups $G = 3$ for a uniform rates distribution with parameters 5 and 55. Recall that



(a) (b)

Figure 8.23: The increase in the source rate as a function of (a) the number of the receivers per router (B) and (b) the number of routers (M)

G is the maximum number of subgroups per router which does not mean that every router effectively builds G subgroups. Moreover, we can see that the ratio is more significant when the heterogeneity degree is greater. For a maximum number of subgroups ($G = 2$) and eight receivers per router ($B = 8$), the source rate (figure 8.23a) is not increased significantly for the less heterogeneous set of rates (uniform parameters 5 and 10) while it is doubled for a more heterogeneous set of rates (uniform parameters 5 and 55).

Figure 8.23a shows that the rate ratio decreases when the number of receivers per router increases. This is not a drawback of our approach since we estimate that in a multicast tree, the number of downstream links per router is limited. Furthermore, the ratio decreases also when M, the number of routers increases. This is quite normal, since increasing the number of routers increases the number of receivers that belong to the source subgroup P_0.

Receiver-based Versus Sender-based Replication

Hereafter, we are interested in comparing our receiver-based replication to the sender-based one. Figure 8.24 plots the gain in utility due to the receiver-based approach as a function of the number of routers involved in the partitioning process for $B = 10$ and different G parameters. Receivers rates are generated uniformally between 5 and 10 (figure 8.24a) and 5 and 55 (figure 8.24b). We can see that independently of the heterogeneity degree, the utility gain increases with the number of routers. For $G = 2$ and 9 routers, the gain in utility is 1.4 when the receivers heterogeneity is higher (figure 8.24b) instead of 1.04 (figure 8.24a)

Figure 8.25 plots the utility ratio for a set of rates distributed following an exponential law with mean 30 which is of a greater heterogeneity degree than the previous uniform distributions. Once again, we see that the gain is more significant. For instance for $G = 2$, we get a gain of 2.4 instead of 1.4 and 1.04 for the two previous cases. Finally, curves of figure 8.25 show that increasing G (the maximum number of subgroups per router) beyond a given threshold does not increase significantly the gain in utility.

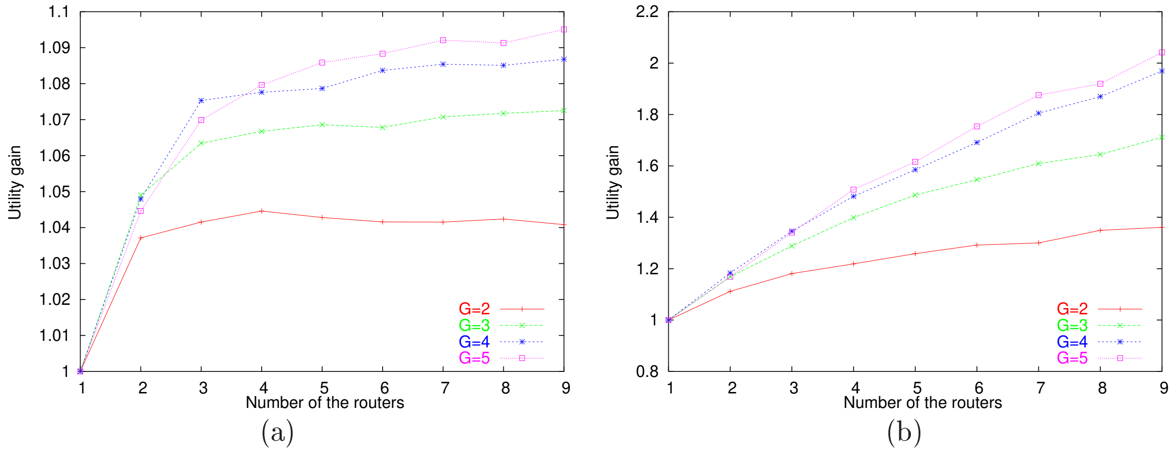


Figure 8.24: Utility gain as a function of the number of routers, $B = 10$ and rates generated randomly following a uniform distribution with parameters (a) 5 and 10, (b) 5 and 55.

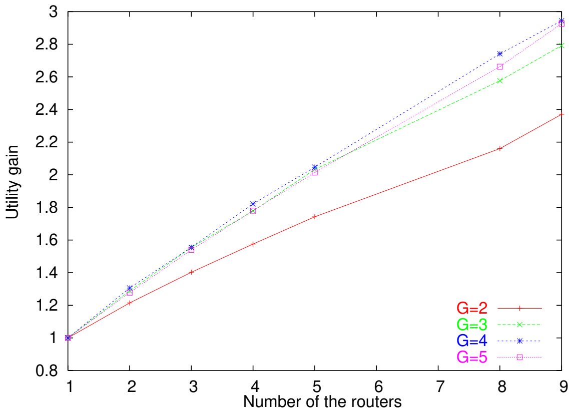
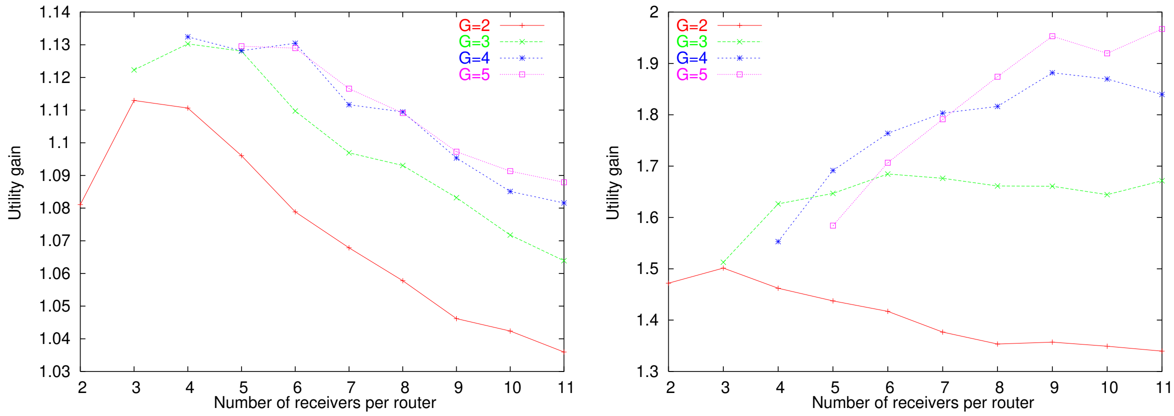
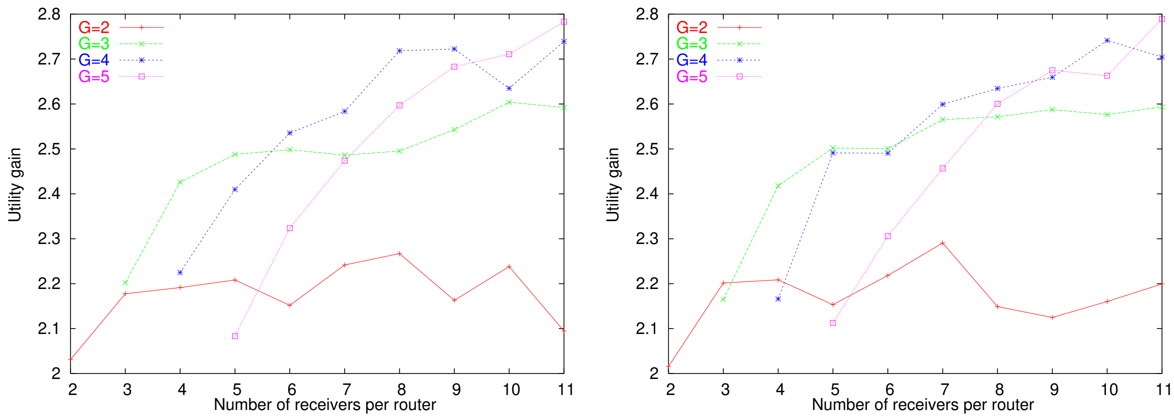


Figure 8.25: Utility gain as a function of the routers M, $B = 10$ and rates are generated following an exponential distribution with mean 30.



(a) (b)

Figure 8.26: Utility gain as a function of B, $M = 8$, rates are generated uniformally between (a) 5 and 10, (b) 5 and 55.



(a) (b)

Figure 8.27: Utility gain as a function of B, $M = 8$, rates are generated following an exponential distribution with means (a) 5 and (b) 30.

In order to see the impact of increasing B, the number of receivers associated to each router, figures 8.26 and 8.27 plots the gain in utility as a function of B for uniform and exponential rate distributions. We can see in figure 8.26a that the gain is near 1 and that it decreases with B. However in a more heterogeneous environment (the case of figures 8.26b and 8.27), we can observe a different behavior of the curves. In these latter cases the gain is more significant and increases with B. For instance, with $G = 3$ and 11 receivers per router, we obtain a gain of 1.7 and 2.6 for the uniform distribution (5,55) and the exponential distributions respectively.

