RELIABLE MULTICAST TRANSPORT TO LARGE GROUPS

Thèse N^{o} 1832

PRÉSENTÉE À LA SECTION DE SYSTÈME DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR
JÖRG NONNENMACHER
Titulaire du Diplôme d'Informatique (Dipl.-Inf.) de l'Université Fridericiana de Karlsruhe, Allemagne.
de nationalité Allemande

Composition du Jury :

Prof. Alain Wegmann, EPFL, Switzerland,	président de commission
Prof. Ernst W. Biersack, Eurécom, France,	directeur de thèse
Prof. Pierre Humblet, consultant, USA,	corapporteur
Prof. Jim Kurose, UMASS, USA,	corapporteur
Prof. Jean-Yves Le Boudec, EPFL, Switzerland,	corapporteur
Prof. Keith Ross, Eurécom, France,	corapporteur

Sophia Antipolis, Eurécom, July 1998

Reliable Multicast Transport to Large Groups

Jörg Nonnenmacher

Submitted to the Section of Communication Systems.
Swiss Federal Institute of Technology (EPFL),
for the degree of docteur ès sciences.

Abstract

The thesis investigates the reliable transport from one sender to a group of receivers (multicast). It is often stated that network support is required to allow for efficient reliable multicast. This thesis shows that this is not the case. Efficient reliable multicast is possible on a pure end-to-end basis even for a very large number of receivers.

In the case of data loss in the network, reliability is efficiently provided, if receivers signal the loss to the sender and the sender in turn performs a retransmission of the missing data. The complexity of such a system increases with the number of receivers. Both, the number of retransmissions to be performed by the sender and the amount of feedback returned to the sender increases with the number of receivers.

The thesis makes three major contributions to the field of reliable multicast:

For the first time the scalability of protocol mechanisms is analyzed for groups of size 1 up to 1 million receivers.

For loss recovery a single parity packet can repair different losses at different receivers. Therefore, loss recovery by retransmission of parities outperforms retransmitting originals. The comparison of loss recovery by parity and loss recovery by originals is investigated for burst loss, for correlated loss among receivers and for a limited processing capability at sender and receivers.

Feedback from very large groups is dangerous, since the sender may get overwhelmed by feedback messages. The thesis contains an end to end solution for feedback from an unknown number of receivers. The feedback method allows to control the average number of feedback messages and to estimate the number of receivers. It is shown that this feedback method results in faster feedback than provided by other end-to-end solutions.

Finally, the proposed multicast protocol mechanisms are compared to protocol mechanisms that have support from the network.

Reliable Multicast Transport to Large Groups

Jörg Nonnenmacher

Thèse présentée à la Section de Système de Communication.
École Polytechnique Fédérale de Lausanne (EPFL),
pour le titre de docteur ès sciences.

Résumé

Cette th`ese ´etudie le transport fiable d'un ´emetteur vers un groupe de r´ecepteurs (multicast). On affirme souvent que l'on a besoin des services des r´eseaux pour permettre un transport multicast fiable efficace. Cette th`ese montre que ce n'est pas le cas. Le transport multicast fiable efficace est possible de bout-en-bout uniquement même pour un tr`es grand nombre de r´ecepteurs.

Dans le cas de pertes de donn´ees dans le r´eseau, la fiabilit´e est efficacement pr´eserv´ee, si les r´ecepteurs signalent la perte `a l'´emetteur, et que l'´emetteur ex´ecute une retransmission des donn´ees manquantes. La complexit´e d'un tel syst`eme augmente avec le nombre de r´ecepteurs. Le nombre de retransmissions `a ex´ecuter par l'´emetteur et la quantit´e de feedback envoy´ee `a ce dernier augmentent avec le nombre de r´ecepteurs.

Cette th`ese apporte trois contributions principales au domaine du multicast fiable:

Pour la premi`ere fois, la robustesse en fonction du nombre de r´ecepteurs (scalability) est analys´ee pour des groupes allant de 1 jusqu'`a 1 million de r´ecepteurs.

Le domaine de la r´ecup´eration de pertes (loss recovery) a ´egalement ´et´e ´etudi´e. Un seul paquet de parit´e peut r´ecup´erer diff´erentes pertes se produisant pour diff´erents r´ecepteurs. Par cons´equent, la r´ecup´eration de pertes par la retransmission de paquets de parit´es est meilleure que par la retransmission des originaux. La comparaison de la r´ecup´eration de pertes par retransmission de paquets de parit´e et par retransmission de paquets originaux est ´etudi´ee dans les cas suivants : pertes en rafale (burst losses), pertes corr´el´ees parmi les r´ecepteurs et capacit´e de traitement limit´ee `a l'´emetteur et aux r´ecepteurs.

Le feedback dans des groupes tr`es grands est dangereux, puisque l'´emetteur est submerg´e par des messages de feedback. Cette th`ese explore une m´ethode de bout-en-bout in´edite pour r´esoudre ce probl`eme de feedback pour un nombre inconnu de r´ecepteurs. Cette m´ethode permet de contrôler le nombre moyen de messages de feedback et d'estimer le nombre de r´ecepteurs. On a montr´e que cette m´ethode est plus rapide que toutes les autres m´ethodes de feedback de bout-en-bout actuelles.

En conclusion, les m´ecanismes multicast propos´es sont compar´es aux m´ecanismes multicast s'appuyant sur les services r´eseaux.

Acknowledgments

First of all I would like to thank my advisor Prof. E. Biersack, who gave me a free hand with my research, and guided me with his strong intuition several times into the right subject. Around spring 1996 I was evaluating different multicast trees for reliable multicast and he asked me: "Can't you do the same for FEC ?". After the work on FEC it was also him who had the feeling that for feedback there is still a lot of potential. He was right again. I am also indebted to him and to Jacques Labetoulle, the department head, for giving me the possibility to present my work at conferences, to meet other researchers and to establish valuable contacts. A possibility that is not encountered in every department.

Chapter 3 presents joint work with Don Towsley from UMASS and Ernst Biersack. I gratefully acknowledge their guidance and permission to use this work in my thesis. I want to thank Don Towsley for organizing my visit at AT&T. From him I learned that spending more time on analytical work than starting simulation straight away provides much deeper insight and can lead faster to more results. Both, him and Jim Kurose I want to thank for enabling my stay at UMASS.

I am indebted to Jim Kurose for his detailed comments on every aspect of this thesis.

During the three years of the thesis, several other researchers located in Sophia Antipolis influenced this work in different ways. A great thanks for the fruitful discussion and the time spent. The experience of Jean C. Bolot, INRIA, allowed the choice of a good burst loss model. The analytical skills of Alain-Jean Marie, INRIA, opened my eyes for extreme value theory. The graph theoretic insight of Jean-Claude Bermond, helped me to find the way through the forest towards the right trees. At EURECOM Raymond Knopp, Dirk Slock, and Karim Maouche were always a great help in mathematical problems.

The great environment I encountered at EURECOM allowed me to do my best. Special thanks go to Agnes Rougier from the library who always was an indispensable help. Thanks also to Claire for helping me through the jungle of French administration and to Remi for helping me through the jungle of Swiss administration. Thanks to Agnes Buchwalter and Nathalie Richardin for organizing the trips.

Further, I would like to thank the people that accompanied me through this part of my life and that were of big value to me. Magali Lapchin, the reason that made me consider a thesis in France. Karim Maouche my warm-hearted friend and room-mate with an ever-open ear.

Finally, I want to thank my parents for their strong believe in me, for supporting my studies in Karlsruhe and for helping me with settling down in France.

The thesis is dedicated to my grand-father Heinrich Nonnenmacher.

He transfered a lot of values to me in my early years that I still believe in. Maybe most closely related to the thesis are his words: "If you do something, better take more time and do it right."

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In front of a huge aircraft it takes a long time for a flight attendant to explain to a single passenger why boarding is not possible. Facing the loud noise, the interested passenger will ask whenever a part of the explanation is lost. The friendly flight attendant repeats over and over again, until the passenger finally understands the explanation.

For a group of passengers in the same situation the flight attendant's task becomes difficult. The flight attendant can not listen to all questions at the same time and can only give one answer at a time. Different passengers ask different questions and the flight attendant takes much more time in repeating different parts of the explanation than for a single passenger. The task for the flight attendant becomes more difficult as the size of the group increases.

One concern of this thesis is to reduce the impact of the group size in such a communication. However, we will concentrate on computer communication, and especially on communication via a channel where a *single send* operation will communicate information from a sender to a group of receivers. Such a channel is referred to as a **multicast channel**. The single send operation is referred to as a **multicast**. If the group consists of only one receiver we refer to **unicast**, if the group consists of all possible receivers we refer to **broadcast**.

We will focus on the case where the sender as a source of information is connected to the group of receivers via a lossy multicast channel over an interconnected network. **Reliable Multicast** requires the eventual transfer of all information from the sender over the lossy channel to all the receivers.

1.1 Two Simple Protocols

In order to point out the challenges of reliable multicast, we will use two simple examples of unicast communication and extend them to multicast communication.

Consider a unicast from a sender to a receiver, where messages sent via the channel are subject to loss. Then, the following two simple protocols assure a reliable unicast from the sender to the receiver in the sense that a message sent by the sender is received by the receiver:

The first protocol

- The sender sends the same message over and over again with a certain frequency.

- The receiver listens to the channel and waits for the message.

The second protocol

- The sender sends the same message over and over again with a certain frequency and stops sending when it receives a confirmation of successful reception (ACK).
- The receiver listens to the channel and waits for the message. On receipt of the message it sends an ACK back to the sender.

Both protocols achieve reliable communication after a certain time. The major difference between the two protocols is that the sender in the second protocol is aware of the successful reception and can stop sending. The awareness of the sender requires a feedback channel. It further requires the sender to have the ability to receive and it requires the receiver to be able to send.

Consider now the straightforward extension of the two protocols for a multicast to a group of $R, R > 1$, receivers. The first protocol need not be changed for the case of R receivers. Reliable multicast is provided: after a certain time *all* receivers will have received the message. Again the sender is not aware of the reception status of the receivers.

The second protocol must be changed, since the sender must be able to distinguish the ACKs from different receivers. When the sender has received an ACK from every receiver, it stops sending. The main difference from unicast is that the sender must receive the ACK of *every* receiver. In this protocol the sender's burden increases with every receiver: one ACK per receiver needs to be processed and the identity of every receiver needs to be known. Further, ACKs may be returned at the same time and overwhelm the sender. Therefore, an additional mechanism is required.

From these simple examples we observe that reliability without awareness by the sender about the reception status is achieved easily - whatever the number of receivers will be. We further observe that if awareness at the sender is needed, problems arise with the number of receivers.

Mechanisms that keep the complexity of a communication with R receivers close to constant with R are referred to as **scalable**.

Our major goal is to achieve *scalable* reliable multicast from one sender to multiple receivers with awareness of the sender about the reception status of the receivers.

1.2 Technological Context

The foundation for multicast transmission in the Internet was pioneered by the work of Steve Deering [1, 2]. Based on his work the MBONE [3, 4] was developed. The MBONE is an overlay network on top of the Internet, capable of multicast by copying data in the network towards different receivers. We will adopt the IP multicast notion of a multicast group. A multicast group consists of a set of members. Members can join and leave a group. At the network layer a multicast group is identified by a network address. A network capable of multicast ensures that a message sent to a multicast address is routed and forwarded towards all group members via the multicast channel. For the case of a lossy multicast channel, the message is not guaranteed to be received at all group members.

For instance, in the Internet packets are sent via a best effort service and may be discarded in the network due to buffer overflows in network nodes. As a consequence, only a subset of the group members may receive the message. The size of the subset of successful receptions can range from one, to some, to all group members.

Our aim is to investigate mechanisms for the reliable delivery in the sense that a message sent by a sender to a group of receivers is received at *all* receivers after a certain time.

The deployment of multicast in the Internet and the emerging number of satellite systems, enables a range of multicast applications. In general, multicast *data* applications can be classified into one of the following four categories:

Bulk data transfers

One sender sends a large amount of data to a potentially very large number of receivers. Bulk data transfer applications have no stringent time-constraint, but data needs to be transmitted reliably to all receivers. A software distribution, or the replication of a database falls in this category.

Data Sharing

A collaborative group, where single members initiate a data share with the other members of the group, is an example of data sharing. Data sharing applications typically have a small number of senders in the collaborative group, but the number of passive group members, the receivers, is potentially high. There is no strong requirement for timely delivery, but there is a strong requirement for reliable transport. An example for such an application is a shared whiteboard, where things written, or drawn, by single members of the group are seen by all other members.

Data Feeds

In the case of data feeds small items of data are frequently sent from one sender to a group of receivers. An example for a data feed is the distribution of stock quotes. Such an application has high requirements for the timely and reliable delivery of each item of data. The potential number of receivers of such a stream can become very large.

Interactive Gaming

Here, every member of a group of players frequently sends small amounts of data to all other members. Interactive gaming has very high requirements for the timely delivery and to the reliable transfer. Typically the multicast groups involved in an interactive game are rather small, although this may change in the future.

The applications in these four classes require that data be sent *reliably to all receivers*. Real-time applications, such as multimedia conferencing have tight real-time constraints that restrict the reliability that can be achieved for a lossy channel with fixed capacity. Therefore, real-time applications are built to tolerate a certain loss rate. We will focus on applications that require full reliability in the sense that all data sent by the sender must be received at all the receivers.

1.3 Issues for Reliable Multicast

1.3.1 Requirements

The two example protocols from section 1.1 showed that there is no single reliable multicast protocol for all applications. Already, a different number of receivers changes the requirements for reliable multicast. The requirements for a reliable multicast protocol are determined by the specific type of multicast application. Different multicast applications can be distinguished by:

- **The number of senders**: Applications like a shared whiteboard or a multi-player game include several senders that all send to the same group. On the other hand a software update from a software vendor, or a stock market data distribution have only one sender that sends to the group of receivers. For multiple senders sending to the same multicast channel a certain access policy must be determined. Anyhow, data can just have one origin, one source, and we will investigate reliable multicast protocols that transmit the data from the source to multiple receivers.

- **The number of receivers**: The number of receivers differs significantly from application to application. The update of a company's replicated sites is a reliable multicast application with a small number of receivers. The distribution of a popular Web browser is an application that needs reliable multicast to a high number of receivers. The requirements for reliable multicast for both applications are not the same: a high number of receivers imposes additional challenges. The aggregate loss of all receivers seen by the sender is higher for more receivers, since different receivers will lose different data. Therefore a high number of receivers imposes a challenge for efficient loss recovery for the group of receivers. Equivalently, the amount of feedback from the receivers increases with the number of receivers and requires means to handle the large amount of feedback from the receivers.

- **The amount of data to transmit**: A news ticker transmits a small amount of data from time to time, while the amount of data for a software distribution is usually several Megabytes. Small amounts of data restrict the number of techniques that can be employed in reliable multicast protocols: block spreading, cumulative feedback and packet level FEC are techniques that require a large amount of source data in order to realize efficiency gains.

- **Time-constraints**: Stock market distribution requires the timely delivery of data in the order of seconds, while software distribution does not have such a stringent time-constraint. A time-constraint is the time between the point in time of the sending of data by the sender and the point in time the data needs to be received at latest by the receiver. Mechanisms that recover from loss require a certain time. Therefore, the presence of a time-constraint restricts the probability of a successful reception. In the presence of a time-constraint and a fixed capacity channel the goal for reliable multicast can be formulated as to deliver most data to most receivers.

- **Security constraints**: Security considerations, such as the authenticity of the sender, impose restrictions on reliable multicast protocols. A natural approach to ensure reliability for a group of receivers is to have a hierarchical setup of retransmitters that care of reliable transfer to a receiver subgroup. Such retransmitters may not be trusted sources of information and the design space for reliable multicast protocols is restricted.

- **Ease of deployment**: New technology achieves faster acceptance when the technology is easily deployed. Reliable multicast protocols should be easily deployable. When network support is needed to allow for reliable multicast the deployment of reliable multicast is slowed down.
- **Sender Awareness**: A sender that needs to be aware of the successful reception at every receiver is more complex, since it needs to maintain the identity of every receiver and needs to receive a positive acknowledgment (ACK) about the successful reception from every receiver. When awareness at the sender about the successful reception at the receivers is not required a reliable multicast protocol based solely on negative acknowledgments (NAK) is sufficient.