Now, we investigate the influence of the overall number of receivers. Figure 8.28 plots for different rate distribution, the utility gain as a function of the number of receivers subscribed to the multicast session. We can see that for the least heterogeneous set of receivers (figure 8.28a), the gain does not exceed 1.12 and is almost constant even if the number of receivers

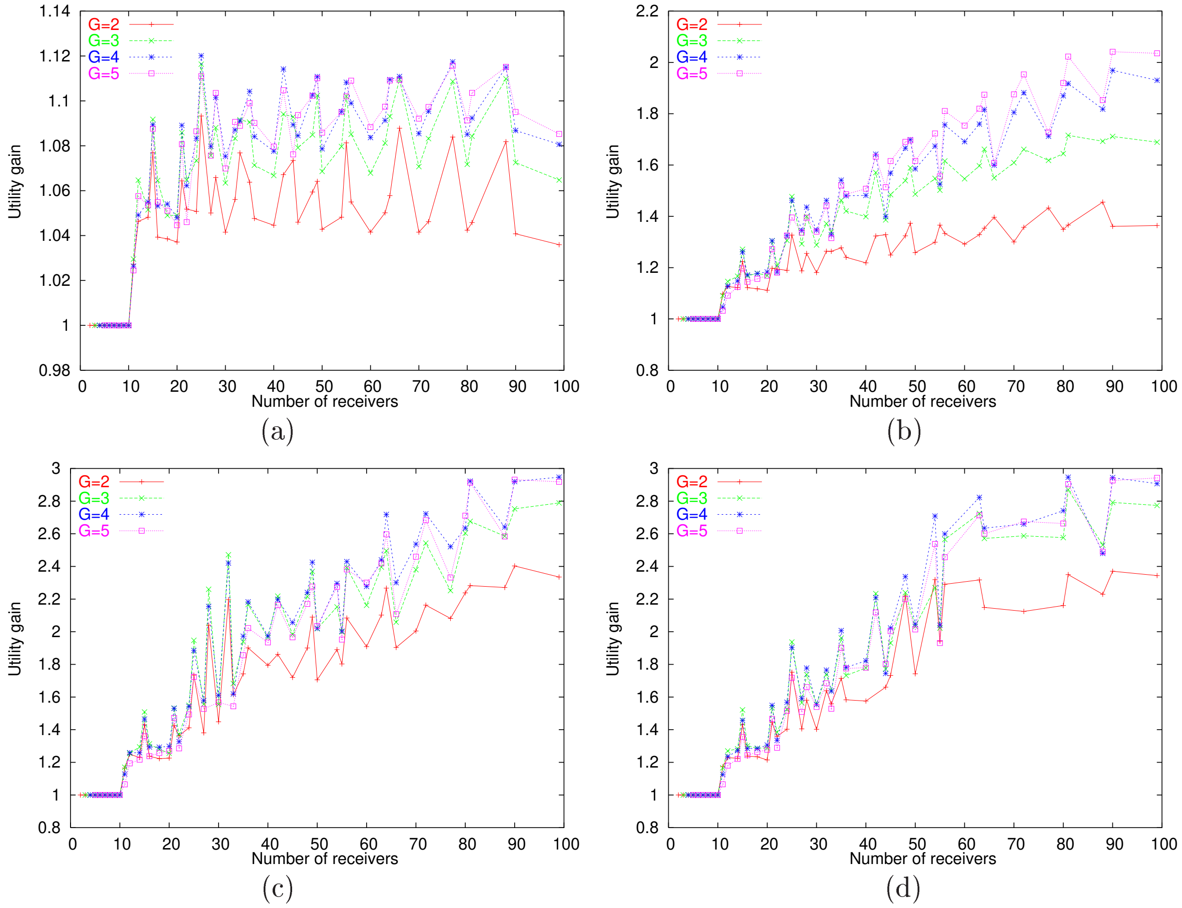


Figure 8.28: Utility gain as a function of R, (a) $U[5,10]$, (b) $U[5,55]$, (c) $E(5)$, (d) $E(30)$

increases. For more heterogeneous receivers (figure 8.28b,c,d), we observe that the benefit increases with the number of receivers which allows for more scalability. Once again, we observe that increasing the number of subgroups per router does not necessarily allow for a significant benefit.

8.4 Inter-session Fairness

In our scheme, data replication is performed by some receivers designated as a result of the execution of the partitioning algorithm at the routers. The replication tree is built with respect to the multicast tree topology. In addition to scalability, this allows to avoid the coexistence of multiple flows on the same link. This is due to the fact that receivers that belong to the same subtree are put in the same subgroup. In a source-based replication, receivers far from each other could belong to the same subgroup which could result in data replication on some common links mainly near the source. Furthermore, our approach allows for a fine-grained rate adaptation (congestion control, see section 8.5) since independently of the source and other replicators, each replicator adjusts its rate according to the feedback it receives from its associated subgroup.

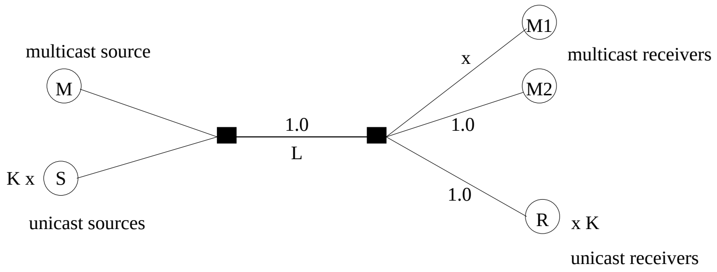


Figure 8.29: Fairness with unicast flows.

In order to show how our replication scheme could enhance the intra-session as well as the inter-session fairness, we consider the star topology of figure 8.29 where one multicast session shares one link (L) with K unicast sessions. The multicast source M multicasts data to 2 receivers M_1 and M_2. All the links have a bandwidth of 1.0 except the M_1 link which has a bandwidth of $x \leq 1.0$. We note by m_1, m_2 and s_i the (max-min) isolated rates of the receivers M_1, M_2 and R_i $(1 \leq i \leq K)$ respectively. Without loss of generality we put $\forall i, s = s_i$.

In what follows, we consider three bandwidth allocation policies that results from applying a single-rate, a multi-rate source-based replication and a multi-rate receiver-based replication scheme. For a given policy, we note by $\widehat{m}_1$, $\widehat{m}_2$ and $\widehat{s}$ the allocated rates by this policy to the multicast and the unicast receivers.

To evaluate the inter-receiver fairness in the multicast session, we use as a fairness measure, the utility function (8.4) introduced in section 8.2.1. For the considered topology of figure 8.19, the inter-receiver fairness of the multicast session can be given by:

$$U = 0.5 \left(\frac{\widehat{m}_1}{m_1} + \frac{\widehat{m}_2}{m_2}\right) \tag{8.33}$$

where $\forall i$, we set $\alpha_i = 0.5$. For the purpose of evaluating both unicast and multicast receivers utility (inter-session fairness), we use a global measure similar to the one proposed in [62] called the *global deviation measure*. However ours measures the deviation of allocated bandwidth to the different receivers from their isolated rates, that is:

$$GD = \frac{\sum_{i=1}^{C_m} \sum_{k=1}^{n_i} \frac{r_{k,i} - \widehat{r}_{k,i}}{r_{k,i}} + \sum_{j=1}^{C_u} \frac{s_j - \widehat{s}_j}{s_j}}{\sum_{i=1}^{C_m} (n_i)^y + C_u} \tag{8.34}$$

where :

- C_u and C_m are the number of unicast and multicast sessions.
- n_i is the number of receivers in the multicast session i.
- $r_{k,i}$ is the isolated rate of the kth receiver of the ith multicast session.
- s_j is the max-min rate of the jth unicast receiver
- $\widehat{r}_{k,i}$ and $\widehat{s_j}$ are the allocated rates for the kth receiver in the multicast session i and jth unicast session respectively.

Policy	M_1	M_2	R	U	$GD, y = 0$	$GD, y = 1$
Isolated rates	0.1	0.25	0.3	-	-	-
Single-rate	0.1	0.1	0.3	0.7	0.15	0.12
Sender-based	0.1	0.2	0.23	0.9	0.1056	0.0844
Receiver-based	0.1	0.25	0.25	1.0	0.0417	0.0333

Table 8.2: Star topology, $x = 0.1$ and $K = 3$

- $y \in]0, 1[$ is used to obtain different relative weightings of multicast session.

8.4.1 Example

We consider a special case of the star topology (figure 8.19) where the first multicast receiver link bandwidth x is set to 0.1 and the number of unicast sessions K is 3. This example is the same as the one given in [61]. Table 8.2 summarizes for each of the considered policies, the utility U achieved by the multicast session and the global deviation GD.

In the presence of the three unicast receivers, M_1 would get an isolated rate of 0.1. M_2 would get $1/4 = 0.25$ if unconstrained by M_1. The three unicast receivers each would get $1 - 0.1)/3 = 0.3$.

In a single-rate multicast, M_1 and M_2 gets a rate of 0.1 since they are limited by M_1 isolated rate. The three other receivers each gets a rate of 0.3. In the case of a source-based replication scheme, the five receivers share the bottleneck bandwidth. That is, each of them would get 0.2. However M_1 is limited to 0.1 and thus the remaining bandwidth is supplied to the unicast receivers $(0.23 = (1 - 0.1 - 0.2)/3)$.

In the receiver-based replication, M_1 is limited by 0.1 and M_2 competes fairly with the unicast receivers and all of them get a reception rate of 0.25.

We observe that the multicast session utility achieves its highest value when the replication is performed by a receiver (here M_2). We also note that the minimum deviation is assured by our approach. All the receivers (unicast or multicast) get in general the reception rate which is the closest to their isolated rates.

In what follows, we will consider the general case and see if the previous observation hold when varying K and x for each bandwidth allocation policy, we derive an expression for the different proposed metrics. We will use U_P and GD_P to note respectively the multicast session utility and the global deviation in the policy $P \in \{Sg, S, R\}$ for respectively a single-rate, source-based and receiver-based replication.

8.4.2 Isolated Rates

The isolated rates of the different receivers depend on the number of the unicast sessions K and the M_1 upstream link bandwidth x. The bottleneck link L is shared by $(K + 1)$ sessions (one multicast and K unicast sessions). Every session would get $1/(K + 1)$ of bandwidth unless $x < 1/(K+1)$ (or $K < (1-x)/x$). In this latter case, M_1 would get only x, M_2 will get $\frac{1}{K+1}$ and the unicast receivers will share the remaining bandwidth. As a result, we get:

$$m_1 = \begin{cases} x & \text{if } K < \dfrac{1-x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \qquad m_2 = \frac{1}{K+1} \qquad s = \begin{cases} \dfrac{1-x}{K} & \text{if } K < \dfrac{1-x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \tag{8.35}$$

8.4.3 Single-rate Multicast

In a single-rate multicast, all the receivers subscribing to the same multicast session would get the same reception rate dictated by the worst receiver. Since the bottleneck link is shared with K other unicast sessions, we would get a reception rate of $1/(K+1)$ for all of the receivers. However if $x < 1/(K+1)$ then the multicast receivers would get a reception rate of x and the other unicast receivers would share the remaining bandwidth. We get the following:

$$\widehat{m} = \widehat{m}_1 = \widehat{m}_2 = \begin{cases} x & \text{if } K < \dfrac{1-x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \qquad \widehat{s} = \begin{cases} \dfrac{1-x}{K} & \text{if } K < \dfrac{1-x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \tag{8.36}$$

We have,

$$\frac{\widehat{m}_1}{m_1} = 1 \qquad \frac{\widehat{m}_2}{m_2} = \begin{cases} x(K+1) & \text{if } K < \dfrac{1-x}{x} \\ 1 & \text{otherwise} \end{cases}$$

giving,

$$U_{Sg} = \begin{cases} \dfrac{xK + (x+1)}{2} & \text{if } K < \dfrac{1-x}{x} \\ 1 & \text{otherwise} \end{cases} \tag{8.37}$$

For the global deviation, we have:

$$GD_{Sg} = \begin{cases} \dfrac{1 - xK - x}{2^y + K} & \text{if } K < \dfrac{1-x}{x} \\ 0 & \text{otherwise} \end{cases} \tag{8.38}$$

8.4.4 Multi-rate Source-based Replication

In a source-based replication scheme, the bottleneck bandwidth would be fairly shared by the $(K+2)$ receivers. Thus each of them would get a reception rate of $1/(K+2)$. However if $m_1 = x < 1/(K+2)$ (or $K < (1-2x)/x$, then M_1 would be limited to x and the other unicast receivers would get back the remaining bandwidth $(1 - 1/(K+2) - x)/K$. In the opposite case ($K \geq (1-2x)/x$), the multicast receivers would get the same rate ($1/(K+1)$) as the other unicast receivers. In this case there is no replication from the source since M_1 and M_2 would have the same rate. In summary:

$$\widehat{m}_1 = \begin{cases} x & \text{if } K < \dfrac{1-2x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \qquad \widehat{m}_2 = \begin{cases} \dfrac{1}{K+2} & \text{if } K < \dfrac{1-2x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases}$$

$$\widehat{s} = \begin{cases} \dfrac{1-x}{K} - \dfrac{1}{K(K+2)} & \text{if } K < \dfrac{1-2x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \tag{8.39}$$

We have:

$$\frac{\widehat{m}_1}{m_1} = \begin{cases} 1 & \text{if } K < \dfrac{1-2x}{x} \\ \dfrac{K+1}{K+2} & \text{if } K = \dfrac{1-2x}{x} \\ 1 & \text{otherwise} \end{cases} \qquad \frac{\widehat{m}_2}{m_2} = \begin{cases} \dfrac{K+1}{K+2} & \text{if } K \leq \dfrac{1-2x}{x} \\ 1 & \text{otherwise} \end{cases} \tag{8.40}$$

giving the following utility for the multicast session:

$$U_S = \begin{cases} \dfrac{2K+3}{2(K+2)} & \text{if } K < \dfrac{1-2x}{x} \\ 1 & \text{otherwise} \end{cases} \tag{8.41}$$

The global deviation can be expressed as follows:

$$GD_S = \begin{cases} \dfrac{2-x}{(2^y+K)(1-x)(K+2)} & \text{if } K < \dfrac{1-2x}{x} \\ \dfrac{x}{(2^y+K)(1-x)} & \text{if } K = \dfrac{1-2x}{x} \\ 0 & \text{otherwise} \end{cases} \tag{8.42}$$

8.4.5 Multi-rate Receiver-based Replication

In our receiver-based replication scheme, unicast receivers and the fastest one (M_2) in the multicast session would share the available bandwidth. M_1 will be regulated by M_2 and would get a reception rate of x. This holds when the reception rates of M_1 and M_2 are not the same. When $m_1 = x > 1/(K+1)$ (or $K < (1-x)/x$, then M_1 and M_2 would be in the same group and would get each $1/(K+1)$. Finally:

$$\forall K,\ \widehat{s} = \widehat{m}_2 = \frac{1}{K+1} \qquad \widehat{m}_1 = min(\widehat{m}_2, x) = \begin{cases} x & \text{if } K < \dfrac{1-x}{x} \\ \dfrac{1}{K+1} & \text{otherwise} \end{cases} \tag{8.43}$$

We have,

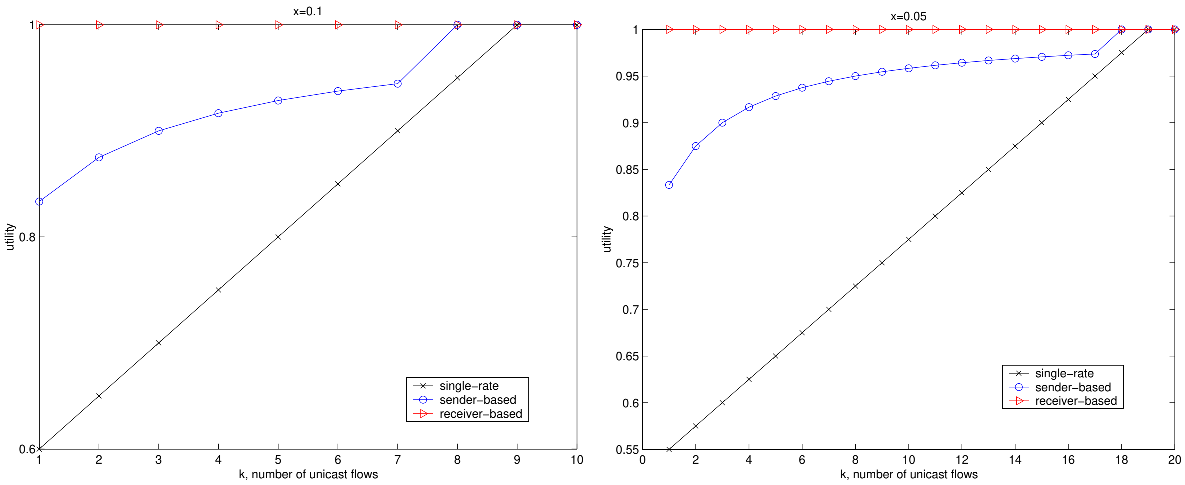
$$\forall K, x\ :\ \frac{\widehat{m}_1}{m_1} = \frac{\widehat{m}_2}{m_2} = 1 \tag{8.44}$$

giving a utility value of:

$$U_R = 1,\ \forall x, K \tag{8.45}$$

The global deviation can be expressed as follows:

$$GD_R = \begin{cases} \dfrac{1-(K+1)x}{(K+1)(1-x)(2^y+K)} & \text{if } K < \dfrac{1-x}{x} \\ 0 & \text{otherwise} \end{cases} \tag{8.46}$$



(a) (b)

Figure 8.30: Multicast session utility (a) $x = 0.1$ and (b) $x = 0.05$.

8.4.6 Numerical Results

We first examine the impact of the different approaches on the inter-receiver fairness in the multicast session. We plot in figure 8.30, the utility value achieved by the multicast session as a function of the number of the competing unicast sessions for $x = 0.1$ (figure 8.30a) and $x = 0.05$ (figure 8.30b). We can see that the receiver-based replication scheme achieves its maximum value independently of K and x. This is due to the fact that we have only two receivers and a 2-subgroup partitioning is sufficient to get this maximum utility. Even with only two receivers, the source-based replication scheme achieves the maximum utility only in the presence of a minimum number of unicast flows ($K = 8$ when $x = 0.1$ and $K = 18$ when $x = 0.05$). It is clear that the single-rate approach is the worst one. This value of K corresponds to the minimum number of unicast sessions from which there is no data replication, since the allocated rates to the multicast receivers is the same ($K \geq (1-2x)/x$). We can also see that the single-rate approach is the worst one. As we will see, unicast sessions could be very aggressive towards single-rate multicast sessions.

Figure 8.31 plots the global deviation for each of the three polocies. x is set to 0.1 in figure 8.31a and 0.05 in 8.31b while y is set to 0.5 in both of them. We observe that independently of the number of unicast flows (K) and the isolated rate (x) of the slowest receiver in the multicast session, our approach incurs the smallest deviation from the isolated rates of the different receivers. This shows that our approach allows for more inter-session fairness than the two other approaches. The largest deviation is introduced by the single-rate scheme where the unicast sessions would be aggressive with the multicast session. The global deviation decreases with the number of the unicast flows since each unicast receiver would get a smaller rate, but closer to its isolated rate.

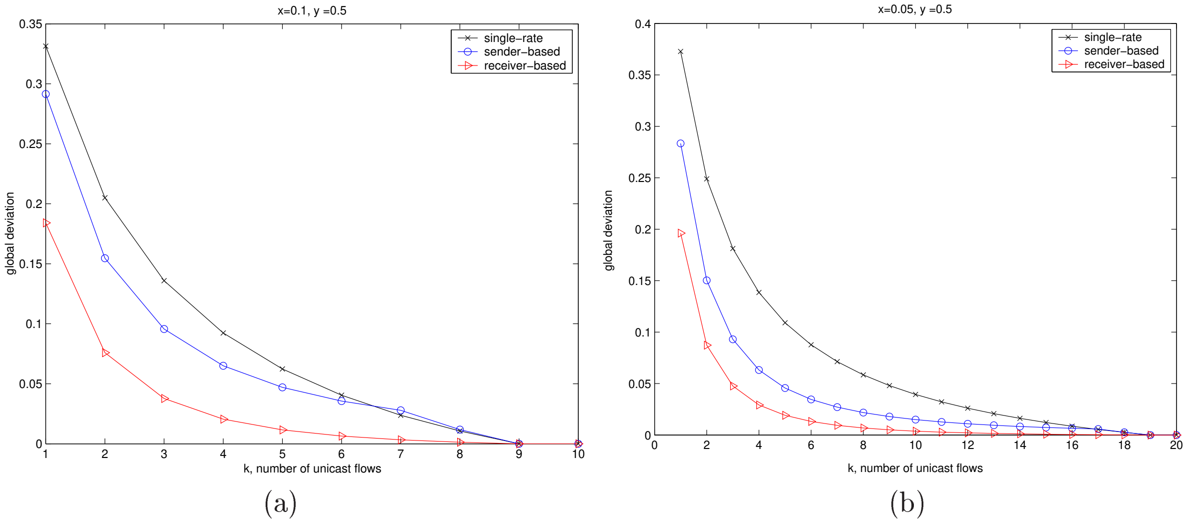


Figure 8.31: Global deviation, $y = 0.5$, (a) $x = 0.1$ and (b) $x = 0.05$.