1.3.2 New Challenges

Several new challenges are encountered for reliable multicast transmission that do not exist in unicast communication:

- **Scalability**: Scalability is identified as a key requirement for multicast.

  In general, a system is scalable when the system keeps working, as it grows in size. Another possible definition of scalability is to define scalability as the independence of the system performance of the system size.

  In the case of a multicast the system size is given by the number of receivers. Different numbers of receivers should not lead to a major difference in the throughput of the protocol. Scalability with the number of receivers is therefore a requirement for reliable multicast protocols.

- **Heterogeneity of receivers**: Different receivers have different properties, such as different processing rates, different memory resources, and different network connections, and finally a different path through the network originating at the sender. With the different paths through the network, different loss rates and different delays are encountered at different receivers. Reliable multicast must therefore deal with a variety of heterogeneous receiver settings.

- **Feedback implosion**: Feedback implosion describes the sending of feedback by several receivers at the same time, leading to a burst of feedback messages at the sender. This occurs for receiver-based loss detection, where data is lost at several receivers, when all receivers want to send a NAK back to the sender. Reliable multicast protocols have to develop a mechanism to avoid such synchronous feedback sending.

- **Congestion control**: A separate issue, but closely related to reliability is congestion control for multicast. A general formulation of the problem can be found in [5] were congestion is defined as the loss of the network utility to a specific user due to an increase of traffic in the network. Congestion control is the mean to maximize the network utility to a user and to be fair to other network users. Congestion control is a difficult problem already for unicast communication and is still under investigation. Even more dimensions are added to the problem by the consideration of multicast. As with reliable multicast, challenges for multicast congestion control include the scalability of the control with the number of receivers, the challenges due to the heterogeneity of paths through the network for different receivers, and the problem of providing timely feedback while avoiding feedback implosion.

1.4 Approaches to Reliable Multicast

The current body of literature for reliable multicast is very large. In the following only a short overview and discussion of the most relevant work to date will be given. A more comprehensive review of the literature can be found in each of the following chapters about modeling, loss recovery, feedback, and the comparison of different reliable multicast approaches. We will outline the two major classes of approaches and then give a brief overview of the most influential work.

Approaches to reliable multicast can be classified in two main classes, where most approaches fall in the first class:

- **Approaches based on support from inside the network.** Such approaches have support from the network in the form of additional functionality or additional resources, such as CPU and memory. Due to the additional resources provided from the network such approaches typically result in better performance. The network support for reliable multicast can be provided in the form of server nodes, located somewhere in the network, or by network nodes themselves. Support can include, but is not limited to, the capability of retransmission, the accumulation of feedback and the filtering and redirection of retransmissions to subgroups that require a retransmission.

- **Approaches that are not based on support from the network** work on an end to end basis, only including the multicast group members. Such approaches can further be divided into approaches, where only the source performs retransmissions and in approaches where potentially every receiver can affect retransmissions.

1.4.1 Landmarks

Around 1980, the first approaches towards reliable multicast were driven by satellite communication. By that time, satellites had a simple repeater function. A message sent to a satellite was simply mirrored at the satellite towards the receivers located in a large area of terrestrial coverage. The approaches to reliable multicast therefore were limited to approaches without support from the network. The proposed protocols applied unicast loss recovery techniques, such as stop and wait [6], Go-Back-N [7, 8] and selective ARQ [9, 10, 11].

With the introduction of multicast in the Internet, reliable multicast approaches with support from the network became popular. A large variety exists among these approaches.

The RMTP approach in [12] uses a fixed setup of designated receivers in a hierarchy, where designated receivers represent a multicast subgroup. Designated receivers perform retransmission via unicast and multicast to the subgroup in the case of loss and accumulate feedback down the hierarchy towards the sender.

The LBRM approach [13] is, as RMTP, an approach that requires support from the network. LBRM is based on the existence of special log-servers associated with the multicast channel. Requests for retransmissions are directed to log-servers, which in turn affect the retransmission.

A reliable multicast protocol that has support from the network directly by the network nodes themselves is proposed in [14]. Instead of locating servers in the network, or assigning a special role to some receivers, the proposal makes minimal changes to routers. The changes in the network routers allow to redirect requests for retransmission to the closest group member and allow to limit the exposure of retransmissions to the group members that experience a loss.

The idea to find the closest group member for retransmission is also present in the SRM protocol [15]. SRM does not need support from the network, but instead uses delay between group members as a distance measure. SRM requires a steady exchange of status messages among all group members to allow for delay measurements and to exchange the status of reception. For loss recovery, SRM uses a combination of probabilistic and deterministic delays that in the best case results in (i) a request to the group member located closest (in delay) to the loss and (ii) a retransmission by the group member closest (in delay) to the group member that requests the retransmission. The probabilistic component is needed to resolve concurrent requests of members that are at the same distance from the loss.

An important piece of work is the comparison of sender-based versus receiver-based reliable multicast approaches [16]. The impact of this paper is twofold, first it shows that receiver-based reliable multicast approaches scale better with the number of receivers and achieve higher throughput than sender-based reliable multicast approaches. Second, a processing analysis for reliable multicast protocols is given that was subsequently used by several authors [17, 18, 19].

An alternative approach to reliable multicast [19] shows that the number of retransmissions is tremendously reduced, when hybrid type 2 ARQ is used. Parity data is computed at the sender over original data and transmitted on request from a receiver. Receivers in turn use parity to recover from loss of any part of the original data. Important in this context is the availability of software parity coders [20] that achieve very high coding throughput of several Megabytes per second on current personal computers.

A currently active research topic is congestion control for multicast transmission. While receiver-based approaches are shown to be scalable for routing [1] and reliability [16] recent proposals suggest the same for multicast congestion control [21, 22, 23].

1.5 Thesis Contributions

We study the problem of reliable data transport from one sender to multiple receivers that are connected via a lossy multicast channel. We investigate mechanisms that ensure full reliability in the sense that all data transmitted by the sender is received by all receivers. We do not want to limit our findings to a special network and therefore consider only mechanisms that do not require support from the network.

With this general problem setting our aim is to achieve *scalability* of reliable multicast with the number of receivers. We verify our findings for different size receiver groups ranging from one receiver up to a million receivers.

The thesis contains the following contributions to the field of reliable multicast:

Modeling

One major contribution is in the field of modeling. Up to now, protocols that have been proposed for scalable reliable multicast are not shown to be scalable to very large numbers of receivers. This is due to the fact that simulation and experimental setups have a high complexity that increases with the number of receivers. We give several analytical means that allow the evaluation of reliable multicast with respect to its scalability with the number of receivers:

- *The evaluation of the loss correlation* among receivers dependent on the multicast tree topology.

- *The number of multicast transmissions* it takes for the sender until all receivers have received all packets. The number of transmissions is needed for several performance metrics: the time to complete a reliable multicast transfer, the effi ciency of a reliable multicast protocol and the network bandwidth required.

  We give new formulae that allow to assess the expected number of transmissions for independent burst loss among the receivers, for packet-level FEC and for hybrid type 2 ARQ, where parity is coded on demand.

- *Timer-based feedback mechanisms* are common means to deal with the suppression of feedback. Receivers delay feedback sending for a random time and suppress sending feedback, when feedback from another receiver is received. We give formulae that allow us to calculate the expected number of feedback messages and the feedback delay. These formulae are valid for arbitrary distributions of the timer choice at the receivers.

Besides the contributions in the fi eld of modeling, three other major contributions are made in the fi eld of reliable multicast.

Impact of Multicast Routing on Reliable Multicast

We evaluate the impact of the multicast routing algorithm on reliable multicast. We show that source-based multicast routing algorithms result in better performance of reliable multicast than non-source based multicast routing algorithms. A comparison of the multicast trees constructed by the routing algorithm with generic multicast trees show that the full binary tree (FBT) is a good multicast tree model with respect to the loss correlation among receivers and the expected number of transmissions.

Loss Recovery

For multicast loss recovery we show that parity transmission is extremely useful in the context in reliable multicast and that inherently scalable reliable multicast protocols can be built, when parities are employed in the right way:

- *A single parity packet can repair the loss of different packets at different receivers.*

We investigate parity transmission for reliable multicast protocols in different forms. We compare protocols where Forward Error Correction (FEC) is introduced as a transparent sublayer (layered FEC) and where the reliable multicast protocol is aware of parity coding (integrated FEC) and compare both cases to a protocol without FEC. We make this comparison in several settings: for a homogeneous and a heterogeneous receiver group and for temporally and spatially correlated loss.

Our fi ndings show that decoding in software at the receiver does not limit the throughput of the protocol.

Feedback

We give a new timer-based feedback method based on exponentially distributed timers. Exponentially distributed timers have better scaling properties than the usually used uniform distributed timers:

- *The feedback latency for exponentially timers is lower than with the same number of feedback messages obtained with uniform distributed timers.*

We further show that the feedback mechanism with exponentially distributed timers is *robust* against the loss of feedback messages, heterogeneous delays in the network and a wrong choice of parameters.

For the exponential feedback method we give a method that allows us to obtain an estimation of the number of receivers. The estimate adapts within seconds to a receiver population that changes by orders of magnitude.

Comparison

Finally, we quantify the difference in performance of an end to end reliable multicast protocol that uses parity retransmission by the sender and exponentially distributed timers at the receivers to a reliable multicast protocol that has support from the network. Our performance measures are the network bandwidth used and completion time for the transfer. Our findings expose that the performance of a reliable multicast protocol without support from the network is very close to the performance of a reliable multicast protocol that is supported by the network, when the amount of data to transfer is large. For small amounts of data to transfer, reliable multicast protocols with support from the network achieve better performance.

1.6 Organization of the Thesis

The following chapters are mostly self-contained, since modular subjects are treated one at a time. Chapter 2 aims to evaluate the effect of different tree topologies and different routing algorithms on reliable multicast. One outcome of this chapter is that a full binary tree is a good model for a typical multicast tree. This result is used in chapter 3, where mechanisms for retransmission and efficient loss recovery are examined. In chapter 4 we give a new method for soliciting feedback from the receivers and show that feedback is faster than with existing end to end methods. Chapter 5 compares a protocol based on the given end to end mechanisms for feedback and loss recovery to a protocol that has support from the network. Finally, chapter 6 concludes the work.

Chapter 2

Modeling

2.1 Introduction

For a large number of receivers the experimental evaluation of a reliable multicast becomes very costly. To allow for the comparison and evaluation of different reliable multicast protocols, modeling techniques are needed. In the following we compare different multicast models for the purpose of reliable multicast.

When designing or evaluating reliable multicast transport protocols, one needs to be able to compute performance measures such as delay or the number of retransmissions. We will derive the formulas for computing

- the probability mass function (*pmf*) for the number of receivers that successfully receive a packet that is emitted once.
- the mean number of retransmissions until all receivers have successfully received a packet.

Since the exact expression for the mean number of retransmissions is difficult to compute we also give a simple approximation.

Our aim is to investigate reliable transmission for multicast communication and explore its relationship to multicast routing. Recent multicast routing algorithms have been evaluated in terms of cost and delay [24, 25, 26], blocking probability [27, 28] and overhead [29]. The impact of the routing algorithm on reliable multicast transmission has not yet been studied.

Very little work [30, 31, 15] investigated the impact of the topology on reliable multicast. Bhagwat et al. [31] studied reliable multicast for three generic multicast trees. We will use similar generic trees and additionally use multicast routing algorithms to generate more realistic trees for the evaluation of reliable multicast performance dependent on the tree.

Nearly all performance studies [11, 7, 32, 10, 33] of reliable multicast communication assume multicast trees where the loss on any link affects only a single receiver. We will consider this special case of a multicast tree, referred to as *MFAN* (see figure 2.1), consider and compare it both, with trees that are the outcome of multicast routing algorithms and with two other generic multicast trees. We will show that the full binary tree (see figure 2.3) is a more realistic model for a multicast tree than *MFAN*.

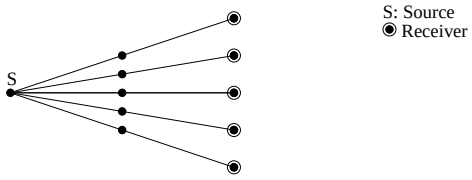

Figure 2.1: Multi-hop-Fanout (*MFAN*).

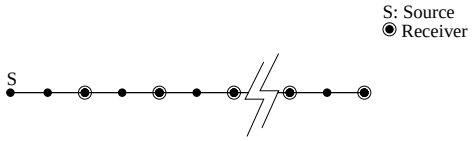

Figure 2.2: Linear Chain (*LC*).

2.2 Multicast Trees

The formulas we derive are valid for all types of multicast trees, i.e. they are independent of the topology of the multicast trees. In order to evaluate the formulas we define three generic multicast trees and additionally use two of the most popular multicast routing algorithms to compute multicast trees for artificially generated networks. A multicast tree connects a sender to R receivers. The loss in a multicast tree is dependent on the topology. A tree topology has several parameters, each of them having a different influence on loss: (i) tree height, (ii) number R of receivers (members in the multicast group), (iii) number of nodes in the tree[2], and (iv) the number of receivers affected by a loss over a single link.

We have chosen the following three generic multicast trees because they behave very differently with respect to the impact of packet loss on a single link:

- For *MFAN* (figure 2.1), always only a single receiver is affected by a packet loss on a link.
- For the linear chain *LC* (figure 2.2), depending on what link the loss occurs, the number of affected receivers can range from one to all receivers.
- For the full binary tree *FBT* (figure 2.3), the impact of loss lies between the one for *MFAN* and *LC*, affecting either a single receiver or a subgroup of all receivers.

By keeping the ratio of the number of receivers and the number of tree nodes approximately at 0.5 for all three trees (see Table 2.1) we collapse the two parameters (ii) and (iii) that influence loss into a single one. However, as the tree grows, the tree height will vary if we keep the ratio of receivers and nodes in the tree fixed (see Table 2.1). To generate "real" multicast trees we use two different multicast routing algorithms that optimize either cost or delay:

Cost optimization tries to minimize the sum of the edge costs in the multicast tree. The Kou Markovsky Berman algorithm [34], referred to as *KMB*, is a well known heuristic to approach the optimal cost solution for a multicast tree. It constructs a **Heuristic Steiner Tree** (*HST*) [35] based on the minimum spanning tree algorithm.

[2]The number of edges in a tree is not stated, since for a tree: $edges = nodes - 1$.

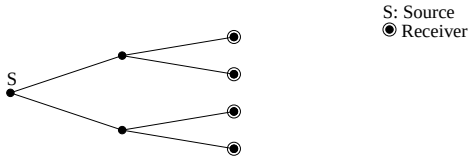

Figure 2.3: Full Binary Tree (*FBT*).

	MFAN	*FBT*	*LC*
$\frac{receivers}{nodes}$	$\frac{1}{2+\frac{1}{R}}$	$\frac{1}{2-\frac{1}{R}}$	$\frac{1}{2+\frac{1}{R}}$
tree height	2	$\log_2(R)$	$2R$

Table 2.1: The characteristic of the three generic multicast trees with respect to the number R of receivers.

Delay optimization minimizes the delay from the source to every receiver. The Shortest Path Algorithm analyzed by Doar [25] optimizes delay and constructs a **shortest path tree** (*SPT*) that connects every receiver to the source via the shortest path.

10 random networks with 200 nodes and an average outdegree of 3.0 were constructed following a method proposed by Waxman [36] with the modifi cation of Doar [25] that avoids the influence of the number of nodes on the average outdegree. The method of Waxman is commonly used by the Multicast Routing community [24, 25, 36, 37] to compare the performance of different Multicast Routing Algorithms on random networks.

On each of the 10 random nets, 100 multicast groups with varying group sizes ($5 \ldots 140$) and receivers at random locations had been routed by the two algorithms for Cost (*HST*) and Delay (*SPT*) optimization. The number of nodes in the multicast tree was the number of nodes in the *HST* or *SPT* built in the original graph of 200 nodes. Two sample multicast trees generated by the *SPT* algorithm and the *HST* algorithm for the same network and the same group of 5 receivers are shown in fi gure 2.4.

Figure 2.5 and fi gure 2.6 show the characteristics of an average *SPT* and *HST* dependent on the number R of receivers. Comparing the characteristics of *SPT*s and *HST*s to the characteristics of the generic trees given in table 2.1, it can be stated that the ratio $receivers/nodes$ of the generic trees is about 0.5, comparable to *HST*s and *SPT*s for the case of $R = 40$ receivers (see fi gure 2.5). The tree height of *SPT*s and *HST*s is the maximum number of links between the source and any receiver. Figure 2.6 shows the tree height of *SPT*s and *HST*s as a function of the number R of receivers. The tree height is modeled best by the *FBT* with its logarithmically increasing height (table 2.1) for a small number of receivers $R < 50$. For a larger number of receivers $R \geq 50$, the tree height of *SPT* and *HST* is constant, since the growth of the tree is limited by the network diameter.

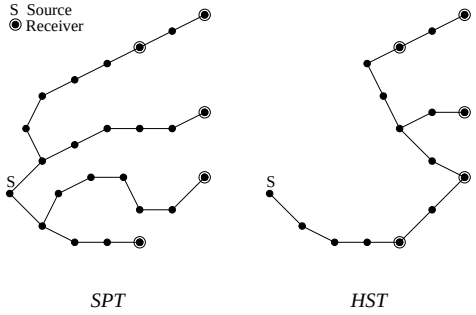

Figure 2.4: Multicast trees *SPT* and *HST* for the same group of $R = 5$ receivers in the same random network.

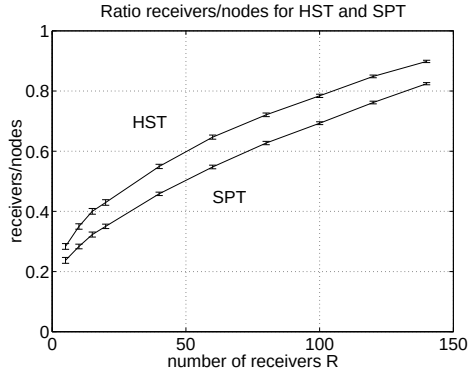

Figure 2.5: The ratio $\frac{receivers}{nodes}$ for *HST* and *SPT*.

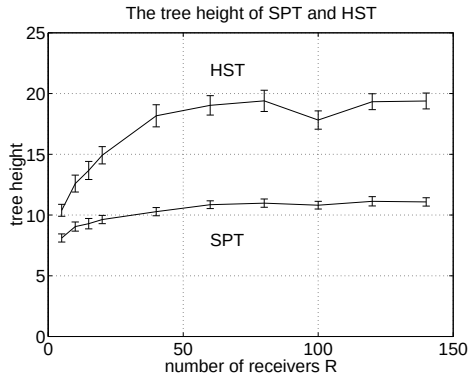

Figure 2.6: Characteristic of multicast trees *SPT* and *HST*.

2.3 Loss Characteristics of a Multicast Tree

Loss in a multicast tree affects several receivers if it happens on a link that leads to several receivers. We will call such a link a **shared link**.

Reliable multicast transmission must deal with two major problems:

- **Feedback implosion**: Receivers in a reliable multicast communication must provide the source with the status of the reception. Loss on shared links causes loss at several receivers and increases the amount of feedback.

- **High number of retransmissions**: The higher the number of receivers, the higher the number of links in the multicast tree and the average number of retransmissions.

We derive a formula to analytically evaluate the **feedback implosion** at the source, by calculating the probability mass function (*pmf*) of successful and unsuccessful receptions at R receivers for *a single packet emission*. We also show that shared links have no influence on the expected number of successful receptions.

We give the expected number of **retransmissions** needed to deliver one packet to all receivers and propose a tight approximation that enables loss prediction for adaptive error control mechanisms.

2.3.1 The Number of Successful Receptions in a Multicast Tree

Supposed that a packet is sent **once**, we are interested in the *pmf* of the number of receivers that successfully receive this packet.

Given is a multicast tree *mct* with:

- source S as the root

- R receivers placed at arbitrary nodes and at all leaves. We allow at most one receiver at any node in the tree and we assume a receiver is not at the source.

- homogeneous link loss probability q of a packet.

Let X_S be the number of receivers out of the R receivers in the multicast tree rooted at S that receive the packet successfully when transmitted once from S. We will give a method to calculate the corresponding probability mass function for $P(X_S = k)$ that enables us to capture the loss characteristic of different multicast trees. For the defi nition of the variables used in the following see table 2.2.

The *pmf* can now be calculated in a recursive way, starting at the leaves of the multicast tree. We need to distinguish two cases:

Node n is a leaf. Then there are no receivers located behind node n and the probability that no receiver is receiving a packet is 1 and the *pmf* evaluates trivially to:

$$P(X_n = 0) = 1 \tag{2.1}$$

n	A node in the *mct*.
R_n	The number of receivers in the subtree rooted at n. If n is a receiver, it is *not* included. The number R of receivers in the whole tree equals R_S.
X_n	A random variable, describing the number of receivers out of the R_n receivers in the subtree rooted at n that successfully receive a packet, when transmitted from node n.
$P(X_n = k)$	The *pmf* of X_n, $k = 0, \ldots, R_n$.
$child(n)$	The set of children (immediate successors) of n.
c_n	The number of children of n, $c_n = card(child(n))$.
$s_n \in \{0,1\}^{c_n}$	Link success vector for the links leading from n to its c_n children. $s_n(i) = 0$ indicates packet loss on the link to child i, $s_n(i) = 1$ indicates success.
$x_n \in \{0,1\}^{c_n}$	The children receiver vector. Indicates which child of n is a receiver. $x_n(i) = 1$ indicates that child i of node n is a receiver, otherwise $x_n(i) = 0$.
$a_n \in \times_{i=1}^{c_n} \{0, \ldots, R_i\}$	Behind child receptions vector. $a_n(i)$ is the number of receivers behind child i of n that received successfully.

Table 2.2: Definition of variables.

Node n is not a leaf. Then $P(X_n = k)$ is given by the sum of the probabilities of all different combinations of k successful receptions in the tree rooted at n. The recursive way of calculating the *pmf* allows the use of already known probabilities $P(X_i = a_n(i))$ at the children $i \in child(n)$ of n. For every node n we have therefore just to look at the adjacent links leading to the children.

We must sum over all the combinations of link success that allow in total k successful receiving receivers located at the children i of n and in the subtrees rooted at each of the children.

For one combination s_n of link success the number of successful receptions at the direct children, being also receivers, is given by the inner product $s_n^T x_n$. The number of receptions in the subtrees rooted at the children is given by $s_n^T a_n$.

To obtain the number k of successful receptions for a given s_n the following condition must hold:

$$k = s_n^T (a_n + x_n) \tag{2.2}$$

Since x_n is constant and s_n is given, equation (2.2) selects a subset A_n of combinations of receptions in the subtrees rooted at the children of n:

$$A_n(s_n) = \{a_n \,|\, k = s_n^T (a_n + x_n)\}$$

A different number of receptions in subtrees behind a failing link does not change the probability $P(X_n = k)$. $A_n(s_n)$ can therefore be reduced by masking the number of receptions in subtrees behind failing links.

$$A_n(s_n) = \{\ a_n \quad | \quad k = s_n^T (a_n + x_n) \\ \wedge\ \forall i :\ s_n(i) a_n(i) = a_n(i)\} \tag{2.3}$$

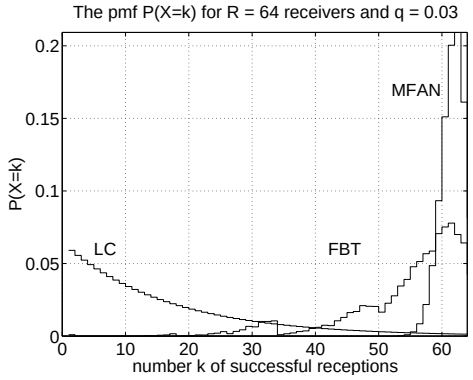

Figure 2.7: The probability mass function $P(X_S = k)$ for *FBT*, *MFAN* and *LC* for 64 receivers and $q = 0.03$.

The probability for one combination s_n of link success and one $a_n \in A_n(s_n)$ is then given by the product over the children:

$$P(a_n, s_n) = \prod_{i \in child(n)} (1-q)s_n(i)P(X_i = a_n(i)) + q(1 - s_n(i)) \tag{2.4}$$

Since the link to child i is successful ($s_n(i) = 1$) with probability $(1 - q)$ and the probability of $a_n(i)$ successful receptions in the subtree rooted at child i is $P(X_i = a_n(i))$. The packet gets lost ($(1 - s_n(i)) = 1$) on the link to child i with probability q, in which case $a_n(i)$ has no contribution.

The probability $P(X_n = k)$ is then given by summing over all link success combinations s_n and all $a_n \in A_n(s_n)$:

$$P(X_n = k) = \sum_{s_n} \sum_{a_n \in A_n(s_n)} P(a_n, s_n) \tag{2.5}$$

We depict $P(X_S = k)$ for the generic multicast trees with a link loss probability of $q = 0.03$ in figure 2.7 for $R = 64$ receivers and for $R = 128$ receivers in figure 2.8.

We can see that the *pmf*s vary significantly for the three generic multicast trees. This is due to the fact that the number of receivers affected by a loss on a single link also differs widely for the three generic multicast trees.

The *pmf* of the *MFAN* is the binomial *pmf*, the *pmf* of the *LC* approximates the geometric *pmf* for a large number of receivers. The curve of the *FBT* is multi-modal with peaks at $k = 2^{h-1}, 2^{h-1} + 2^{h-2}, \ldots$. These peaks are due to a high number of full binary subtrees with $2^{h-2}, 2^{h-3}, \ldots$ receivers and therefore a high number of possible combinations that amount to a sum of $k = 2^{h-1}, 2^{h-1} + 2^{h-2}, \ldots$ successful receptions, whereas for $k + 1$ successful receptions the number of possible combinations of full binary subtrees is much lower.

The *pmfs* for the *SPT* and the *HST* for the same multicast group in the same network (figures 2.9 and 2.10) indicate that the variance of the number of successful receptions for the *HST* is higher than for the *SPT*. The high probabilities for low numbers of successful receivers are due to loss on shared

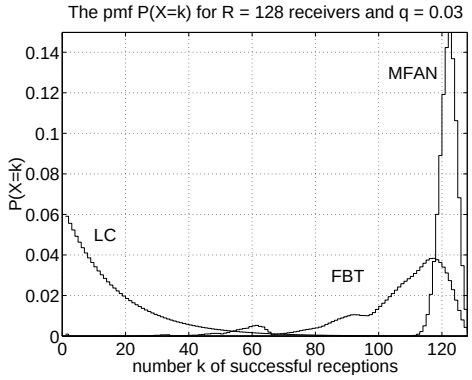

Figure 2.8: The probability mass function $P(X_S = k)$ for *FBT*, *MFAN* and *LC* for 128 receivers and $q = 0.03$.

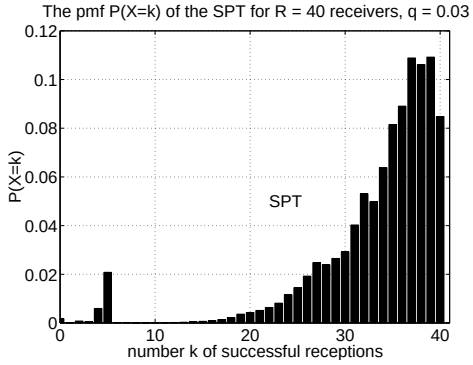

Figure 2.9: The probability mass function $P(X_S = k)$ for *SPT* with $R = 40$ receivers and $q = 0.03$.

links near the source. We observe that the $pmfs$ for the *SPT* and the *HST* resemble most closely the *pmf* for the *FBT*.

2.3.2 The Number of Responses

Given a packet is emitted once by the source, we are interested in the number of responses, which can be either positive or negative ACKs, that can be expected from the R receivers in the multicast tree. We make the assumption that the feedback channel from the receivers to the source is loss–free, in which case the number of ACKs/NAKs is identical to the number of receivers that have received or have not received a packet.

The number X_S of successful receptions in the whole multicast tree is the sum of receptions $X_{S,r} \in \{0, 1\}$ of all single receivers r: $X_S = \sum_{r=1}^{R} X_{S,r}$. Since we assume uniform link loss q on all links, the probability of a successful reception for receiver r, which lies h_r hops away from the source, is $(1 - q)^{h_r}$. The expected number of ACKs for every single receiver, which lies h_r from the source is therefore $E[X_{S,r}] = (1 - q)^{h_r}$. The expected number of successful receptions $E[X_S]$

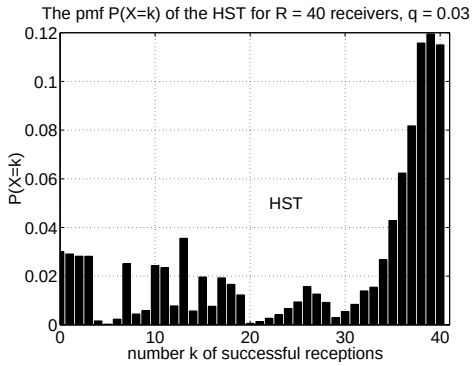

Figure 2.10: The probability mass function $P(X_S = k)$ for *HST* with $R = 40$ receivers and $q = 0.03$.

in a tree with R receivers is then:

$$E[X_S] = E[\sum_{r=1}^{R} X_{S,r}] = \sum_{r=1}^{R}(1-q)^{h_r} \tag{2.6}$$

Please note that $E[X_S]$ is *not* dependent on the number of shared links, since in (2.6) the path from the source to every receiver accounts by its full length. This is due to the fact that the expectation of a sum of random variables equals the sum of the expectations of the single random variables, even if they are not independent.

We can also express $E[X_S]$ dependent on the receiver distribution over the tree levels h, by accumulating receivers that have the same distance from the source. Let n_h be the number of receivers that lie in tree level h, e.g. h hops from the source, then the expected number of ACKs is given as:

$$E[X_S] = \sum_{h=1}^{h_{max}} n_h(1-q)^h \tag{2.7}$$

The expected number $E[X_S]$ of ACK-packets at the source is shown in figure 2.11 as a function of the number of receivers in the multicast group for a link loss probability $q = 0.03$. For *HST*, the number of ACKs is slightly lower than for *SPT*, accounting for the fact that the number of links traversed between the source and a receiver is higher for *HST* than for *SPT*.

An error control feedback scheme may use positive ACKs or negative ACKs (NAKs). Let $Y_S = R - X_S$ be the random variable that describes the number of unsuccessful receptions, then the *pmf* of Y_S is:

$$P(Y_S = k) = P(X_S = R - k)$$

and the expected number of NAKs for R receivers is given as:

$$E[Y_S] = R - E[X_S].$$

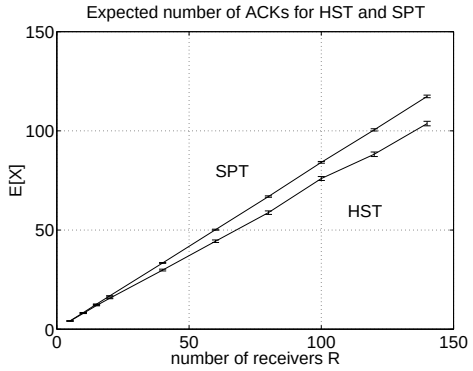

Figure 2.11: Expected number $E[X_S]$ of ACK-packets at the source for a link loss probability of $q = 0.03$.