8.5 AMCA Extension

In order to implement our receiver-based replication scheme, two approaches are possible. One approach could consist in the use of multiple multicast addresses, one per subgroup. Each replicator sends its data flow to a multicast address subscribed to by only its children in the regulation tree. In this case we need to implement a mechanism that would allow receivers (associated to a given replicator) to be aware of the multicast address on which the replicator is transmitting. For the extension of the AMCA protocol, we have adopted another approach based on the active networking technology. An active router maintains local information about the regulation tree and notifies the replicators to start the data replication. A replicator sends its flow towards its upstream active router which will forward every flow on the appropriate set of links.

The AMCA congestion avoidance protocol has been extended to implement the receiver-based replication approach using ns-2.1b8 (network simulator [46]). For practical considerations and ease of implementation, we chose to limit the number of subgroups maintained by an active router to 2. In this way, routers are not overloaded by the management of multiple subgroups. However, since every router performs locally its own partitioning procedure, the overall number of subgroups seen by the source can be much higher. For instance, if we consider a two-level multicast tree with M intermediate nodes (routers); if every router performs a 2-subgroup partition of its downstream receivers, we will get $(M + 1)$ subgroups for the whole session. This choice is also motivated by the analytical and simulation results of sections 8.3.3 and 8.3.6 respectively where it was observed that one does not need to have a large number of subgroups per router to significantly improve the receivers satisfaction. It has also been proved formally in [53] that only 2 subgroups are sufficient to achieve a 50% of the best case (one receiver per subgroup), independently of the receivers heterogeneity.

8.5.1 Regulation Tree Construction within AMCA

An active router executes the partitioning algorithm proposed in section 8.3.1 and will split the set of its downstream links into two subgroups noted H and L which contain respectively the set of links with higher and lower capacity. Initially, and according to our partitioning algorithm, H contains all the downstream links, L is empty and the router is in the "initial phase". A router exits the initial phase when a partitioning is performed and a replicator is chosen. When a router receives a CR packet indicating a relative RTT variation greater than the b parameter, then it goes through the LS structure to determine if other links have experienced a relative RTT variation greater than the a threshold. If so all those links are moved from the H set to the L set. Once the two partitions H and L are built, the active router selects one link among those belonging to H as the replicator. To avoid having replication from far receivers, replicators have to be chosen the closest possible to the active router. Only direct receivers can be elected to act as a replicator by a router. This is why we have to be careful that the H set must contain at least one link leading directly to a receiver. To make a router aware of the links with direct receivers, the LS structure (introduced in section 5.3.2) contains a $flag$ field set to 1 if this link leads to one direct receiver and to 0 otherwise. Moreover in order to be able to evaluate the relative RTT variation experienced by every downstream link during the partitioning phase, the LS structure will contain the RTT, the RTT variation ($RTTvar$) and the period ($period$) reported by the last CR received on each link. For the replicator election, we use the following metric to evaluate the capacity of the candidate links (those leading directly to a receiver):

$$weight = \kappa\ (-RTTvar) + (1-\kappa)\ \frac{1}{RTT \times (lr - lo + 1)} \tag{8.47}$$

where κ is a weighting parameter to be chosen in $]0,1[$. $RTTvar$ varies inversely with the available bandwidth on the considered link. The RTT gives an idea on the potential replicator distance to the active router. As for $(lr - lo + 1)$, it can be seen as an estimation of the loss rate experienced by the receiver located downstream this link. Information on $RTTvar$, RTT, lr and lo are retrieved from the CRs (eventually from NACKs) received on every link.

Since we have limited the number of subgroups to only 2, we just need to maintain the identity of one replicator link in addition to the H and L contents for every subgroup. In order to make a replicator aware of its election, a data packet header is extended to contain an additional field ($IsRep$). Once a replicator is designated by a router, the first data packet received is forwarded on all the downstream links with the $IsRep$ field set to 1 only for the replicator link. Since an elected link leads directly to the replicator, only this latter will receive a data packet with this field set to 1. The reason behind sending this first data packet to all the links downstream is to make the replicator aware of its election. On the other hand this data packet, when forwarded on the L links, will have its $rate$ field set to half the source current rate. This is because the replicator will begin the replication from the next data packet but with a transmission rate set to half the source current rate. This is important for setting the timeouts (especially for requesting repairs) at the receivers side.

Subsequent data packets received from the source are forwarded only on the H links. A replicator will retransmit every data packet it receives (with $IsRep$ set to 1) with an appropriate rate to its upstream router. This latter will retransmit the data packets received from downstream (from a replicator) only on the L links. For the receivers feedback, an active router forwards one aggregated CR based on the CRs received from the L links to the replicator. Similarly, the CRs received from the H receivers are aggregated into one CR to be sent upstream towards the source. The rate adjustment performed at the replicator side is the same as an AMCA sender (section 7.5). One difference consists in the fact that there is no slow-start and the replicator enters immediately in the congestion avoidance phase. Since its upstream router forwards feedback from the L receivers to it, the replicator would achieve a transmission rate which matches the slowest receiver among those located downstream the L links. The source will send data packets with a rate that matches the slowest receiver located downstream the H links.

We must mention that a replicator has to perform the cache of data packets to be retransmitted to its children. A replicator removes data packets from its cache as soon as it receives the corresponding acknowledgments (piggybacked in the CRs).

8.5.2 The Regulation Process at the Replicator's Side

A receiver, when it receives a data packet that indicates that it is the selected replicator, initializes a timer for each subsequent data packet it receives in order to schedule its retransmission to a time which corresponds to its regulation rate. For the data packet with sequence number i (see figure 8.32), a timer will be set depending on the desired reemission rate, to δ_i computed as follows:

$$\delta_i = \Delta + \delta_{i-1} - x_{i-1} \tag{8.48}$$

where x_{i-1} is the interval between the arrival of the $(i-1)$th and the ith data packets. x_{i-1} is inversely proportional to the source transmission rate. δ_{i-1} is the timeout value for the previous data packet. Δ is the reemission interval which is inversely proportional to the reemission rate. The replicator has to update the regulation timeout values every time the reemission rate changes.

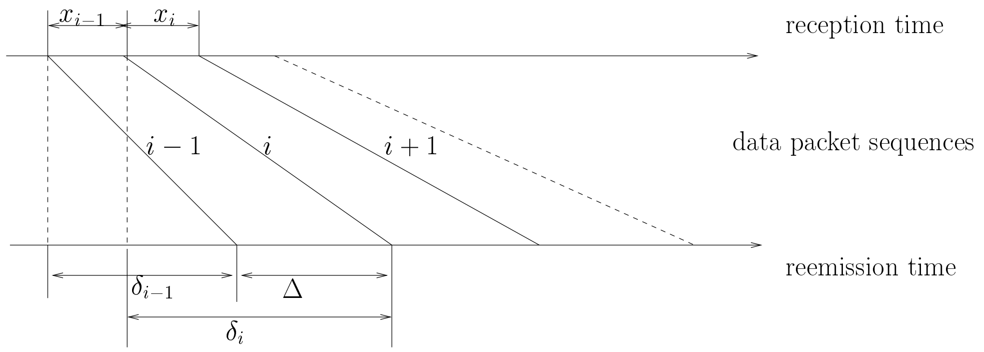


Figure 8.32: Setting the reemission timers for the ith data packet

8.5.3 Partitioning Dynamics

We argue (as will be seen in the simulation results) that our partitioning algorithm converges rapidly so that the initial partitioning is not disturbed by receivers changing their isolated rates. After the initial partitioning was performed, if any link experiences an RTT variation where $|\Delta\dot{\tau}| > b$ for a sufficiently long period[5], then a decision to move this receiver to the other subgroup could be taken.

Moving a link from a subset to another does not require more than updating the L and H contents. The main concern is when the link leads to the replicator. If this link is the unique direct link in H, then a merge decision of the H and L subgroups can be taken. In this case, the router re-enters the initial phase and would eventually perform a new partitioning in the future depending on the network dynamics. Otherwise, the replicator is moved to L and a new replicator would be designated according to our capacity metric (8.47). Once done, the router will set the $isRep$ field to 1 when it forwards the subsequent data packet on the new replicator link. On the reception of this data packet, the old replicator knows that it is no longer the replicator and continues the replication of the standing packets. The new replicator, upon the reception of this data packet, would start its replication process as have been described.

8.5.4 Simulation Results

Using the ns-2 implementation of our 2-subgroup AMCA protocol, a set of simulations have been conducted. The original congestion parameters α, β, and N are respectively set to 1, 3 and 32. The partitioning parameters a and b are respectively set to 0.05 and 0.2. The κ parameter is set to 0.5. For our simulations, we have adopted the topology of figure 8.33 where one source S is multicasting data packets to four receivers (with isolated rates $0.9Mbps$ for $R1$ and $R2$, $0.5Mbps$ for $R3$ and $R4$) through two active routers $A1$ and $A2$. The partitioning algorithm is enabled at the $A2$ router in order to increase the satisfaction of the four receivers. The simulation is run for 100 seconds of the real system.