2.3.3 The Expected Number of Transmissions for Reliable Delivery

The expected number of multicast transmissions to deliver a packet to **all** receivers is an important measure in reliable multicast communication. The expected number of transmissions captures the global packet loss behavior in the tree and the cost and the time of a reliable multicast delivery. The expected number of multicast transmissions depends on the link loss probability q and the topology of the multicast tree.
In [31], the expected number of multicast retransmissions is given for the case of loss at **nodes** in the multicast tree. It is more appropriate to consider loss on a link due to two reasons: loss at the source node is unlikely and link loss can be associated with loss in output buffers in routers. In [38], the expected number of multicast transmissions for homogeneous link loss is given by a slight modifi cation of the formula given in [31]:
Let M_n be the random variable describing the number of transmissions of a packet until it is received by node n and all receivers in the subtree rooted at n, given that the packet is always successfully received by the predecessor (parent) of node n. The Cumulative Distribution Function (*CDF*) of M_n, $F_n(i) = P(M_n \leq i)$, can be calculated in a recursive fashion, starting at the leaves of the multicast tree. It must be distinguished if node n is a leaf, an internal node, or the source S:

Node n is a leaf. In this case the probability that fewer than $i + 1$ transmissions are needed to deliver the packet over one link from the parent to the leaf is:

$$F_n(i) = P(M_n \leq i) = 1 - q^i \tag{2.8}$$

Node n is an internal node. In this case there exists one link leading to n and at least one child c. If there are i attempts to deliver the packet over the link leading to node n and it is lost exactly u times with the probability $q^u(1-q)^{(i-u)}$, then a copy of the packet is forwarded $i - u$ times on every outgoing link to every child. The conditional probability that all children of n and the nodes in the subtrees rooted at the children are receiving the packet during these $i - u$ times is

$\prod_{c \in child(n)} F_c(i-u)$. So we obtain $F_n(i)$ by summing over all possible u:

$$F_n(i) = \sum_{u=0}^{i} \binom{i}{u} q^u (1-q)^{(i-u)} \prod_{c \in child(n)} F_c(i-u) \tag{2.9}$$

Node n is the source S. In this case there is no link leading to S and consequently only the loss experienced by its children c has to be considered:

$$F_S(i) = \prod_{c \in child(S)} F_c(i) \tag{2.10}$$

Using $F_S(i)$, the expected number $E[M_S]$ of multicast transmissions from the source S is:

$$E[M_S] = \sum_{i=0}^{\infty} (1 - F_S(i)) \tag{2.11}$$

The expected number of retransmissions is:

$$E[M_S - 1] \overset{(2.11)}{=} \sum_{i=1}^{\infty} (1 - F_S(i)) \tag{2.12}$$

2.3.4 An Approximation for the Number of Retransmissions

Reliable multicast protocols need to know the expected number of retransmissions. However, the exact calculation of $E[M_S]$ as derived above is not practical for several reasons:

- The expected number of retransmissions is hard to calculate, since the calculation of the recursive *CDF* in Eq. (2.9) is computationally intensive for arbitrary topologies.
- Adaptive transport protocols need simple but effective mechanisms to decide.

We give a tight and very simple approximation. The expected number of retransmissions is approximately the product of the link loss probability q and the number of links L in the multicast tree:

$$E[M_S - 1] \approx qL \tag{2.13}$$

Consequently, the number $E[M_S]$ of transmissions can be approximated by $1 + qL$.

Proof For the sake of clarity and to shorten the proof of (2.13) Lemma 1 is used. The exact Lemma 1 and its proof is given in appendix A.1. Lemma 1 states that $F_S(i)$ can be expressed in the form

$$F_S(i) = 1 - \sum_{j_S^-} (Q_{j_S^-})^i + \sum_{j_S^+} (Q_{j_S^+})^i,$$

where the $Q_{j_S^-}$ and $Q_{j_S^+}$ are polynomials in q: $Q = \sum_k \zeta_k q^k$, with a minimal exponent $k_{min} \geq 1$. The difference between the sum $\Sigma_S^- = \sum_{j_S^-} \zeta_{1_{j_S^-}}$ of the ζ_1 of all the polynomials $Q_{j_S^-}$ with $k_{min} =$

1 and the sum $\Sigma_S^+ = \sum_{j_S^+} \zeta_{1_{j_S^+}}$ of the ζ_1 of all the polynomials $Q_{j_S^+}$ with $k_{min} = 1$ equals the number L_S of links in the tree rooted at the source S:

$$L_S = \Sigma_S^- - \Sigma_S^+$$

Using Lemma 1 the proof of Eq. (2.13) proceeds as follows. The expected number of retransmissions is:

$$\begin{aligned} E[M_S - 1] &= \sum_{i=1}^{\infty}(1 - F_S(i)) \\ &= \sum_{j_S^-} \frac{Q_{j_S^-}}{1 - Q_{j_S^-}} - \sum_{j_S^+} \frac{Q_{j_S^+}}{1 - Q_{j_S^+}} \end{aligned}$$

Then, the ratios $\frac{Q}{1-Q}$ are approximated by Q, yielding

$$E[M_S - 1] \approx \sum_{j_S^-} Q_{j_S^-} - \sum_{j_n^+} Q_{j_n^+}$$

Finally are we interested in the term q of the polynomial Q, due to its relevance compared with the terms $q^2, q^3, \ldots$. Every polynomial $Q = \sum_k \zeta_k q^k$ is approximated by $\zeta_1 q$, resulting in an approximation for the expectation of retransmissions as:

$$\begin{aligned} E[M_S - 1] &\approx q(\sum_{j_S^-} \zeta_{1_{j_S^-}} - \sum_{j_n^+} \zeta_{1_{j_S^+}}) \\ &= q(\Sigma_S^- - \Sigma_S^+) = qL \qquad \square \end{aligned}$$

The last approximation, where higher order terms are suppressed, also gives the condition for which the whole approximation of the expected number of retransmissions (Eq. 2.13) is tight:

$$qL << 1$$

For $qL \geq 1$, the polynomials $Q = \sum_k \zeta_k q^k$ can not be approximated by $\zeta_1 q$, since higher order terms become more important. For example, a second order term q^2 accounts at least as one additional link in the approximation of the expectation: $q^2 \cdot L = q(qL) \geq q \cdot 1$.

We compare the quality of the approximation for the two most extreme cases of multicast topologies. The first one is called linear chain (*LC*) and is just a chain of L links, the other one is the *MFAN*. The *MFAN*[3] has one separate link from the source to each of the L receivers. In both cases, we have L links. *LC* is the deepest, and *MFAN* is the broadest multicast tree that can be built with L links. Figure 2.12 shows that the approximation lies between the number of retransmissions of *LC* and *MFAN* for a wide range of link loss probabilities q and number L of links. The high number of retransmissions of *LC* compared to *MFAN* further shows that the tree height has a major impact on the number of retransmissions.

[3]To get the most extreme multicast tree, we reduce the tree height of the *MFAN* to 1, compared to the previous definition of *MFAN* with a tree height of 2 (figure 2.1).

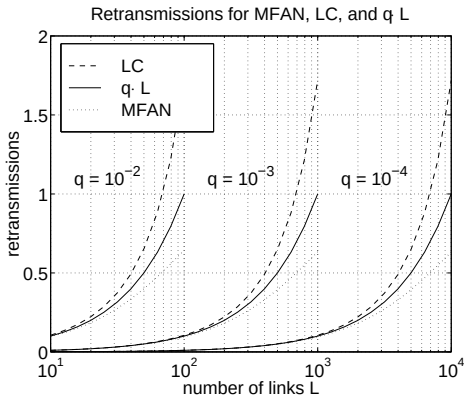

Figure 2.12: Expected number of retransmissions for *LC* and *MFAN* and the approximation qL.

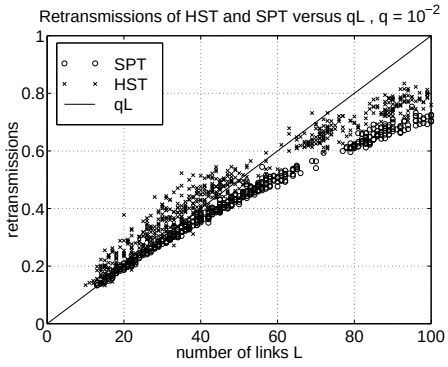

Figure 2.13: The Approximation qL versus the real number of retransmissions for *SPT* and *HST* and $q = 0.01$.

The approximation qL is also compared to the number of retransmissions of various *HST*s and *SPT*s via simulation of reliable multicast transmission. The link loss probability is $q = 0.01$. Only *SPT*s and *HST*s with up to 100 links were considered in order to meet the condition $qL < 1$ for the approximation. Figure 2.13 shows that qL approximates very well the number of retransmissions for *SPT*s and *HST*s with $L < 40$ links. Note that the *SPT*s and *HST*s represent a wide range of various tree topologies.

2.3.5 The Number of Transmissions for Large Numbers of Receivers: A step-like behavior

For an increasing number R of receivers, the number L of links in the tree increases. For a large number L of links, the approximation $E[M_S - 1] \approx qL$ can not be applied, since the condition $qL < 1$ will often not hold.

In [19] it is shown that a *FBT* can be modeled by a *MFAN* if the number of transmissions is of interest. For homogeneous link loss, the expected number of transmissions for a *FBT* with R_{FBT} receivers is approximately given by the expected number of transmissions for a *MFAN* with a smaller number of receivers $R < R_{FBT}$.

We will therefore use the *MFAN* as a model for the examination of the expected number of retransmissions for a large number R of receivers. For the sake of simplicity we assume a *MFAN*, where every receiver is connected to the source via one link ($R = L$).

The expected number of transmissions in a *MFAN* is given by Eq. (2.12) as:

$$E[M_S - 1] = \sum_{i=1}^{\infty} 1 - (1 - q^i)^R$$

Figure 2.14 shows the expected number $E[M_S - 1]$ of retransmissions for $R = 1, \ldots, 10^6$ receivers. We observe that the number of retransmissions is logarithmically increasing with the number of receivers. The function $R \mapsto E[M_S - 1]$ exhibits a step-like behavior. We further observe that as q becomes smaller the period becomes longer and the steps become flatter. Each step appears to be exactly one retransmission higher than the preceding one.

We are interested in explaining the behavior of $E[M_S - 1]$ with the number of receivers. $E[M_S - 1]$ can be rewritten using the binomial formula and exchanging summation. The obtained sum with alternating signs does not lead to an explanation of the step like behavior.

Expansion in series of $E[M_S - 1]$ results in an explanation of the steps. By expansion with the exponential series and the logarithmic series (1668, Nicolaus Mercator) we obtain:

$$\begin{aligned} E[M_S - 1] &= \sum_{i=1}^{\infty} 1 - \sum_{n=0}^{\infty} \frac{(log(1 - q^i))^n}{n!} R^n \\ &= \sum_{i=1}^{\infty} \left(1 - \sum_{n=0}^{\infty} \frac{(-1)^n R^n}{n!} \left(\sum_{k=1}^{\infty} \frac{q^{ik}}{k} \right)^n \right) \end{aligned}$$

Since the loss probability q is small, we approximate

$$\sum_{k=1}^{\infty} \frac{q^{ik}}{k} \approx q^i$$

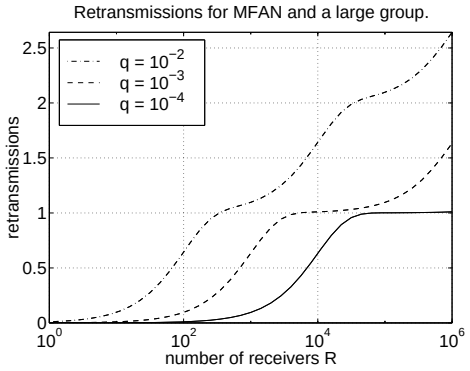

Figure 2.14: The expected number $E[M_S - 1]$ of retransmissions for a *MFAN* with different loss probabilities $q = 10^{-2}, 10^{-3}, 10^{-4}$.

and obtain this way:

$$E[M_S - 1] \approx \sum_{i=1}^{\infty} 1 - e^{-Rq^i}$$

For a given number R of receivers and a given loss probability q the terms $1 - e^{-Rq^i}$ in the sum lead to the explanation with an increasing i:

- For small i the terms become 1.
- For a certain i' the term $1 - e^{-Rq^{i'}}$ will neither become 0, nor 1.
- Increasing i further, all following terms with $i > i'$ become 0.

Therefore, the behavior of $E[M_S - 1]$ between two steps, for the corresponding range of receivers, is almost completely determined by one term $1 - e^{-Rq^{i'}}$, see fi gure 2.14. The number of terms with small $i < i'$ determine the height of step i'. Other terms have no impact on $E[M_S - 1]$.

In [39] another approximation of $E[M_S - 1]$ is obtained via Mellin transforms and bounds on the amplitude of the oscillating function are given.

2.4 Implications of our Work

We now demonstrate the impact of our results in the following two domains:

- We show that multicast routing algorithms that optimize delay achieve better delay and throughput performance for reliable multicast communication than algorithms that optimize the tree cost.
- We show that the *FBT* is a good generic model of a multicast connection and that more realistic results are obtained than with the commonly used *MFAN*.

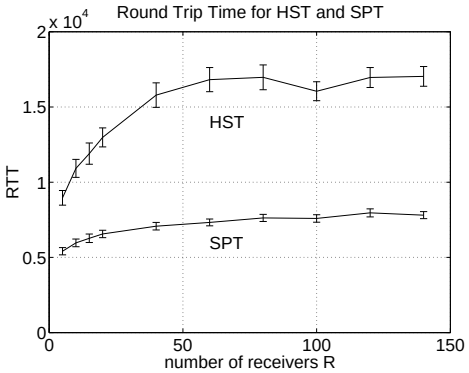

Figure 2.15: The Round Trip Time *RTT* of *SPT* and *HST*.

2.4.1 Impact of Routing on Error Recovery

Multicast routing algorithms have been designed that take mainly into account cost and delay. However, the impact of the routing on the performance for reliable transfer is left aside.

For a given loss rate, the performance of error recovery schemes for point-to-point connections is determined by the Round Trip Time (*RTT*) between the source and the receiver. We define the Round Trip Time as two times the sum of the propagation and transmission delays of the links on the path from the source to the receiver. In the following, the impact of *SPT* routing and *HST* routing on the performance of reliable delivery is evaluated by comparing the *RTT* and the number of retransmissions.

For a multicast connection, the receiver connected to the source via the longest path (in terms of delay) is the bottleneck for the error recovery scheme that uses positive ACKs. The *RTT* of a multicast connection is therefore defined as two times the sum of the propagation and transmission time on the links on this *longest* path. This path, of course, depends on the routing algorithm.

The number of retransmissions for *SPT* and *HST* is obtained via simulation, since the computation of $E[M_S - 1]$ (2.12) via (2.10), (2.9) and (2.8) is very expensive. The link loss probability is $q = 0.01$. Every point in figure 2.16 is obtained as an average for 100 trees, each tree being constructed for a different set of R receivers, where a different random network is used every 10 trees. Figure 2.16 shows that the difference between *HST* and *SPT* in terms of the number of retransmissions is minor. However, on the other hand is the *RTT* for a *HST* about two times higher than than the *RTT* for the *SPT* (see figure 2.15).

From the two observations, we conclude that delay optimization (*SPT*) in multicast routing algorithms yields better delay and throughput performance for reliable transmission than does cost optimization (*HST*).

In addition, applications with stringent time–constraints also profit from routing algorithms that optimize delay (*SPT*). In recent years, routing algorithms have been designed that optimize cost and try to meet a delay–constraint. However, most of the algorithms optimizing cost do not support dynamic multicast group membership changes – the *SPT* does.

We showed that reliable multicast performs better when the underlying multicast routing algorithm does delay optimization instead of cost optimization. Besides the benefits for reliable multicast other arguments are appealing to adopt *SPT* routing for multicast routing: *SPT* routing can use the

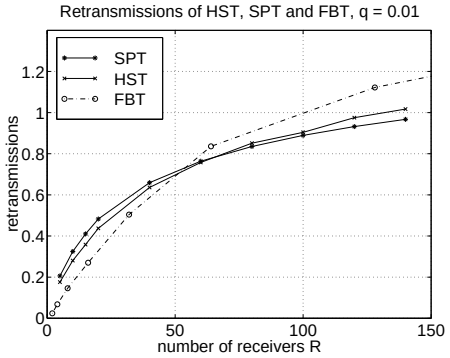

Figure 2.16: The expected number of retransmissions of *SPT* and *HST* for $q = 0.01$.

underlying unicast routing algorithm, supports dynamic membership changes, and assures good performance for time-constraint delivery.

2.4.2 A Good Multicast Tree Model: Full Binary Tree

We saw in previous sections that the loss characteristics of the *FBT* are very close to the loss characteristics of *HST* and *SPT*.

To confirm that the *FBT* is a good generic model for a multicast tree, we compare the **link share** in different trees, i.e. to what degree do receivers in a tree share common paths.

Let L be the number of links and R be the number of receivers in the multicast tree, then the link share of one link l_i, $i = 1, \ldots, L$ can be defined as the number of receivers $rd(l_i)$ that share the cost on link l_i divided by the total number of receivers: $ls(l_i) = \frac{rd(l_i)}{R}$. The link share ls for the entire tree *mct* is defined as the average link share of all links:

$$ls(mct) = \frac{1}{L}\sum_{i=1}^{L}\frac{rd(l_i)}{R} \tag{2.14}$$

For a tree, there are several methods to define a measure of link share. We compared measures of link share and found that the definition given in (2.14) reflects well the degree to which receivers share links in a tree. For a further discussion on definitions of link share see [40].

The link share of the *FBT* is nearly identical with the link share of the *SPT* (see figure 2.4.2). The *HST* has a higher link share than the *SPT* since the routing algorithm tries to connect the receiver set with a minimal cost, resulting in a high number of receivers that share an average single link in the multicast tree.

The choice of the *FBT* as a good multicast tree model due to the degree to which receivers share the links is also based on the *pmf* of the number of successful receptions. The number of successful receptions is highly dependent on the tree topology, since loss may affect several receivers - due to shared links. The similarity in shape of the *pmf* for the *FBT* (figure 2.7) and the *pmf*s of *SPT* (figure 2.9) and *HST* (figure 2.10) suggest the *FBT* as a good tree model. The *FBT* tree model is further confirmed by the number of retransmissions needed for reliable delivery. Figure 2.16 shows that the performance of the *FBT* topology is close to the performance of *HST* and *SPT*.

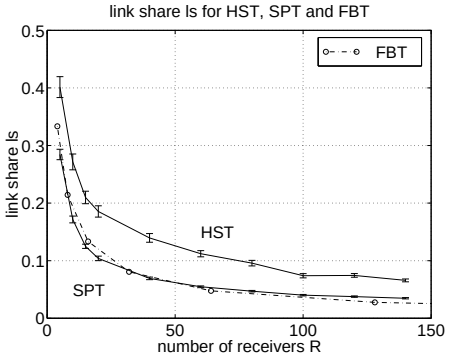

Figure 2.17: The link share ls for *SPT*, *HST* and *FBT*.

Our results so far are based on a homogeneous link loss probability. For heterogeneous link loss the *pmf* of the number of successful reception mainly depends on the location of bottleneck links with high loss probability and on the number of receivers in the subtree rooted at such a bottleneck link. The *FBT* provides a rich model that allows for a variety of heterogeneous link loss settings, including: several bottlenecks and bottlenecks in sequence.

Another model for a multicast tree is proposed in [30]. The tree model is the outcome of loss measurements on the MBONE. The authors report high loss at the source and high loss at the receivers, while backbone loss is minor. This spatial loss correlation among receivers is reflected by the proposed tree model, referred to as **modified star**: a source is connected via one link to a *MFAN* topology[1].

For homogeneous link loss, the modified star models exactly a GEO (geosynchronous earth orbit) satellite with one uplink and multiple downlinks. For more complex tree topologies the modified star does not reflect well shared loss, since only one shared link exists. Therefore the *pmf* of the number of successful receptions for the modified star is comparable to the one of the *MFAN* with an additional peak at $X_S = 0$ (compare figure 2.7).

For heterogeneous link loss, the authors [30] derive link loss probabilities from the loss measurements, such that the modified star can serve as a tree model for a small number of receivers in a real world scenario.

2.5 Conclusion

In this chapter, we evaluated the impact of multicast routing on reliable multicast and obtained two main results. First, multicast routing that optimizes delay was shown to achieve better throughput and delay performance for reliable multicast than cost optimal routing. Second, the full binary tree (*FBT*) was shown to be a good generic model for the loss characteristics of real multicast trees and provides more realistic results than the *MFAN* for which a loss affects always only one receiver. We derived two characterizations that enable the comparison of routing algorithms and error recovery mechanisms with respect to the multicast tree topology, namely a *pmf* for the number of successful receptions when a packet is emitted once from the source and the expected number of retransmissions

[1] A *MFAN* that connects every receiver with one link to the source

needed to deliver a packet from the source to all receivers. We also showed that the product $q\,L$ of the link loss probability q and the number of links L in an arbitrary multicast tree tightly approximated the expected number of retransmissions under the condition that $q\,L < 1$.

Chapter 3

Loss Recovery

3.1 Introduction

To recover from loss, two well known techniques exist: ARQ (Automatic Repeat Request), which retransmits the lost data and FEC (Forward Error Correction) which transmits redundant data, called **parity** data along with the original data. With FEC, if the amount of original data lost is not more than the amount of parity data received, the parity data can be used to reconstruct the lost original data. Coding theory [41] distinguishes between two types of errors: (i) **corruption** of data, where bits are corrupted and (ii) **erasure** of data, where the whole packet is lost. While FEC is able to recover from both types of errors, we use FEC only to recover from erasures.

FEC by itself cannot provide full reliability. However, when coupled with ARQ, FEC can be used to produce inherently scalable reliable multicast transport protocols. If introduced as a separate layer beneath the ARQ layer, it has the effect of reducing the packet loss probability and thus reducing the number of packet retransmissions and network bandwidth requirements. If integrated with ARQ, then FEC has a very high repair efficiency and, therefore, substantially reduces the network bandwidth requirements of an application requiring reliable multicast data transport. Integrated FEC/ARQ schemes are also referred to in the literature as **hybrid ARQ** [42].

In the thesis, we study the effectiveness of both approaches of combining FEC with ARQ. We find that a layered approach makes more efficient use of network resources than an approach based solely on ARQ except in conditions where losses are temporally very bursty. When integrated with ARQ, FEC provides a substantial reduction in the usage of network resources, even when losses are temporally correlated. Moreover, increased processing efficiency is achieved by the sender and receivers, even when FEC is implemented in software.

There is a large body of literature on the subject of reliable multicast - some of which focus on the use of FEC. An early paper by Metzner [43] studied the use of hybrid ARQ for reliable multi-point transmission in the context of a block data transfer with independent and homogeneous loss. Metzner suggested the use of Reed Solomon codes for parity computation and quantified the benefits of such a scheme in terms of the mean number of parity packets transmitted in the first retransmission round for a small number (up to 50) of receivers. Recently, Huitema [44] studied the benefits of FEC when used as a layer underneath the reliable multicast layer for the case of independent and homogeneous loss.

Most other reliable multicast protocols use only ARQ to assure reliability. In order to achieve scalability and avoid feedback implosion these protocols use slotting and damping [45, 46] or introduce a hierarchy [12, 47, 48]. Both solutions are not without disadvantages:

- Slotting and damping requires a careful choice/ estimation of parameters.
- Introducing a hierarchy poses the problem of selecting designated intermediate nodes that are required to have special functionalities. Also special care must be taken to cope with the failure of a designated node.

Coupled with these approaches, however, FEC can increase efficiency and scalability.

The rest of this chapter is organized as follows. Section 3.2 provides a brief overview of FEC. Section 3.3 compares the two approaches on how to combine FEC and ARQ in a reliable multicast protocol stack and evaluates their performance in the context of homogeneous and heterogeneous populations of receivers. Section 3.4 considers the performance of the two approaches in the presence of spatially and temporally correlated losses. As the focus of Sections 3.3 and 3.4 will be on network bandwidth requirements, Section 3.5 will focus on the end-host processing requirements by comparing the performance of two generic reliable multicast protocols: one with and one without FEC. Conclusions are drawn in Section 3.6.

3.2 How FEC works

3.2.1 Theory

The use of Forward Error Correction (FEC), as an alternative or complement to ARQ for reliable multicast transmission, has been investigated by different authors [43, 49, 50, 44, 38]. FEC involves the transmission of original data along with additional redundant data which can be used to reconstruct the original data if some of it is lost.

A Reed Solomon Erasure correcting code (RSE code), such as the one described by McAuley [51], is used to generate the redundant data. Suppose we have k data packets $d_1, d_2, \ldots, d_k$ each of which is P bits long. The RSE encoder takes $d_1, \ldots, d_k$ and produces **parities** $p_1, \ldots, p_{n-k}$ each P bits long. We also use the parameter h to denote the number $n - k$ of parities.

For the purpose of coding, we consider the vector $\vec{d} = [d_1, \ldots, d_k]$ of data packets as elements of the Galois field $GF(2^P)$ [41]. Given a primitive element α of $GF(2^P)$, a (n, k) matrix $G' = (g_{i,j})$ with elements in $GF(2^P)$ is defined:

$$g_{i,j} = \alpha^{i \cdot j}, 0 \leq i \leq 2^P - 2, 0 \leq j \leq k - 1 \tag{3.1}$$

The RSE coder can produce up to $n = 2^P - 1$ FEC packets as components of $\vec{y'} = [y'_1, y'_2, \ldots, y'_n]$:

$$\vec{y'} = G'\vec{d}^T \tag{3.2}$$

The matrix G' has the property that *any* k out of the $2^P - 1$ row vectors are linearly independent. Therefore, at the RSE **decoder** any k components of $\vec{y'}$ are sufficient to uniquely specify $d_1, d_2, \ldots, d_k$.

This basic RSE scheme is not a systematic code, i.e. the data packets $d_1, d_2, \ldots, d_k$ are not part of $\vec{y'}$. As a consequence, the RSE decoder must *always* solve k simultaneous linear equations to retrieve the data packets $d_1, d_2, \ldots, d_k$ from k components of $\vec{y'}$.

Fortunately, there is a simple solution to avoid this decoding complexity. Prior to coding, gaussian elimination on the matrix G' is used, to turn the first k row vectors of G' into a (k, k) identity matrix. Using this G, coding $\vec{y} = G\vec{d}^T$ results in the first k components of $\vec{y}$ being copies of $d_1, d_2, \ldots, d_k$. The other components of $\vec{y}$ are the parities: $p_i = y_{k+i}$ for $i \in \{1, \ldots, n-k\}$.

The RSE **decoder** at the receiver side can reconstruct the data packets $d_1, \ldots, d_k$, whenever it has received *any* k out of the n packets $d_1, \ldots, d_k, p_1, \ldots, p_{n-k}$. The k data packets will also be referred to as a **transmission group**, or **TG**. The n packets $d_1, \ldots, d_k, p_1, \ldots, p_{n-k}$ will be referred to as a **FEC block**. Sending the original data as the first k packets of the FEC block simplifies decoding:

- If all k data packets are received, no decoding at all is required at the receiver.
- If $l < n - k$ out of the k data packets are lost, the decoding overhead is proportional to l.

There are multiple benefits using parity packets for loss recovery instead of retransmitting the lost original data packets:

- **Improved transmission efficiency:**
  A single parity packet can be used to repair the loss of *any* one of the n data packets. This means that a *single parity packet* can repair the loss of *different data packets at different receivers*. Hence, FEC is particularly well suited for a multicast scenario.
- **Improved scalability in terms of group size:**
  An ARQ scheme which retransmits the original packets requires the sender to know the sequence numbers of the lost packets. By using parity packets for loss repair, the sender only needs to know the maximum number of packets lost by any receiver within a TG but not their sequence numbers. Feedback is reduced from per–packet feedback to per–TG feedback.
- **Reduction of unnecessary receptions:**
  Multicasting retransmissions for loss recovery results in unnecessary receptions at all receivers that do not need the retransmission. Such unnecessary receptions waste processing capacity. As we will show later, the number of unnecessary receptions is significantly reduced with parity transmission.

3.2.2 Practical Aspects

The size of a packet, P, will typically be on the order of several hundred bits (ATM cell) to several thousand bits (IP datagram). RSE coders that operate on symbols of that size are difficult to implement. The hardware architecture for the RSE coder proposed by McAuley [51] uses a symbol size m with $m = 8$ or $m = 32$. The software (SW) implementation of the coder by Rizzo [20] uses $m = 8$ and can therefore use a fast table lookup for multiplication and division.

In the case of $P > m$, we need to choose $P = S \cdot m$ where S is an integer. We then perform multiple parallel RSE encodings for each m-bit symbol in each data packet (see Figure 2 of [51]). For example, we perform RSE on the first m-bit symbol of each of the k data packets and obtain $n - k$ m-bit parity packets. We repeat this on the 2nd m bit symbol of the k data packets and so on.

The number of elements in a Galois field $GF(2^m)$ is limited to 2^m elements. Therefore, the symbol size m must be picked sufficiently large such that $n < 2^m$. For our purposes, $m = 8$ will be sufficiently large.

In order to assess the encoding and decoding speed we measured the throughput of the software coder by Rizzo on a Pentium PC 133 running Linux. The amount of original data is split equally into k data packets of $P = 1$ KByte each. The amount of redundancy, given as the percentage $h/k \cdot 100\%$ of the original data, is produced (encoding) or used for reconstruction (decoding) together with the originals received.

Figure 3.1 shows the encoding and decoding throughput. The encoding throughput denotes the amount of original data processed per second, given that a percentage $h/k \cdot 100\%$ of parity data is produced from the original data. The decoding throughput denotes the amount of parity and original data processed per second to reconstruct a percentage of $h/k \cdot 100\%$ of original data that are lost.

The analysis of the coding and decoding algorithm in [20] is confirmed by our own measurements. We observe that:

- The encoding speed is inversely proportional to the amount of parity data produced.
- The decoding speed is inversely proportional to the amount of original data lost and reconstructed, i.e. the amount of parity data used for reconstruction.
- Both encoding and decoding throughput are inversely proportional to the TG size k (see eq. B.6 and eq. B.7 in the Appendix).
- The decoding throughput is slightly less than the encoding throughput.
- For a constant TG size, k, the throughput in Mbit/s of the encoder and decoder is insensitive to the packet size P.

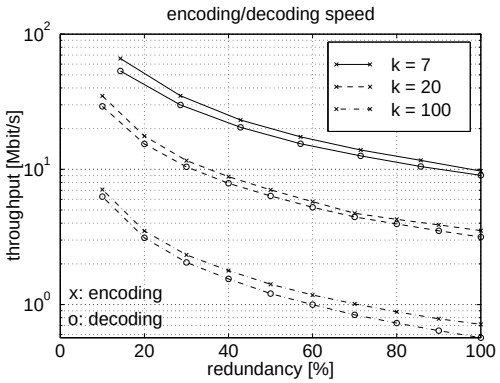

Figure 3.1: Coder and decoder throughput in Mbits/s with respect to the percentage of redundancy $h/k \cdot 100\%$ and the TG size k.

If we look at the absolute encoding performance (Figure 3.1), we observe in the case of a TG size of $k = 7$, that we achieve a coding rate of 65 Mbit/s, when one parity packet is produced (14.3% redundancy). This means the first parity packet for a TG of size $k = 7$ is available after 126 μs. Such high performance and the fact that the transmission rates of many current multicast applications are typically less than 100 KByte/s, suggests that coding in software will not affect the packet sending rate and that loss recovery using parity data is feasible. This point will be further investigated in Section 3.5.1.

3.3 Placement of FEC in a Reliable Multicast Protocol Stack

There are a number of different ways that FEC can be introduced to a reliable multicast protocol stack. The simplest approach is to add a layer responsible for FEC between the network layer and the reliable multicast layer (RM), such that the RM layer sees a more reliable network. The second approach is to integrate FEC with reliable multicast and place it into a single layer. These approaches, henceforth referred to as **layered FEC** and **integrated FEC** are illustrated in Figure 3.2.

The advantage of layered FEC is the separation of FEC and the reliable multicast layer. Thus a designer can focus on the problem of reliable multicast without worrying about the intricacies of FEC. In addition, an FEC layer can be used by other applications that may benefit from a more reliable network. Huitema [44] has established the benefits of layered FEC in terms of reducing the network traffic. We will review this work in Section 3.3.1.

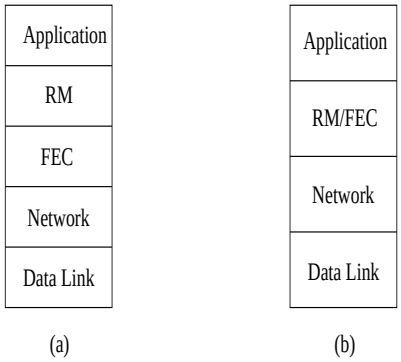

Figure 3.2: (a) Layered FEC. (b) Integrated FEC.