Figure 8.34 shows the throughput achieved by the two subgroups $H = \{R1, R2\}$ and $L = \{R3, R4\}$ built by the active router $A2$. We can see that both the H and L receivers obtain their isolated rates of $0.9Mbps$ and $0.5Mbps$. Figure 8.35a shows the transmission rates of both the source and the chosen replicator (here $R1$). We can see that the source achieves rapidly a transmission rate of $900Kbps$, that the partitioning is performed in less than 2 seconds, and that the replicator achieves approximately a transmission rate of $500Kbps$ which corresponds to the isolated rate of the L receivers. Figure 8.35b shows the evolution of the data packets reception through their sequence numbers. At the beginning the two lines have the same slope. When the partitioning algorithm converges, one line keeps the same slope and the other achieves twice the original slope. Figure 8.36 shows the same experiment conducted with the same topology but with isolated rates of $1Mbps$ for the $R3$ and $R4$ and $2Mbps$ for the two others. Once again, the different receivers achieve a reception rate the closest to their isolated rates.

[5]such a decision could be taken on the reception of K subsequent CRs with $|\Delta\dot{\tau}| > b$. K is a parameter to be set appropriately to limit oscillations while avoiding performances degradation

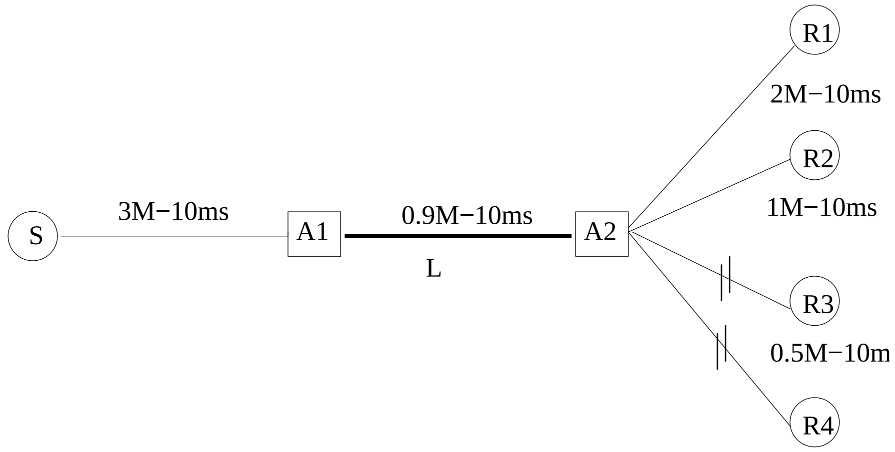


Figure 8.33: The used topology

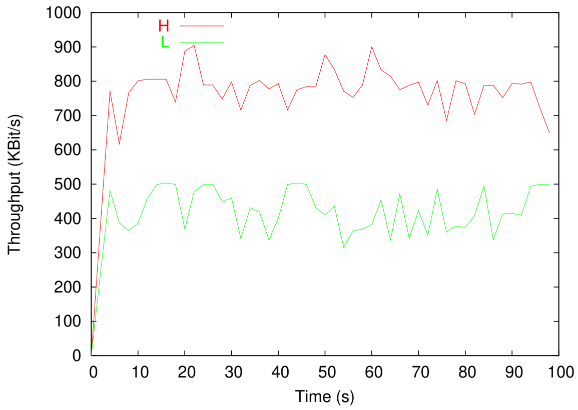
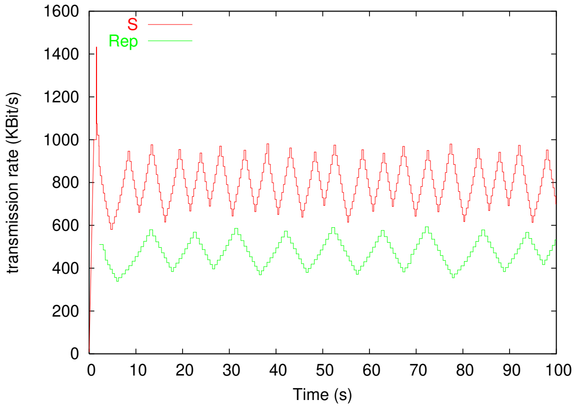


Figure 8.34: The 2 subgroups achieved throughput.

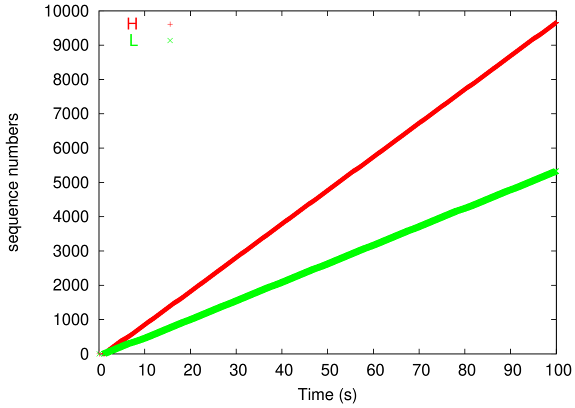
8.6 Conclusion

In order to accommodate heterogeneity in a multicast session, we proposed a new replication mechanism to implement a fine-grained multi-rate congestion control. Our approach consists in implying a set of receivers to replicate data they receive to other receivers with lower capacities. A replicator in the same way as a single-rate source, will adapt its rate depending on feedback it receives from the members of its associated subgroup. A partitioning algorithm is provided to split a set of receivers into subgroups of similar capacities. The main feature of this algorithm in addition to its simplicity, is that it does not rely on a prior knowledge of the receivers' capacities. The partitioning is performed on-the-fly as soon as feedback from the receivers are collected. The knowledge of the RTT variation experienced by every receiver is required but there is no assumption on how the RTT variations are measured, therefore a simple ping method could be suitable. Moreover, our partitioning algorithm guarantees a minimum utility value (depending on the ρ parameter), and with a minimum computation effort, it converges, or at least approximates the optimal solution.

To be more scalable and fairer with other sessions while improving the receivers satis-



(a)



(b)

Figure 8.35: (a) Transmission rate of the source and the replicator, (b) sequence numbers of data packets received by the H and the L receivers .

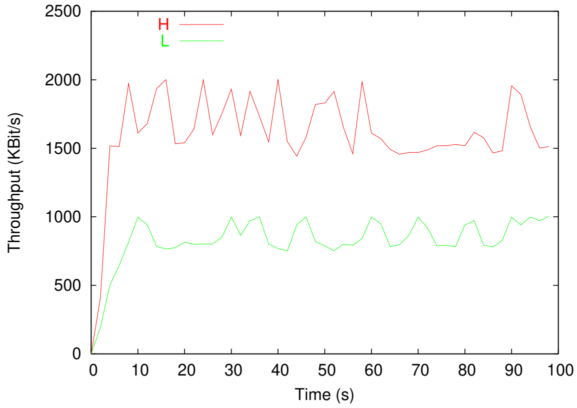


Figure 8.36: The 2 subgroups achieved throughput in the second experiment.

faction, we suggest to execute the partitioning algorithm at the routers. This additionally, allows for the construction of a regulation tree close to the multicast delivery one. Analysis and simulations showed that, our approach compared to single-rate and source-based replication schemes, is more scalable and allows for a better intra-session fairness. Compared to a layered approach, ours allows for a fine-grained congestion control without requiring a large number of subgroups. Using ns to validate our approach, an extension of a single-rate congestion avoidance algorithm (AMCA) has been implemented. Preliminary simulation results showed that the partitioning algorithm converges rapidly.

Chapter 9

Conclusion

The exponential growth of the Internet and its applications is becoming more and more demanding of bandwidth and scalability. A wide spectrum of the current and emerging Internet applications involve multiple parties and thus require the availability of a multi-point communication mechanism. Multicast offers new opportunities to these applications since it is an efficient way to disseminate data to a large number of recipients by avoiding multiple duplications. For many years, multicast has been a hot research topic where a multitude of problems are addressed at different levels. Despite the considerable effort and progress in solving multicast-related problems, there are still many issues to be dealt with.

In this thesis, we tackled some transport level problems where more attention is focused on reliable multicast which is essential to a multitude of applications. A primary concern is to achieve scalability to a large number of receivers while insuring reliability. We first, considered the problem of error recovery and provided a new reliable multicast protocol. Afterwards, we considered the problem of congestion, an inherent property of the best-effort Internet, and elaborated a new congestion avoidance algorithm for multicast communications. Finally, we dealt with the Internet intrinsic heterogeneity where multiple receivers with different capacities are likely to be subscribed to the same multicast group. For this last problem, we provided some solutions that are targeted to a fully reliable multicast. However, these solutions can be adopted to loss-tolerant applications. All the solutions we proposed were evaluated and validated through analysis, simulation or real implementation.

To achieve our goal, an in-network approach is adopted to provide solutions to the problems we faced. This is mainly due to the availability of network information which are necessary for developing more accurate solutions. There are many trade-offs between end-to-end and in-network approaches. Despite the potential of in-network services, their deployment is still subject to ISPs and carriers. We chose to adopt the active networking technology to validate our proposals. This choice is motivated by the flexibility provided by the active networks and the availability of active execution environments suitable to implement our solutions.

Throughout this thesis, we mainly showed how a set of receivers belonging to the same multicast session can cooperate in a distributed fashion to achieve better performances and more satisfaction among them. In fact, to achieve reliability, a receiver can operate as a retransmitter of lost data packets. To deal with heterogeneity, some receivers can operate as replicators for other ones with lower capacities. In this chapter, most of this work contributions are enumerated before giving a list of potential future works.

9.1 Contributions

First, generic versions of some active reliable multicast protocols (ARM [74] and AER [64]) that already exist in the literature are evaluated and compared using a throughput analysis. Through this study, the benefit of their adopted active services such as local recovery from the routers, NACK suppression and the subcast are evaluated. We showed for instance that these services allowed for improving the overall throughput achieved by a multicast session. Particularly, it has been shown that the subcast allows for a higher and constant throughput from the receivers perspective. We also considered the active routers density as well as their processing power. We mainly showed that increasing the active routers density allows for higher throughput even with slower processing power.

Afterwards, another service which consists in the early loss detection of losses by the routers, has been considered. An evaluation study based on a delay analysis is performed in order to quantify the benefit of such a service. We mainly showed that a loss detection-capable router location with respect to the source, is deciding to achieve any benefit. Such a router has to be neither too far nor too close to the source. A suited location could be in the source side just before the backbone. In this way, a maximum of the source link losses are detected without being so far from the source. When the loss detection-capable router location is well chosen, we showed that this new service allows for increasing the performances in terms of recovery delay. We also considered other factors such as routers' density and processing power.

Based on the previous analytical work, a new active reliable multicast protocol called DyRAM has been proposed. DyRAM is an attempt to satisfy both the congruence with the physical multicast tree as well as the adaptability requirements implied by the dynamic nature of a multicast group. DyRAM relies on some in-network services in order to build a logical recovery tree which is close to the multicast tree. This latter is used by DyRAM in order to perform feedback suppression/aggregation at the intermediate nodes. Some soft-state information are kept within the routers in order to implement the different functionalities of DyRAM. The active services have been conceived to be as light as possible in order to do not slow down the forwarding functionalities of the routers that belong to the multicast tree. A prototype of DyRAM has been implemented in java using the TAMANOIR [54] execution environment. Some experimental results on the processing cost at the routers have been presented.

Chapter 6 has been dedicated to the extensive simulation-based evaluation of DyRAM. The purpose was to evaluate the potential of the different active services of DyRAM through comparison studies with other reliable multicast protocols. This evaluation showed that DyRAM allows for less load on the different nodes in terms of consumed bandwidth. The DyRAM's different services allow for reducing the repair latency and implosion. Once again (and now by simulation), the benefit of increasing the density of the active routers has been highlighted. A more interesting feature of DyRAM is that its specific replier election service allows for a load balancing among the receivers. Finally, we showed that DyRAM has good scalability properties since the performances do not decrease when increasing the number of the receivers.

Aware that most of the experienced losses in the Internet are due to congestion situations, a congestion avoidance protocol has been proposed to operate in conjunction with DyRAM. This protocol called AMCA is based on RTT measurements between every two consecutive nodes in a multicast tree to probe for available bandwidth. Estimating the RTTs in a per section manner is scalable and avoids the rerouting problems. We first provided the rationale behind the solution we proposed before its description. Afterwards, simulations have been conducted to evaluate our congestion protocol. We mainly showed that it converges, makes use of the available bandwidth and reacts rapidly to bandwidth changes. We also proved the fairness of our approach towards TCP which is an interesting property for any congestion protocol. Finally it is worth mentioning that AMCA can operate with any reliable multicast or unicast protocol that benefit from the assistance of the routers.

Finally, heterogeneous receivers support is considered where the fastest ones' satisfaction has not to be affected. In chapter 8, we first, proposed a new partitioning algorithm of the receivers in subsets of similar capacities. This algorithm can be used by any multi-rate multicast protocol with layered or replicated approaches. Our proposed algorithm is based on RTT measurements sent by the subscribed receivers to perform an on-the-fly partitioning without any prior knowledge of the receivers' capacities. Despite its simplicity, our partitioning algorithm achieves or at least approximates the optimal solution. Furthermore, it guarantees a minimum utility value (i.e. receivers' satisfaction) depending on the algorithm parameters.

Afterwards, a new replication reliable multicast has been proposed where data replication is no longer the responsibility of the source. We called this new approach "receiver-based replication" in opposition to the traditional source-based replication schemes. The partitioning algorithm could be executed by the source or for more scalability by the routers. This latter approach allows for more fairness with other sessions while improving the multicast session receivers satisfaction. Analysis and simulation have been performed in order to compare our receiver-based replication scheme to a single-rate approach as well as a source-based replication scheme. We also studied the impact of our approach on the inter-session fairness and showed that it has better fairness properties compared to a source-based replication and a single-rate approach. Each replicator performs independently of other replicators, a fine-grained control of its transmission rate as a response to its subgroup feedback. In order to validate our receiver-based replication approach, we implemented a 2-subgroup AMCA extension to support more heterogeneous receivers. Simulation results have mainly shown the rapid convergence of our approach.

9.2 Perspectives

In this work, we were concerned by providing router-assisted solutions to some multicast transport level problems. Other problems such as security and deployment problems were beyond the scope of this thesis. However, they can be considered in the future. Beside general security requirements, we mainly have to address those related to our receiver-based recovery scheme where any member (in the current open model of IP multicast) could flood other receivers with requests or repairs. A main issue is how to avoid such attacks. One possible solution is to limit the set of receivers that can act as repliers and to be able to detect malicious members and pruning them. Another concern is then how to determine the set of receivers that can be trusted to perform retransmissions.

Deployment issues of both IP multicast and active networks in the Internet were not considered in this work. However, our proposed mechanisms still work even in a non fully deployed network. We argue that they can be incrementally deployed even if the performances will be degraded compared to the case of fully deployed services.

DyRAM, the reliable multicast protocol proposed has been extensively evaluated using simulation and analysis. A first implementation is available, however many of its mechanisms such as timeout setting, RTT estimations and congestion control still under development. As a future work, we plan to finish the implementation of DyRAM/AMCA framework so we would be able to test the whole in a real world application of grid computing.

AMCA has been evaluated using simulations on a very simple topology and scenarii. As a future work, AMCA has to be experimented on larger topologies that reflect the actual Internet. Moreover the congestion parameters that we used have to be more investigated to determine their optimal values and real influence on the protocol behavior in different circumstances. Moreover, AMCA can be used in unicast and thus can be subject of investigation in such case and compared to TCP, to mainly evaluate its behavior to rerouting paths problem.

The problem of heterogeneity when addressed, has brought many issues to be addressed in the future. First, larger scale simulations have to be conducted on our receiver-based replication scheme to mainly evaluate its dynamic behavior in the case of receivers changing their capacities over time. Indeed, a receiver may change its subgroup which could be split again or even merged with other subgroups depending on network dynamics. Some other analysis can be done in order for instance to evaluate the buffering means at the replicators side. Since the approach is only considered for the case of bulk data files which are of finite size, we argue that the buffering means required are limited and of course depend of the used algorithm to elect replicators and the transmitted file size. Finally, as a future work, the receiver-based replication scheme we proposed in the case of reliable data transmission can be adapted to multimedia applications such as video streaming.

Bibliography

[1] B. Adamson, C. Bormann, M. Handley, and J. Macker. Nack-oriented reliable multicast (norm) building blocks. Internet draft (work in progress) draft-ietf-rmt-bb-norm-06, IETF, June 2003.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. Nack-oriented reliable multicast protocol (norm). Internet draft (work in progress) draft-ietf-rmt-pi-norm-07., IETF, June 2003.

[3] M. D. Amorim, O. C. M. B. Duarte, and G. Pujolle. Improving user satisfaction in adaptive multicast video. *IEEE/KICS Journal on Communications and Networks*, 4(3), September 2002.

[4] Arturo Azcorra, María Calderón, Marifeli Sedano, and José I. Moreno. Multicast congestion control for active network services. *European Transactions in Telecommunications*, 10(3), May/June 1999.

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

[6] A. Ballardie. Core based trees (cbt) multicast routing architecture. RFC 2201, IETF, September 1997.

[7] L. Becchetti, C. Gaibisso, M. Draoli, and F. Lombardi. A java implementation of a reliable multicast file transfer protocol: Design and evaluation. In *MFTP component of JANE, IASTED 2001*, Honolulu, August 2001.

[8] D. B. Bertsekas and R. Gallagher. *Data Networks*. Prentice Hall, 1992. Second Edition.

[9] S. Bhattacharjee, K.L. Calvert, and E.W. Zegura. Active networking and the end-to-end argument. In *ICNP*, Atlanta, GA, October 1997.

[10] Supratik Bhattacharyya, James. F. Kurose, Don Towsley, and Ramesh Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. In *IEEE INFOCOM*, 1998.

[11] Supratnik Bhattacharyya, Don Towsley, and Jim Kurose. The loss path multiplicity problem in multicast congestion control. In *IEEE Infocom*, volume 2, pages 856–863, New York, USA, March 1999.

[12] J-C. Bolot and T. Turletti. Experience with control mechanisms for packet video in the internet. *ACM SIGCOMM Computer Communication Review*, 28(1):4–15, January 1998.

[13] Nicolas Bonmariage and Guy Leduc. Adaptation dynamique des debits des couches pour la transmission video multipoint. In *Actes du Colloque Francophone sur l'Ingenierie des Protocoles (CFIP'2002*, 2002.

[14] L. Brakmo and L. Peterson. End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995.

[15] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacherand A. Roetter, and W. Shaver. Flid-dl: Congestion control for layered multicast. In *NGC*, pages 71–81, November 2000.

[16] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM'98*, Vancouver, Canada, September 1998.

[17] J. W. Byers, M. Luby, and M. Mitzenmacher. Fine-grained layered multicast. In *IEEE INFOCOM*, pages 275–283, April 2001.

[18] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet group management protocol, version 3. RFC 3376, IETF, October 2002.

[19] B. Cain, T. Speakman, and D. Towsley. Generic router assist (gra) building block motivation and architecture. Internet draft (work in progress) draft-ietf-rmt-gra-arch-00.txt, IETF, October 1999.

[20] B. Cain and D. Towsley. Generic multicast transport services: Router support for multicast applications. In *NETWORKING*, pages 108–120, May 2000. Also available as CMPSCI technical report TR99-74.

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

[22] J.-M. Chang and N. F. Maxemchu. Reliable broadcast protocols. *ACM Transactions on Computer Systems*, 2:251–273, August 1984.

[23] Y. Chawathe, S. McCanne, and E. Brewer. Rmx : Reliable multicast in heterogeneous networks. In *IEEE INFOCOM*, March 2000.

[24] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestions avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[25] D. M. Chiu, M. Kadansky, and J. Provino. A congestion control algorithm for tree-based reliable multicast protocols. Technical Report TR-2001-97, Sun Microsystems Labs, 2001.

[26] D. M. Chiu, M. Kadansky, J. Provino, J. Wesley, and H. Zhu. Pruning algorithms for multicast flow control. In *NGC 2000*, November 2000.

[27] Dah Ming Chiu, Stephen Hurst, Miriam Kadansky, and Joseph Wesley. Tram: A tree-based reliable multicast protocol. Technical Report TR-98-66, SUN, July 1998.