Given the simplicity of layered FEC and its potential performance benefits, it is reasonable to ask whether integrating FEC and reliable multicast can provide additional performance benefits that would outweigh the additional complexity required of such an approach. We shall examine this question in Section 3.3.2.

The emphasis of this section and Section 4 will be to compare the average number of transmissions required to transmit a packet reliably to all receivers using layered FEC, integrated FEC, and no FEC. The average number of transmissions is an important metric because it reflects directly the network bandwidth required to support reliable multicast. In addition, it is also correlated to the processing requirements at the end/hosts (which will be examined in Section 3.5). Last, although we do not examine the latency reduction benefits of FEC, we expect that a reduction in the required number of transmissions will often lead to a reduction in latency, especially for a multicast transmission of large amounts of data (bulk transfer).

Throughout this section we will consider a single sender with an infinite supply of packets sending to R receivers. We assume that packet losses occur as independent events, both spatially and temporally with probability p. We examine the effect that different loss probabilities have on the performance of reliable multicast in Section 3.3.3.

3.3.1 Layered FEC

Consider layered FEC based on the RSE codes described in the previous section. At the sender, the RM layer passes the packets to be sent to the FEC layer. The FEC layer constructs and sends a FEC block: it takes k original packets, produces h parities and sends all of the packets to the receiving FEC layer. Whenever the FEC layer receives an original data packet, it passes it to the RM layer and keeps a copy for decoding purposes in case of loss. Whenever the FEC layer receives at least k out of $k+h$ packets, all of the lost original packets are reconstructed and delivered to the RM layer. If fewer than $k+h$ packets are received, the lost original packets cannot be reconstructed and the FEC layer discards the received parity packets. The sending RM layer then retransmits the lost originals as part of a new FEC block.

Let $q(k,n,p)$ denote the probability that the RM receiver does not receive a random data packet from the TG sent by the RM sender. In particular, a packet from a TG is lost at the RM receiver if it is lost by the FEC receiver and if more than $h-1 = n-k-1$ of the other $n-1$ packets from the FEC block are lost. This was first proposed and studied in [44]. Therefore the packet loss probability at the RM receiver is given by

$$q(k,n,p) = p\left(1 - \sum_{j=0}^{n-k-1} \binom{n-1}{j} p^j (1-p)^{n-j-1}\right), 1 \le k \le n \tag{3.3}$$

Let M' denote the number of times an arbitrary data packet is transmitted before all RM receivers have received it. The probability that fewer than $i+1$ data packet transmissions are required by a single RM receiver is $1-q(k,n,p)^i, i = 0, 1, \ldots$. The cumulative distribution of M' is therefore:

$$P(M' \le i) = (1 - q(k,n,p)^i)^R, i = 0, 1, \ldots$$

Let M be the equivalent of M' with the additional accounting of parity packets added at the FEC layer. This quantity is difficult to define precisely; however, its average is given by

$$\begin{aligned} E[M] &= n/k \cdot E[M'] \\ &= n/k \cdot \sum_{i=0}^{\infty} (1 - P(M' \le i)) \end{aligned} \tag{3.4}$$

Figures 3.3 and 3.4 illustrate the typical behavior of layered FEC for different size transmission groups, $k = 7, 20, 100$, when the numbers of parity packets are $h = 2$ and $h = 7$. First, we observe a decrease in the expected number of transmissions when FEC is introduced and the receiver population is large; an observation first made in [44]. Second, FEC introduces a constant overhead of h/k, which results in $E[M] \approx n/k$ for a small number of receivers: this overhead results in worse performance than a non FEC protocol. Once the number of receivers is sufficiently large, the FEC overhead is amortized by the usage of FEC packets to repair different losses at different receivers and the introduction of FEC for large numbers of receivers results in a better performance than a non FEC approach.

Figures 3.3 and 3.4 show the difficulty of choosing the right parameters h and k for layered FEC. For example, a TG of size $k = 100$ with $h = 2$ parity packets exhibits worse performance than TGs of sizes $k = 7$ and $k = 20$ with the same number of parity packets. On the other hand a TG of size $k = 100$ coupled with $h = 7$ parity packets performs better than TGs of sizes $k = 7$ and $k = 20$ for

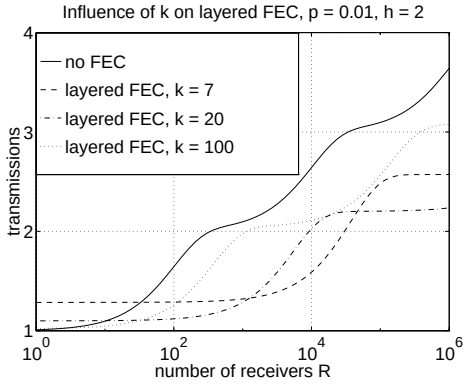

Figure 3.3: Non FEC versus layered FEC with $h = 2$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

receiver populations in the 1 to 200,000 range. We have observed similar behavior for a wide range of packet loss probabilities.

The mean number $E[M]$ of transmissions is logarithmically increasing with the number of receivers. The curve for $E[M]$ has a periodicity of $-\log(q(k, n, p))$ and a step like behavior with increments of n/k. As the loss probability $q(k, n, p)$ becomes smaller, the period becomes longer and the steps flatter. For a detailed analysis of the behavior of $E[M]$ see [39].

3.3.2 Integrated FEC

We now turn our attention to integrated FEC where the RM layer uses FEC to enhance its performance.

There are many ways that FEC can be included within the RM layer. We will propose and evaluate one such protocol in Section 3.5.1. In order to assess the potential benefits without becoming involved in a detailed analysis, we will study the following generic protocol.

- The sender sends a TG along with $a \leq h$ parity packets from the associated FEC block.
- If there are a or fewer missing packets among the $k + a$ packets sent, all packets within the TG can be recovered. Each time that a receiver detects a missing packet beyond a, it requests a new parity packet from the sender until it has received a sufficient number of packets (k) out of the n packets in the FEC block to complete the decoding of all k packets.
- The sender multicasts parity packets in response to requests until all parity packets associated with the TG have been used up. At that time, packets requiring retransmission are placed into a new TG.

We first derive an unachievable lower bound to the expected number of packet transmissions required to transmit an arbitrary packet to all receivers. This corresponds to the performance of the above protocol when $n = \infty$. Let L_r denote the number of additional packet transmissions required

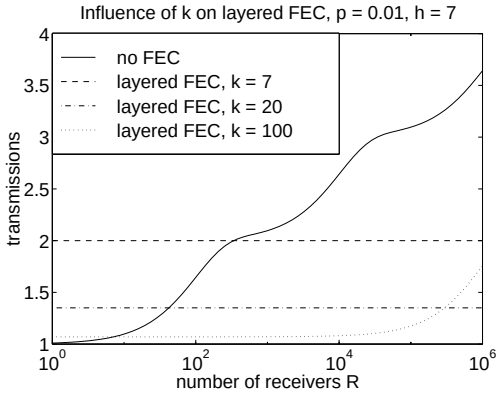

Figure 3.4: Non FEC versus layered FEC with $h = 7$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

by a random receiver. The distribution of L_r is

$$\begin{aligned} P(L_r = 0) &= \sum_{j=0}^{a} \left(\begin{array}{c} k+a \\ j \end{array} \right) p^j (1-p)^{k+a-j}, \\ P(L_r = m) &= \left(\begin{array}{c} k+a+m-1 \\ k-1 \end{array} \right) p^{m+a} (1-p)^k, \qquad m = 1, \ldots \end{aligned}$$

Let L denote the maximum number of additional packets that are transmitted. Its cumulative distribution is given by

$$P(L \leq m) = [P(L_r \leq m)]^R, \quad m = 0, 1, \ldots \tag{3.5}$$

where

$$P(L_r \leq m) = \sum_{i=0}^{m} P(L_r = i), \quad m = 0, 1, \ldots$$

The mean number of additional transmissions is

$$E[L] = \sum_{m=0}^{\infty} \left(1 - P(L \leq m)\right). \tag{3.6}$$

Finally, defining M as before, the mean number of transmissions per arbitrary packet is

$$E[M] = (E[L] + k + a)/k. \tag{3.7}$$

The protocol analyzed here sends parities pro-actively when $a > 0$. Pro-actively transmitted parities reduce the feedback and may benefit applications with time constraints.

On the other hand the previous section on layered FEC shows that transmitting unused parities lead to worse performance than with no FEC, when the number of receivers is small (compare

Figures 3.3 and 3.4). In the remainder of the paper we no longer consider pro-active sending of parities and $a = 0$.

Next we derive an expression for $E[M]$ in the case that $n < \infty$. Let B denote the number of times that an arbitrary packet was transmitted, i.e., the number of blocks transmitted that included it. Note that the transmission of the first $B - 1$ groups requires the transmission of n packets each, just like layered FEC. On the other hand, the number of packets transmitted as part of the last group is a random variable whose distribution is identical to that of L given that $L \leq n$. Given this, the expected number of transmissions is

$$\begin{aligned} E[M] &= ((E[B]-1)n + E[L \mid L \leq n])/n \\ &= \frac{n}{k}\left(\sum_{i=1}^{\infty} 1 - (1 - q(k,n,p)^i)^R\right) + \frac{1}{k}\sum_{m=0}^{n}(1 - P(L \leq m \mid L \leq n)) \end{aligned}$$

where $q(k, n, p)$ is computed using expression (3.3) and $P(L \leq m \mid L \leq n)$ is the properly conditioned version of $P(L \leq m)$ given in (3.5).

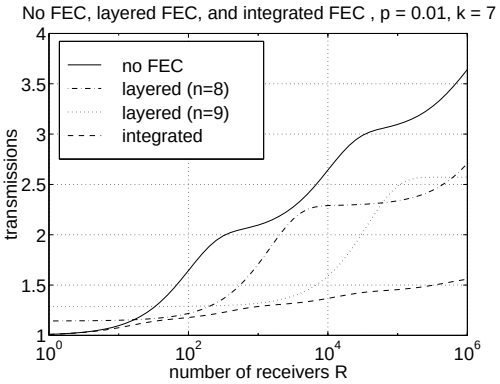

Figure 3.5: Comparison of non FEC, layered FEC ($n = 8, 9$), and integrated FEC ($n = \infty$) for $k = 7$ and $p = 10^{-2}$.

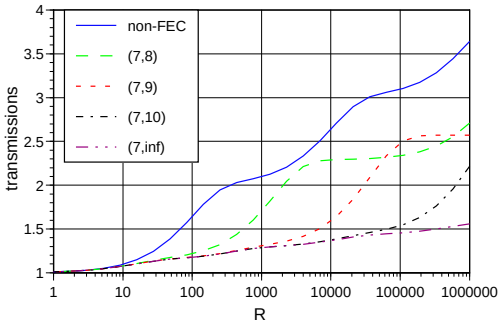

Figure 3.6: Integrated FEC with $k = 7$ and $p = 10^{-2}$ for different values of $n = 8, 9, 10, \infty$.

Figure 3.5 compares the expected number of transmissions per correctly received packet under layered FEC to a lower bound under integrated FEC for a TG size of 7 and loss probability $p = 10^{-2}$. We observe that integrated FEC has the potential for a large performance improvement over layered FEC. In Figure 3.6 we examine integrated FEC more closely to determine how many parity packets are needed to achieve a performance close to that of the lower bound. We observe that in the case of a TG size of $k = 7$, $h = 3$ parity packets suffi ce to attain the lower bound for receiver populations up to 100,000. Henceforth, we will use the lower bound when comparing integrated FEC to other approaches.

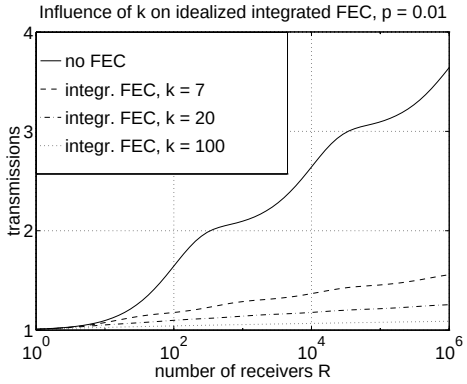

Figure 3.7: Influence of number of receivers on integrated FEC with different TG sizes $k = 7, 20, 100$ and packet loss probability $p = 10^{-2}$.

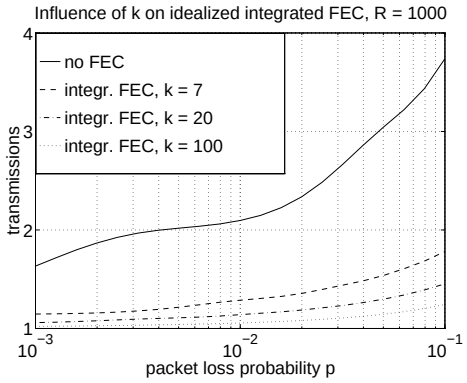

Figure 3.8: Influence of loss probability p on integrated FEC with different TG sizes $k = 7, 20, 100$ and $R = 1000$ receivers.

Last, Figure 3.7 shows the influence of the TG size k on the performance of integrated FEC. Increasing the TG size reduces the number of transmissions for integrated FEC to nearly one, even for a large number of receivers. From Figure 3.8 we observe that this behavior is fairly insensitive to the loss probability; a large increase in the loss probability has little effect on the performance of

integrated FEC.

3.3.3 Heterogeneous Receivers

We end this section with a discussion of how our observations change in the presence of heterogeneous receivers, i.e. receivers with different loss probabilities. Let $p(r)$ denote the probability that receiver $r = 1, \ldots, R$ loses a packet. If we continue to assume that losses are spatially and temporally independent, then for layered FEC

$$E[M] = \sum_{i=0}^{\infty}(1 - \prod_{r=1}^{R}(1 - q(k, n, p(r))^i))n/k \tag{3.8}$$

In the case of integrated FEC, $E[M]$ is given by equation (3.7) with

$$P(L \leq m) = \prod_{r=1}^{R} P(L_r \leq m) \tag{3.9}$$

Here $P(L_r \leq m)$ is calculated using $p(r)$ in place of p. We consider a population consisting of two classes of heterogeneous receivers; $R \cdot (1 - \nu)$ receivers with packet loss probability $p(r) = 10^{-2}$ and $R \cdot \nu$ high loss receivers with packet loss probability $p(r) = 0.25$. This allows us to vary the fraction ν of high loss receivers among all receivers.

We investigate the degradation in performance (increase in $E[M]$) as the number of high loss receivers increases. In particular, we take the percentage, $\nu \times 100\%$, of high loss receivers to be $1\%, 5\%$, and 25% of the whole group.

The results for reliable multicast without FEC (Figure 3.9) and with integrated FEC (Figure 3.10) are similar. It can be seen that the influence of the high loss receivers increases with the number of receivers. For a group of one million receivers, the presence of 10,000 high loss receivers (1% of the population) is suffi cient to double the expected number of transmissions (Figures 3.9 and 3.10). On the other hand, the presence of one high loss receiver in a population of $R = 100$ has much less effect on the expected number of transmissions. Comparing Figures 3.9 and 3.10 we observe that the presence of high loss receivers has a greater effect in the case of integrated FEC than no FEC.

We further observe that the performance is almost solely determined by the receivers with high loss rate. For integrated FEC and no FEC we observe that the increase of high loss receivers from 1% over 5% to 25% results in essentially the same curves with a linear translation in the number R of receivers on a logarithmic scale.

For real multicast groups the percentage of high loss receivers is in most cases determined by the position of a high loss router in the multicast tree and the number of receivers that experience the high loss of this router. In this case loss is spatially correlated as the receivers downstream will be equally affected by a loss. In the next section we will examine the influence of spatial correlation on reliable multicast with FEC.

3.4 Effect of correlated loss on the FEC/ARQ performance

Until now, our focus has been on a scenario where losses are spatially and temporally uncorrelated. In this section we relax each of these assumptions in an attempt to understand whether and how

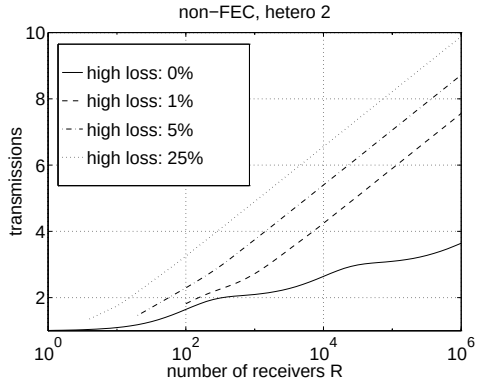

Figure 3.9: Reliable multicast without FEC for different fractions of high loss receivers.