[28] L. Costa, S. Fdida, and O. C. M. B. Duarte. Hop by hop multicast routing protocol. In *ACM SIGCOMM*, June 2001.

[29] Adam M. Costello and Steven McCanne. Search party: Using randomcast for reliable multicast with local recovery. In *Proc. of the IEEE INFOCOM, New york*, March 1999.

[30] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[31] S. Deering. Internet multicast routing: State of the art and open reseach issues. In *Multimedia Integrated Conferencing for Europe (MICE) Seminar*, the Swidish Instute of Computer Science, Stockholm, October 1993.

[32] S. E. Deering and D. R. Cheriton. Multicast routing in datagrams internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[33] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The pim architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.

[34] Steve. Deering. Distance vector multicast routing protocol. RFC 1112, IETF, August 1989.

[35] Steven Deering, Deborah Estrin, Dino Farinacci, and Van Jacobson. Protocol independent multicast version 2, dense mode specification. Internet draft (work in progress) draft-ietfidmr-pim-dm-spec-05.txt, IETF, May 1997.

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

[37] C. Diot and J. Crowcroft. Multicast transport survey. *Journal of Selected Areas in Communications*, 1997.

[38] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: A survey of protocols, functions, and mechanisms. *Journal on Selected Areas in Communications*, 15:277–290, April 1997.

[39] C. Diot, B. Levine, J. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network*, 2000.

[40] Anca Dracinshi and Serge Fdida. Congestion avoidance mechanism for unicast and multicast. In *European Conference on Universal Multiservice Networks, ECUMN'00*, October 2000.

[41] A. El-Sayed, V. Roca, and L. Mathy. A survey of proposals for an alternative group communication service. *IEEE Network magazine, Special issue on "Multicasting: An Enabling Technology*, January/February 2003.

[42] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (pim-sm): Protocol specification. RFC 2362, IETF, June 1998.

[43] S. Kumar et al. The masc/bgmp architecture for inter-domain multicast routing. In *SIGCOM*, 1998.

[44] X. Rex Xu et al. Resilient multicast support for continuous-media applications. In *NOSSDAV*, 1996.

[45] T. Faber. Experience with active congestion control. In *DARPA ACtive Networks Conference and Exposition, San Francisco, CA*, May 29-30 2002.

[46] K. Fall, K. varadhan, and S. floyd. Ns notes and documentation ucb/lbnl/vint. software and documentation available at http://www.isi.edi/nsnam/ns, July 1999.

[47] D. Farinacci, Y. Rekhter, P. Lothberg, H. Kilmer, and J. Hall. Multicast source discovery protocol (msdp). Internet draft (work in progress) draft-farinacci-msdp-*.txt, IETF, June 1998.

[48] W. Fenner. Internet group management protocol, version 2. RFC 2236, IETF, November 1997. absoleted by RFC3376.

[49] W. Fenner and S. Casner. A 'traceroute' facility for ip multicast. Technical Report draft-ietf-idmr-traceroute-ipm-*.txt, IETF, August 1998.

[50] S. Floyd and T. Henderson. The newreno modification to tcp's fast recovery algorithm, rfc 2582, 1999.

[51] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[52] 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.

[53] Rung-Hung Gau, Zygmunt J. Haas, and Bhaskar Krishnamachari. On multicast flow control for heterogeneous receivers. *IEEE/ACM Transactions on Networking*, 10(1):86–101, February 2002.

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

[55] S. Jamaloddin Golestani and Krishan K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *INFOCOM*, pages 990–1000, 1999.

[56] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, and M. Luby. The reliable multicast design space for bulk data transfer. RFC 2887, IETF, August 2000.

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

[58] Christian Huitema. The case for packet level fec. Fifth International Workshop on Protocols for High-Speed Networks, Sophia Antipolis, France, October 1996.

[59] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, 1988.

[60] V. Jacobson, S. McCanne, and M. Vetterl. Receiver-driven layered multicast. In *ACM SIGCOMM'96, Stanford, CA*, pages 117–130, August 1996.

[61] T. Jiang, M. Ammar, and E. Zegura. Inter-receiver fairness: A novel performance measure for multicast ABR sessions. In *Measurement and Modeling of Computer Systems*, pages 202–211, June 1998.

[62] T. Jiang, M. Ammar, and E. Zegura. On the use of destination set grouping to improve inter-receiver fairness for multicast abr sessions. In *IEEE INFOCOM'00*, March 2000.

[63] T. Jiang, E. Zegura, and M. Ammar. Inter-receiver fair multicast communication over the internet. In *the 9th International Workshopon Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 103–114, June 1999.

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

[65] S. K. Kasera, J. Kurose, and D. Towsley. A comparaison of server-based and receiver-based local recovery approaches for scalable reliable multicast. In IEEE, editor, *INFOCOM*, San Francisco, CA, March 1998.

[66] Sneha Kumar Kasera, Gísli Hjálmtýsson, and Donald F. Towsleyand James F. Kurose. Scalable reliable multicast using multiple multicast channels. *IEEE/ACM Transactions on Networking*, 8(3):294–310, 2000.

[67] Sneha Kumar Kasera, James F. Kurose, and Donald F. Towsley. Scalable reliable multicast using multiple multicast groups. In *Measurement and Modeling of Computer Systems*, 1997.

[68] R. Kermode and L. Vicisano. Author guidelines for reliable multicast transport (rmt) building blocks and protocol instantiation documents. RFC 3269, IETF, April 2002.

[69] Ibtissam El Khayat and Guy Leduc. A stable and flexible tcp-friendly congestion control protocol for layered multicast transmission. In *the 8th International Workshop IDMS*, 2001.

[70] T. Kim and M. H. Ammar. A comparison of layering and stream replication video multicast schemes. In *NOSSDAV'01*, pages 25–26, Port Jefferson, NY, June 2001.

[71] Peter S. Kruus. A survey of multicast security issues and architectures. In *the 21st National Information Systems Security Conference*, Arlington, VA, October 1998.

[72] Gu-In Kwon and John Byers. Smooth multirate multicast congestion control. In *IEEE INFOCOM'03, San Francisco, USA*, April 2003.

[73] A. Legout and E. W. Biersack. Plm: Fast convergence for cumulative layered multicast transmission schemes. In *ACM SIGMETRICS*, Santa Clara, California, USA, 2000.

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

[75] B. N. Levine and J. J. Garcia-Luna-Acceves. A comparaison of known classes of reliable multicast protocols. In *the international Conference and network Protocols (ICNP)*, October 1996.

[76] B.N. Levine, S. Paul, and J.J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically according to packet-loss correlation. In *Sixth ACM International Multimedia Conference (ACM Multimedia 98), Bristol, UK*, September 1998.

[77] Brian Neil Levine and J.J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In *IEEE International Conference on Network Protocols (ICNP) '97*, pages 241–250, Atlanta, Georgia, October 1997.

[78] B. Li and J. Liu. Multirate video multicast over the internet: an overview. *IEEE Network, Multicasting: an enabling technology*, 17(1), January 2003.

[79] D. Li and D. R. Cheriton. Oters (on-tree effecient recovery using subcasting): A reliable multi- cast protocol. In *6th IEEE International Conference on Network Protocols (ICNP'98)*, pages 237–245, Austin, Texas, October 1998.

[80] X. Li, S. Paul, P. Pancha, and M.H. Ammar. Layered video multicast with retransmission (lvrm): Evaluation of hierarchical rate control. In *INFOCOM'97*, 1997.

[81] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. Asynchrounous layered coding (alc) protocol instantiation. RFC 3450, IETF, December 2002.

[82] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. Layered coding transport (lct) building block. RFC 3451, IETF, December 2002.

[83] M. Luby, V. K Goyal, S. Skaria, and G. B. Horn. Wave and equation based rate control using multicast round trip time. In *ACM SIGCOMM*, January 2002.

[84] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Steman. Practical loss-resilient codes. In *International Conference on Multimedia and Expo*, volume II, pages 444–447, New York, NY, USA, August 2000.

[85] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. Forward error correction (fec) building block. RFC 3452, IETF, December 2002.

[86] M. Luby, L. Vicisano, and A. Haken. Reliable multicast transport building block: Layered congestion control. Internet draft (work in progress) draft-ietf-rmt-bb-lcc-00.txt, IETF, November 2000.

[87] Moufida Maimour and Cong-Duc Pham. A throughput analysis of reliable multicast protocols in an active networking environment. In *Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001)*, pages 151–158, Hammamet, Tunisia, July 2001.

[88] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. Ietf criteria for evaluating reliable multicast transport and application protocols. RFC 2357, IETF, June 1998.

[89] M. Mathis, J, Semke, J. Madhavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communication Review*, 27(3):62–82, 199.

[90] A. J. McAuley. Reliable broadband communication using a burst erasure correcting code. In *ACM SIGCOMM'90*, pages 297–306, Philadelphia, PA, Septembre 1990.

[91] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean C. Walrand". Analysis and comparison of TCP reno and vegas. In *INFOCOM*, pages 1556–1563, 1999.

[92] T. Montgomery. A loss tolerant rate controller for reliable multicast. Technical Report NASA-IVV-97-011, NASA/WVU, August 1997.

[93] J. Moy. Multicast extensions to ospf, rfc 1584, May 1994.

[94] John Moy. Multicast extensions to ospf. RFC 1584, IETF, March 1994.

[95] John Moy. Ospf version 2. RFC 1580, IETF, March 1994.

[96] J. Nonnenmacher, E. Biersac, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *Transactions on Networking*, 6(4):349–361, August 1998.

[97] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-based loss recovery of reliable multicast transmission. In *IEEE ICCC*, November 1996.

[98] Katia Obraczka. Multicast transport protocols: A survey and taxonomy. *IEEE Communications Magazine*, pages 94–102, January 1998.

[99] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: a simple model and its empirical validation. In *SIGCOMM*, 1998.