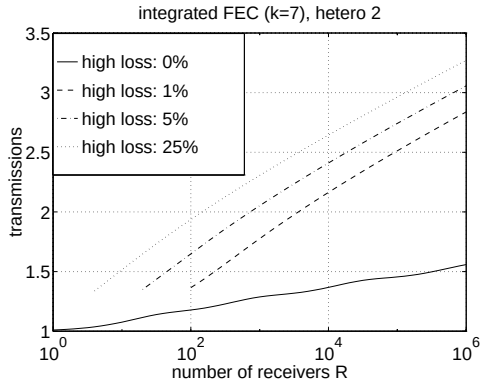

Figure 3.10: Reliable multicast with integrated FEC for TG size $k = 7$ and different fractions of high loss receivers.

correlated losses may affect our conclusions. Section 3.4.1 focusses on spatial loss correlation and Section 3.4.2 focusses on temporal loss correlation.

3.4.1 Shared Loss

Consider a sender and R receivers connected by an arbitrary multicast tree. Until now we have assumed that all losses occur as independent events at the receivers. In a real situation, however, there will be loss within the tree which will be shared by more than one receiver. In this section we explore whether and how the presence of such losses affects the conclusions which we have drawn from the independent loss model. In Chapter 2 the sharing of loss was analyzed for multicast trees built by different multicast routing algorithms. The authors conclude that the loss sharing in multicast trees is modeled well by a full binary tree (FBT). In order to investigate the impact of shared loss we consider a FBT of height d, where the source is the root of the tree and the receivers are the leaves. We compare FEC for:

- **Shared loss**: Losses occur as independent events at each node (including source and leaves) with probability p_n. Here p_n is chosen such that the loss probability at each receiver is p:

$$p = 1 - (1 - p_n)^{(d+1)}.$$

- **Independent loss**: Only the receivers lose packets, each receiver independently with probability p. Other nodes of the multicast tree do not lose packets at all.

Note that each receiver experiences the same loss probability in both models and that there is no temporal loss correlation.

The expected number of transmissions required to correctly transmit a packet reliably over a FBT was first derived in [31]. Because the calculation of this quantity is computationally intensive for large numbers of receivers, we use simulation to investigate the impact of shared loss for numbers of receivers: $R = 2^d$, $d = 0, \ldots, 17$. The packet loss probability is $p = 10^{-2}$, FEC was evaluated under the assumption that transmission groups were of size $k = 7$ with one transmitted parity packet ($h = 1$) in the case of layered FEC.

The impact of shared loss on FEC is shown in Figure 3.11 for layered FEC and in Figure 3.12 for integrated FEC. First, we observe that the mean number of transmissions is lower (often substantially) when losses are shared than when they are independent. Second, we observe that our observations drawn from the independent loss model in the previous section continue to hold. However, receiver group sizes need to be larger before the benefits of layered FEC appear. This is because a parity does not exhibit the same repair efficiency under shared losses as it does in the presence of independent losses and may be transmitted needlessly. Figure 3.11 shows that the overhead of transmitted parities with layered FEC is amortized by the repair efficiency for a number of receivers $R > 60$ in the case of shared loss, whereas in the case of independent loss, layered FEC is already efficient for $R > 20$.

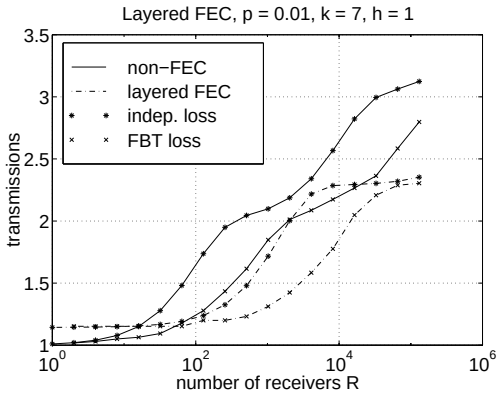

Figure 3.11: Layered FEC with $k = 7$ and $h = 1$ versus non-FEC for Independent and for FBT Shared Loss with loss probability $p = 0.01$.

Another useful observation is that the curves for shared loss appear as translated versions of the independent loss curves and that the performance of a group of size R, regardless of whether FEC is used or not, can be determined by studying the behavior of a group of size $R_{indep} \leq R$ under the

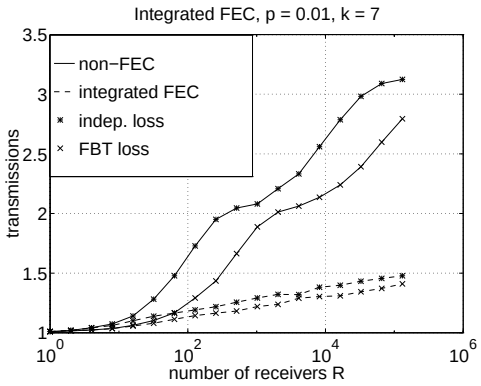

Figure 3.12: Integrated FEC with $k = 7$ versus non-FEC for Independent and for FBT Shared Loss with loss probability $p = 0.01$.

independent loss model. In fact, the extreme case is when all losses are shared by all receivers in which case the system can be modeled by a single receiver under the independent loss model. This carries the following important implication:

- Adaptive transport mechanisms that are based on measurements of receiver loss rates will overestimate the number of transmissions of reliable multicast if they model losses as independent events. Coupled with FEC this could lead to an overestimate of the amount of redundancy needed.

In summary, our results show that shared loss will result in a lower number of transmissions compared to independent loss for all recovery schemes and that the improvement of reliable multicast transmission due to FEC (compare non–FEC and FEC) is lower when losses are shared than when they are totally uncorrelated.

3.4.2 Burst Losses

In this section we reexamine the benefi ts of FEC when losses are bursty. In particular, we assume that packet losses are described by a two state continuous time Markov chain $\{X_t\}$ where $X_t \in \{0, 1\}$. A packet transferred at time t is lost if $X_t = 1$ and not lost if $X_t = 0$. The infi nitesimal generator of this Markov chain is

$$\mathbf{Q} = \begin{bmatrix} -\mu_0 & \mu_0 \\ \mu_1 & -\mu_1 \end{bmatrix}$$

The stationary distribution associated with this chain is $\pi = (\pi_0, \pi_1)$ where $\pi_0 = \mu_1/(\mu_0 + \mu_1)$ and $\pi_1 = \mu_0/(\mu_0 + \mu_1)$. Let $p_{i,j}(t)$ denote the probability that the process is in state j at time $t + \tau$ given that it was in state i at time τ, $p_{i,j}(t) = P(X_{\tau+t} = j \mid X_\tau = i)$. These probabilities are given

by [52, ch. 6].

$$p_{i,j}(t) = \begin{cases} \mu_1(1-\exp(-(\mu_0+\mu_1)t))/(\mu_0+\mu_1), & i=1, j=0, \\ \mu_0(1-\exp(-(\mu_0+\mu_1)t))/(\mu_0+\mu_1), & i=0, j=1, \\ (\mu_0+\mu_1\exp(-(\mu_0+\mu_1)t))/(\mu_0+\mu_1), & i=1, j=1, \\ (\mu_1+\mu_0\exp(-(\mu_0+\mu_1)t))/(\mu_0+\mu_1), & i=0, j=0 \end{cases}$$

for all $t > 0$.

We now consider what effect this kind of loss process has on the expected number of packet transmissions required for each correctly received packet in the absence of FEC, with layered FEC, and with integrated FEC. In all cases, we assume that the loss processes are independent from receiver to receiver.

Let $\lambda = 1/\Delta$ be the packet transmission rate and $\bar{b}$ be the expected number of consecutively lost packets. Given the packet loss probability p, the average burst loss length $\bar{b}$ in packets and the sending rate λ, the parameters for the loss model are:

$$\begin{aligned} \mu_0 &= -p\lambda \log\left(1 - 1/\bar{b}\right) \\ \mu_1 &= \mu_0(1-p)/p \end{aligned}$$

When burst losses occur, the timing of the retransmissions influences the performance of loss recovery. To further investigate this point, we consider different cases as shown in Figure 3.13.

- **No FEC**: In the absence of FEC, we assume the time between successive transmissions of the same packet to be spaced by time $\Delta + T$.
- **Layered FEC**: For layered FEC, we assume the time between the sending of the last packet of a FEC block and the time of the sending of the first packet in the successive FEC block, containing the retransmission, to be spaced by time $\Delta + T$. We further assume that a packet keeps its place in the FEC block for retransmission.

For integrated FEC, the timing considerations depend on the protocol that implements loss recovery by parity transmissions. We distinguish between two cases:

- **Integrated FEC 1**: Parity packets are transmitted with the same rate $1/\Delta$ immediately following the original packets of the TG. When a receiver has received enough parity packets for the TG, it leaves the multicast group and therefore stops receiving packets. No feedback is needed for loss recovery and there is no unnecessary delivery and reception of parity packets, provided that the time needed to depart from the group is smaller than the packet inter-arrival time. We limit ourselves to the transmission of one TG, though several TGs can be delivered, staggered in time on different multicast channels.
- **Integrated FEC 2**: This protocol corresponds to a hybrid ARQ protocol, where receivers send NAKs indicating the number of missing packets. Feedback is sent after the transmission of the original packets, after the first retransmission of parities, etc.. Subsequently the sender transmits the maximum number of parity packets needed by any receiver.

Integrated FEC 2 is motivated by the fact that interleaving improves the performance of FEC in case of burst loss [41]. Interleaving allows the sender to spread the transmission of a FEC block over an interval that is longer than the burst-loss length. The benefi t of interleaving is that burst loss is transformed into random loss. Integrated FEC 2 spaces parity transmissions out by intervals of length $\Delta + T$, whereas Integrated FEC 1 sends all parities just spaced by Δ. The term interleaving comes from the fact that packets from different TGs can be sent simultaneously in an interleaved manner. Note that during the gaps of duration T packets belonging to other TGs are sent.

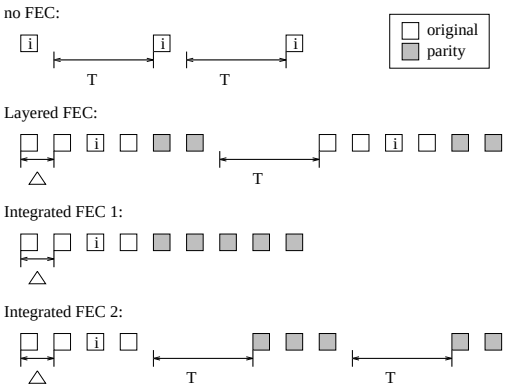

Figure 3.13: Timing considerations of the different approaches for data and parity retransmission.

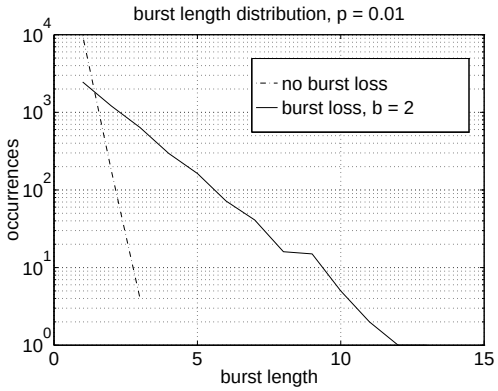

Figure 3.14: Distribution of number of consecutive losses at one receiver, for no burst and burst loss ($\bar{b} = 2$) for a packet loss probability of $p = 0.01$.

First, we will analyze no FEC under the burst loss model. Let M_r denote the number of transmissions required for receiver r to receive an arbitrary packet. Its distribution is

$$P(M_r = m) = \begin{cases} \pi_{0,} & m = 1, \\ \pi_1 p_{1,1}(\Delta + T)^{m-2} p_{1,0}(\Delta + T), & m = 2, \ldots \end{cases}$$

Let M denote the total number of transmissions required to get the packet to all R receivers, It has cumulative distribution

$$P(M \leq m) = \left(1 - \pi_1 p_{1,1}^{m-1}(\Delta + T)\right)^R, \qquad m = 1, \ldots$$

The expectation of M, is given as

$$E[M] = \sum_{m=1}^{\infty} 1 - P(M \leq m)$$

We use simulation to examine the impact of burst loss on the FEC schemes. We choose an average burst length of $\bar{b} = 2$, and $\Delta = 40$ms corresponding to a sending rate of $\lambda = 25$ packets/s as reported by Bolot [53] for a loaded IP path between INRIA (Sophia Antipolis, France) and UCL (London, UK). The packet loss probability is chosen to be $p = 0.01$, T is chosen to be 300 ms.

Figure 3.14 illustrates the burst length distribution at one receiver under the independent and burst loss models for these parameters. It can be seen that the tails of both distributions decrease linearly on a logarithmic scale.

We observe from Figure 3.15 that layered FEC performs worse than reliable multicast without FEC in the presence of burst losses when the TG consists of $k = 7$ packets. A larger FEC block size $n = 20 + 7$ permits layered FEC to perform slightly better than no FEC only if the number of receivers is large. However increasing the FEC block size is not always desirable since the FEC layer is no longer transparent to the RM layer due to high recovery latencies.

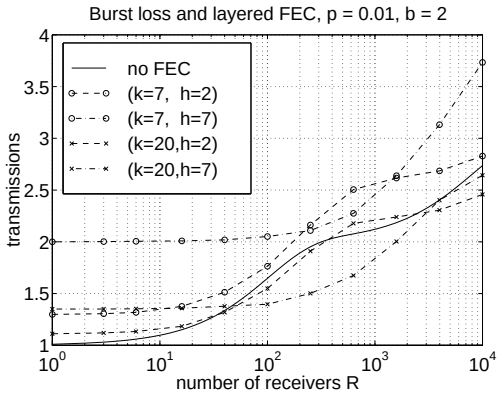

Figure 3.15: Comparison of reliable multicast without FEC and with layered FEC for small ($n = 7 + 2$, $n = 7 + 7$) and large FEC blocks ($n = 20 + 2$, $n = 20 + 7$), $p = 0.01$ and $\bar{b} = 2$.

While large TG sizes are not desirable under layered FEC, it is reasonable to consider large TGs under integrated FEC. Figure 3.16 shows the performance of integrated FEC 1 and integrated FEC 2 for different TG sizes ($k = 7, 20, 100$). For a small TG size of $k = 7$, integrated FEC 1 and integrated FEC 2 outperform reliable multicast without FEC only slightly in the presence of burst loss. Integrated FEC 2 performs better than integrated FEC 1 for $k = 7$, since parity packets belonging to the same TG are spread out over time (see Figure 3.13) and are more likely to bridge a loss period. Figure 3.16 also shows that increasing the TG size from $k = 7$, to $k = 20$ and $k = 100$

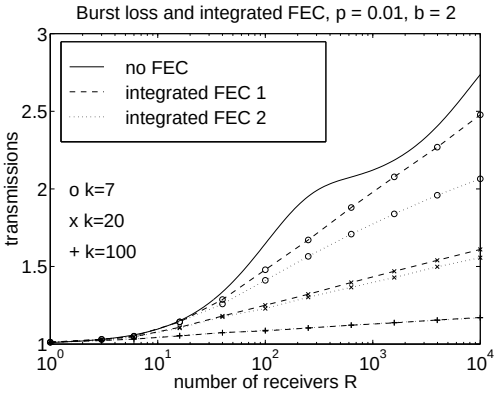

Figure 3.16: Comparison of integrated FEC 1 and integrated FEC 2 for TG sizes $k = 7, 20, 100$, $p = 0.01$ and $\bar{b} = 2$.

significantly improves the performance of integrated FEC. Furthermore, there is little difference between integrated FEC 1 and integrated FEC 2 primarily due to the fact that the transmission of a TG is spread over a sufficiently long period of time to span a loss period such that subsequent parity packets are unlikely to be affected by it. This shows that a large TG size is sufficient to resist burst loss and that additional interleaving (integrated FEC 2) is not necessary.