[100] T. Paila, M. Luby, R. Lehtonen, and V. Roca. Flute - file delivery over unidirectional transport. Internet draft (work in progress) draft-ietf-rmt-flute-01.txt, IETF, August 2003.

[101] 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.

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

[103] S. Pingali, D. Towsley, and J.F. Kurose. A comparaison of sender-initiated and receiver-initiated reliable multicast protocols. In ACM, editor, *SIGMETRICS*, volume 14, pages 221–230, 1994.

[104] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ipv6 and to tcp, October 1998.

[105] R. Rejaie, M. Handley, and D. Estrin. Layered quality adaptation for internet video streaming. *IEEE Journal on Selected Areas in Communications*, 18, December 2000.

[106] Injong Rhee, Nallathambi Ballaguru, and George N. Rouskas. Mtcp : Scalable tcp-like congestion control for reliable multicast. Technical Report TR-98-01, 1998.

[107] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 27(2):14–36, April 1997.

[108] L. Rizzo. pgmcc: a tcp-friendly single-rate multicast. In *SIGCOMM*, pages 17–28, 2000.

[109] L. Rizzo and L. Vicisano. A reliable multicast data distribution protocol based on software fec techniques (rmdp). In *HPCS'97 Workshop*, Chalkidiki, Grece, June 1997.

[110] K. Robertson, K. Miller, M. White, and A. Tweedly. Starburst multicast file transfer protocol (mftp) specification. Internet draft (work in progress) draft-miller-mftp-spec-03.txt, IETF, April 1998.

[111] V. Roca, L. Costa, R. Vida, A. Dracinschi, and S. Fdida. A survey of multicast technologies. Technical Report RPLIP6-2000-20, Universit6 Pierre et Marie Curie LIP6, Paris, September 2000.

[112] V. Roca and B. Mordelet. Design of a multicast file transfer tool on top of alc. In *7th IEEE Symposium on Computers and Communications (ISCC02)*, Taormina, Italy, July 2002.

[113] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving reliable multicast using active parity encoding services (apes). In *IEEE INFOCOM'99*, New York, April 1999.

[114] Dan Rubenstein, Jim Kurose, and Don Towsley. A study of proactive hybrid fec/arq and scalable feedback techniques for reliable rea-time multicast. *Computer Communications Journal*, 24(5–6):563–574, March 2001.

[115] Dan Rubenstein, Jim Kurose, and Don Towsley. The impact of multicast layering on network fairness. *IEEE/ACM Transactions on Networking*, 10(2), April 2002.

[116] J. F. De Rzend, A. Mauthe, S. Fdida, and D. Hutchison. Fully reliable multicast in heterogeneous environment. In IFIP/IEEE, editor, *Protocols for High Speed Network (PfHSN'96), Sophia-Antipolis, France*, 1996.

[117] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, pages 277–288, 1984.

[118] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedlyand N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. Pgm reliable transport protocol specification. RFC 3208, IETF, December 2001. Appeared also in IEEE Networks Magazine, Special issue on "Multicasting: An Enabling Technology, 2003.

[119] T. Strayer. Xtp home page, http://www.ca.sandia.gov/xtp/xtp.html.

[120] 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.

[121] D. Thaler, M. Handley, and D. Estrin. The internet multicast address allocation architecture. RFC 2908, IETF, September 2000.

[122] H. Tzeng and K. Siu. On max-min fairness congestion control for multicast abr service in atm. *JSAC*, 15(3), April 1997.

[123] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Conference on Computer Communications (IEEE Infocom'98), San Fransisco, USA*, pages 1–8, 1998.

[124] B. Vickers, C. Alburquerque, and T. Suda. Source-adaptive multilayered multicast algorithms for real-time video distribution. *IEEE/ACM Transactions on Networking*, December 2000.

[125] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol. RFC 1075, IETF, November 1988.

[126] Huayan Amy Wang and Mischa Schwartz. Achieving bounded fairness for multicast and TCP traffic in the internet. In *SIGCOMM*, pages 81–92, 1998.

[127] 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.

[128] B. Whetten and J. Conlan. A rate based congestion control scheme for reliable multicast. Technical report, Reliable Multicast Research Groupworking document,, December 1998.

[129] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby. Reliable multicast transport building blocks for one-to-manybulk-data transfer. RFC 3048, IETF, January 2001.

[130] Brian Whetten, Dah Ming Chiu, Miriam Kadansky, Seok Joo Koh, and Gursel Taskale. Reliable multicast transport building block for track. Internet draft (work in progress) draft-ietf-rmt-bb-track-02.txt, IETF, November 2002.

[131] Jörg Widmer and Mark Handley. Extending equation-based congestion control to multicast applications. In *ACM SIGCOMM (San Diego,CA)*, August 2001.

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

[133] Lidia Yamamoto and Guy Leduc. An active layered multicast adaptation protocol. In *IWAN*, pages 179–194, 2000.

[134] 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.

[135] Y. Yang, M. Kim, and S. Lam. Optimal partitioning of multicast receivers. In *Proceedings of the 8th IEEE International Conference on Network Protocols (ICNP)*, Osaka, Japan, November 2000.

[136] Y. R. Yang and S. S. Lam. Internet multicast congestion control: A survey. In *ICT 2000, Acapulco, Mexico*, May 2000.

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

[138] Wonyong Yoon, Dongman Lee, Hee Yong Youn, Seungik Lee, and Seok Joo Koh. A combined group/tree approach for many-to-many reliable multicast. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, New York, NY. USA, June 2002.

Publications

Journals

M. Maimour and C. Pham. An Active Reliable Multicast framework for Data Distribution, to appear in the Cluster Computing Journal (selected paper from the AMS'02 workshop).

M. Maimour and C. Pham. Modélisation et évaluation des protocoles de multicast fiable dans le contexte des réseaux actifs. Calculateurs Parallèles, Réseaux et Systèmes Répartis (CPRSR), numéro spécial sur l'évaluation quantitative des réseaux, 13(6), pp 575-599, 2001.

International Conferences

M. Maimour and C. Pham. Dealing with Heterogeneity in a Fully Reliable Multicast Protocol. In the proceedings of the 11th IEEE International Conference on Networks (ICON'03), September 28th - October 1st, 2003 - Sydney, Australia

M. Maimour and C. Pham. A RTT-based Partitioning Algorithm for a Multi-rate Reliable Multicast Protocol. In proceedings of the 6th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC'03), July 23-25, 2003 - Estoril, Portugal.

M. Maimour and C. Pham. AMCA, an Active-based Multicast Congestion Avoidance Algorithm. In Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC'03), June 30 - July 3, 2003, Kemer-Antalya, Turkey. **Best Paper Award**.

M. Maimour and C. Pham. A New Active-based Multicast Congestion Avoidance Algorithm. In Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE'03), May 4-7, 2003, Montreal, Canada.

M. Maimour, C. Pham. Towards an Application-aware Multicast Communication Framework for Computational Grids. In Proceedings of the 7th Asian Computing Science Conference (ASIAN'02), December 4-6, 2002, Hanoi, Vietnam.

M. Maimour, C. Pham. An Analysis of a Router-based Loss Detection Service for Active Reliable Multicast Protocols. In Proceedings of the IEEE International Conference On Networks (ICON'02), August 27-30 2002, Singapore, pp 49-56.

M. Maimour, J. Mazuy, C. Pham (2002) The Cost of Active Services in Active Reliable Multicast. In Proceedings of IEEE 4th International Workshop on Active Middleware Services (AMS'02), July 24-26, 2002, Edinburgh, UK, pp 67-72.

M. Maimour, C. Pham. A Loss Detection Service for Active Reliable Multicast Protocols. In Proceedings of the International Network Conference (INC'02), July 16-18 2002, Plymouth, UK.

M. Maimour, C. Pham. Dynamic Replier Active Reliable Multicast (DyRAM). In Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC'02), July 1-4 2002, Taormina, Italy, pp 275-282.

M. Maimour, C. Pham. An Active Reliable Multicast Framework for the Grids. In Proceedings of the International Conference on Computational Science (ICCS'02), April 21-24 2002, Amsterdam, The Netherlands, pp 588-597.

L. Lefevre, C. Pham, P. Primet, B. Tourancheau, B. Gaidioz, J.P. Gelas, M. Maimour. Active Networking Support for the Grid. In Proceedings of the third International Working Conference on Active Networks (IWAN'01). September 30, October 1-2, 2001, pp 16-33.

M. Maimour and C. Pham. A Throughput Analysis of Reliable Multicast Protocols in an Active Networking Environment. In Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC'01), July 3-5 2001, Hammamet, Tunisia, pp 151-158.

M. Maimour and C. Pham. Active Reliable Multicast for Efficient Data Distribution on an Internet-based Grid Computing Infrastructure. In Proceedings of the International Conference on Internet Computing (IC'01), June 25-28 2001, Monte Carlo Hotel, Las Vegas, USA, pp 437-443 .

Research Reports

M. Maimour and C. Pham. Dealing with Heterogeneity in an Active-based Multicast Congestion Avoidance Protocol. INRIA research report RR-4796, March 2003. Also available as a LIP/ENS research report under RR2003-20.

M. Maimour and C. Pham. A RTT-based Partitioning Algorithm for a Multi-rate Reliable Multicast Protocol. INRIA research report RR-4779, March 2003. Also available as a LIP/ENS research report under RR2003-16.

M. Maimour and C. Pham. AMCA : An Active-Based Multicast Congestion Avoidance Algorithm. INRIA research report RR-4689, January 2003. Also available as a LIP research report under RR2003-07.

M. Maimour and C. Pham. An Analysis of a Router-based Loss Detection Service for Active Reliable Multicast protocols. TR03-2001, November 2001. Also available as an INRIA research report RR-4636, November 2002 and as a LIP research report under RR2003-06.

M. Maimour and C. Pham. Dynamic Replier Active Reliable Multicast (DyRAM). INRIA research report RR-4635, November 2002 and as a LIP research report under RR2003-05.

M. Maimour and C. Pham. A Throughput Analysis of Reliable Multicast Protocols in an Active Networking Environment. TR01-2001. January 2001.