3.5 End-host Throughput of a Hybrid ARQ Protocol

In the previous sections we showed that integrating FEC with reliable multicast greatly reduces the expected number of transmissions over reliable multicast without FEC. This reduction does not come for free however, since there are processing requirements at the sender and the receivers for coding and decoding in the case of loss. We will now evaluate the processing load at sender and receivers and show how the use of integrated FEC affects the achievable end-host throughput of the reliable multicast connection. We will first present a reliable multicast protocol using integrated FEC, called NP, and then compare it with a generic version of a reliable multicast protocol without FEC, called N2.

3.5.1 Protocol NP

There are numerous ways to design a reliable multicast protocol with hybrid ARQ. The design choices are largely influenced by considerations for the specific type of application, e.g., file transfer or audio/video transport and its constraints such as high efficiency or low latency.

Protocol NP emphasizes efficiency at the expense of latency by only transmitting as many parities as are required to reconstruct a TG. NP could be used, for instance, by a reliable file transfer application. Protocol NP is similar to the Integrated FEC 2 scheme from Section 3.4.2, i.e., parity packets are retransmitted in response to received NAKs. A single multicast group is used for the transmission of the data and the parity packets. The entire data block is broken up into multiple TGs, $TG_1, \ldots$, each consisting of k data packets.

Feedback from the receivers consists of the multicast transmission of NAKs coupled with NAK suppression as in SRM [45].

The transmission of $TG_i, i = 1, \ldots$, proceeds in rounds which are interleaved with the rounds of other TGs.

- Round 1: The k data packets of TG_i are sent
- Round $j > 1$: $l^{(j-1)}$ parities for TG_i are sent, where $l^{(j-1)}$ is the maximum number of packets over all receivers that still need to be received after $j-1$ rounds to reconstruct the k data packets of TG_i.

The Sender:

- Transmits the k data packets in transmission group TG_i.
- When done, the sender polls (`POLL(i,k)`) the receivers for feedback about the number of packets missing to reconstruct TG_i and continues by sending the data packets of TG_{i+1}.
- When the sender receives `NAK(i,l)` (see below), it interrupts sending data packets of TG_m if $m > i$. The sender then transmits l new parities for the data packets in TG_i and polls the receivers (`POLL(i,l)`) for feedback about the remaining number of packets required to reconstruct TG_i. It then resumes transmission of the interrupted transmission group TG_m.

The Receiver:

- Stores data packets and parities for TG_i until k packets are received, which allows the receiver to reconstruct TG_i.
- When a `POLL(i,s)` is received, the receiver computes the number of packets, l, needed to reconstruct transmission group TG_i and schedules a timeout for returning this information (`NAK(i,l)`) to occur in the interval $[(s-l)T_s, (s-l+1)T_s]$. Here the **slot size** T_s is chosen by taking the requirements of the application (low latency, high effi ciency) into account.
- When the timeout for `NAK(i,l)` occurs, `NAK(i,l)` is re-sent. The timer for `NAK(i,l)` is canceled on the reception of `NAK(i,m)` with $m \geq l$.

With our slotting and damping mechanism the sender will ideally receive a single `NAK(i,l)` after every round as a reply that indicates the *maximum number* of packets needed by any receiver to reconstruct TG_i.

Protocol NP is similar to protocol N2 of [16] in several aspects: feedback is receiver-initiated and NAKs are sent via multicast. A receiver receiving a NAK for a particular round will not generate a NAK for that round. The major differences between NP and N2 are that NP requires feedback for a TG of k packets rather than for *individual* packets and that NP transmits parity packets for loss recovery while N2 retransmits the original packets that are lost.

In order to quantify the performance impact of the differences between N2 and NP, we compare their processing rates at the sender and receiver and their throughput for the case of a one-to-many transmission. Let Λ_s^w, Λ_r^w be the per-packet send and receive processing rates of protocol $w \in \{N2, NP\}$. The achievable end-system throughput Λ_o^w is defi ned as the minimum of the sender and receiver processing rates:

$$\Lambda_o^w = min\{\Lambda_s^w, \Lambda_r^w\} \tag{3.10}$$

In the following, we compare the processing rates for protocols N2 and NP as a function of the number of receivers. The processing rates $\Lambda_s^{N2}, \Lambda_r^{N2}$ were computed in [16]. The computation of the processing rates $\Lambda_s^{NP}, \Lambda_r^{NP}$ is given in the appendix. To obtain the results, we used the same values for the various processing times as [16] along with our own measured values (on the same DECstation 5000/200 running ULTRIX 4.2a as in [16]) for the encoding and decoding times based on the coder reported in [20]. In our throughput calculations we use $E[X_p] = E[Y_p] = 1000\mu$sec (average processing times for sending or receiving a 2 KBytes packet) and $E[X_n] = E[Y_n] = E[Y_n'] = 500\mu$sec (average processing time to send or receive a NAK). We use $E[X_t] = E[Y_t] = 24\mu$sec for the timer overhead. We measured as coding constants $c_e = 700\mu$sec and the decoding constant as $c_d = 720\mu$sec for 2 KBytes packets and a symbol size of $m = 8$. The reader is referred to the appendix for definitions of the above quantities.

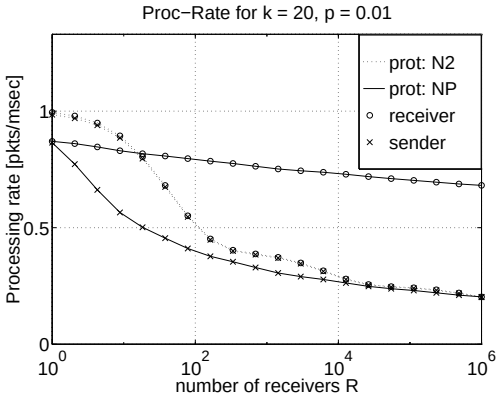

Figure 3.17: Processing rates at sender and receiver for protocols N2 and NP with packet size $P = 2$ KBytes, $k = 20$, and $p = 0.01$.

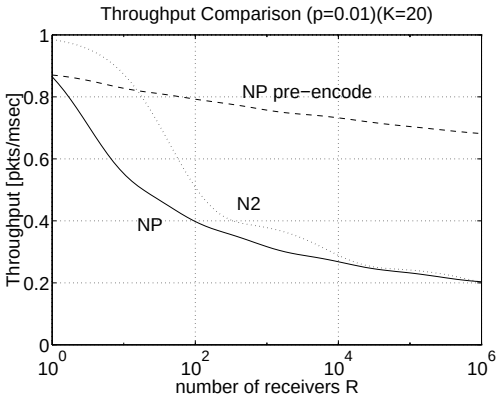

Figure 3.18: Throughput [pkts/msec] for N2 and for NP with and without pre-encoding for packet size $P = 2$ KBytes, $k = 20$, and $p = 0.01$.

In Figure 3.17, we see that the sender and receiver processing rates are nearly identical for protocol N2. The processing rates are largely determined by the mean number of transmissions and NAKs to be processed per packet (see Eq. (B.1) and (B.2)). They decrease as the number of receivers increases due to the fact that the mean number of transmissions per packet increases.

For NP, the processing rate at the sender is largely determined by the packet processing times and the encoding times, both of which *depend linearly* on the mean number of transmissions (see Eq. (B.4) and (B.6)). At the receiver, the processing rate is largely determined by the decoding time, which is *independent* of the number of receivers (Eq. (B.7)) and the packet processing time, which increases linearly with the mean number of transmissions.

The processing overhead due to FEC is much higher at the sender than at the receivers: the sender must encode a number of parities, (with expected value $E[M^{NP}] - 1$) sufficient to allow the reconstruction of the data packets of a TG by *all* receivers. An individual receiver needs to decode (reconstruct) an average of $k \cdot p$ packets per TG. Therefore, the receiver processing rate is much smaller than the sender processing rate.

Protocol NP contains two improvements over N2: loss recovery via parity retransmission and feedback reduction due to the use of a single NAK *per transmission round* instead of per missing packet. By slightly modifying the analysis in the appendix, (Eq. (B.4) and (B.5)) we can obtain the processing rates for the case that *one NAK is returned per missing packet*. The results indicate that reducing the NAKs to one per transmission round, as does protocol NP, has only a minor effect on the processing rates; the sender processing rate does not change and the receiver processing rate decreases only slightly for a very large number of receivers.

The processing rates obtained show that NP scales with the number of receivers, R, since the processing bottleneck is a single point – the sender. For N2, this is not the case, since sender and receivers have the same processing rates. If required, there are some simple solutions to match the speed of the NP sender and the NP receivers: *(i)* the sender can **pre-encode** the packets off-line and store the parity data together with the original data prior to transmission on disk, *(ii)* a more powerful machine can be used at the sender, or *(iii)* dedicated hardware can be used for encoding. Figure 3.18 compares the throughput given by Equation (3.10) for N2 and NP with and without pre-encoding, for $k = 20$, $p = 0.01$. It demonstrates the extent to which encoding impacts the performance of the NP protocol. It can be seen that the throughput of NP with pre-encoding is higher than the throughput of N2 and NP without pre-encoding, even for a small number of receivers.

3.6 Summary

Using FEC in an integrated fashion provides five major benefits:

- Integrated FEC shifts resource usage from the network to the end-systems: the number of transmissions is reduced and, therefore, the network bandwidth used as well – at the extra cost of coding/decoding in the end-systems.
- The achievable end-system throughput for protocols based on integrated FEC is higher than for non-FEC protocols when data is pre-encoded.
- Error-control feedback is reduced: feedback is returned for each transmission group (consisting of k packets) rather than for each packet.
- A moderate TG size of $k = 20$ will tolerate burst losses, even without interleaving.

- Scalability with the number of receivers is achieved for reliable multicast up to 1 million receivers.

We can draw the following conclusions:

- Integrated FEC dramatically reduces the mean number of transmissions as compared to the use of no FEC.
- Integrated FEC is better than layered FEC for all parameters. In addition, low redundancy is sufficient to achieve idealized integrated FEC.
- Layered FEC can reduce the number of transmissions in the case of large receiver populations. However, unlike integrated FEC, its performance is sensitive to the coding parameters and the presence/absence of burst losses. Layered FEC may be useful for applications with delay constraints; this is a topic for future work.
- High loss receivers determine the performance of a reliable multicast, even if the fraction of high loss receivers among all receivers is very small.
- The repair efficiency of FEC is in the case of shared loss not as high as in the case of independent loss. For a given number of receivers, shared loss can be modeled by a reduced number of receivers that suffer independent losses.
- For burst loss, layered FEC can be worse than no FEC. When losses are bursty, the performance of integrated FEC decreases, especially for small values of k. In this case, interleaving can improve performance. However, integrated FEC with a large TG size k does not need interleaving.
- For protocols based on integrated FEC (such as NP) the sender is the bottleneck. Pre-encoding the parity packets or using a higher performance sender machine can yield an end-system throughput that is three times higher than for a reliable multicast protocol with no FEC, even when receivers decode online.

Chapter 4

Multicast Feedback

4.1 Introduction

A major challenge in multicast communication is the *feedback implosion* that occurs when a large number of receivers send feedback simultaneously to the sender.

We investigate feedback of groups up to 10^6 receivers towards a single sender, in the cases that feedback is used for:

- *Reliable multicast*: Reliable multicast guarantees the delivery of data from the sender to every receiver. Feedback messages (FBMs) are needed in order to signal the loss (NAK), or the successful reception of data (ACK).
- *Estimation of the number of receivers*: is required to adapt scalable protocols to the number of receivers, e.g. by adjusting the amount of FEC [19], or to adjust the period of periodic control message emission.

The amount of potential feedback increases with the number of receivers and may lead to a high traffic concentration at the sender, wasted bandwidth, and high processing requirements. The potential for feedback implosion requires a mechanism for *feedback implosion* avoidance. Several solutions exist for implosion avoidance based on hierarchies, timers, tokens, and probing; see section 4.8 on related work.

Dependent on the kind of feedback, different mechanisms are suitable to avoid feedback implosion. Three different types of multicast feedback can be distinguished:

- **Redundant feedback**: If feedback from one group member is sufficient, feedback from further group members is redundant and may be suppressed. This is the case for a NAK.
- **Full feedback**: In this case feedback from every single group member is required and the meaning of a feedback message depends on the feedback sender. This is the case for an ACK, where no suppression is possible.
- **Group feedback**: group feedback requires information about every group member, but the identity of the feedback sender is not important. This kind of feedback is well suited for aggregation. An example is the number of receivers in the group, or a histogram of number of group members per loss rate.

Our concern is loss recovery feedback. In the case where the sender needs to be aware about the reception status of every single receiver ACKs must be sent by receivers to acknowledge a successful reception. NAKs alone are not suffi cient to provide awareness at the sender, since NAKs might be lost. One method of avoiding implosion for full feedback is given in RTCP [54], where feedback message sendings are uniform randomly delayed over a time interval with a size proportional to the number of receivers.

In [16] it is shown that receiver–based loss detection (NAK feedback) results in a higher throughput for reliable multicast than does a sender–based loss detection (ACK feedback). These conclusions were drawn under the assumption that a feedback suppression mechanism guarantees the suppression of redundant feedback. For NAK feedback implosion, avoidance is not as easy, since the number of receivers that wish to send a NAK at the same time depends on the loss and may vary from no receiver to all receivers.

Very little work [55, 15, 56] was done on the analysis of timer-based schemes for multicast feedback. We give the analytical foundation of timer-based feedback where the timer choice, the sender-receiver delays and the delays between receivers can be modeled by arbitrary distributions. The analysis allows us to compute:

- The expected number $E[X]$ of FBMs returned to the sender.
- The expected feedback delay $E[M]$ due to the timers.

We propose a new probabilistic feedback method for multicast based on exponentially distributed timers and show by analysis and simulation for up to 10^6 receivers that feedback implosion is avoided. We show the robustness of our mechanism to loss of FBMs, to homogeneous delays, and to heterogeneous delays.

We further evaluate our mechanism in the context of reliable multicast with respect to NAK implosion avoidance and to NAK latency. A comparison of our mechanism with existing timer-based feedback schemes shows that the feedback latency of our mechanism is lower for the same performance in NAK suppression.

Our mechanism requires very little state and has a low computational complexity at every receiver – independent of the group size. No knowledge about the network topology, nor support from the network is required to implement implosion avoidance.

Using an estimate of the number R of receivers, our feedback mechanism allows us to adjust the average number of FBMs returned to any value greater than 1 by trading off fewer FBMs for an increased feedback latency.

The remaining part of the chapter is organized as follows. In section 4.2 we present an analysis for timer-based feedback schemes. In section 4.3 we evaluate the performance for reliable multicast feedback. Section 4.4 shows the robustness of timer-based feedback for loss and heterogeneous delays. The control of the amount of feedback is discussed in section 4.5. The timer-based feedback scheme for networks providing only a unicast feedback channel is discussed in section 4.6. Section 4.7 shows how an estimate of the group size is obtained. Section 4.8 discusses the work in the context of related work and section 4.9 concludes the work.

4.2 Timer-based Feedback

Consider the case where a sender needs to receive at least one FBM from R receivers and where the total number of FBMs returned should be small in order to avoid *feedback implosion*.

We consider a feedback mechanism with *feedback suppression*: A receiver that receives a FBM of another receiver will suppress its own feedback sending. FBMs are sent on a multicast feedback channel to be received at other receivers. If every receiver delays its multicast feedback sending by a random time, feedback implosion can be avoided. In section 4.6 the necessary modifications are given for the case, where receivers return feedback via unicast.

Our timer-based feedback mechanism works as follows:

1. The sender multicasts a **request for feedback (**I, λ, T**)** to the R receivers. I is the identification for the feedback round and T the interval size.

2. Receiver i receives the **request** (I, λ, T) after d_i time units and schedules a **exponentially distributed timer** z_i in the interval $[0, T]$. The parameter for the truncated exponential distribution is λ. When the timer z_i expires, receiver i:

   - sends the feedback message FBM(I, z_i) back to the sender if no other FBM(I, z_j) was received by i.
   - suppresses its feedback, if a FBM(I, z_j) of some other receiver j was received before (see figure 4.1 for an illustration of the suppression of i's feedback); this requires that j sends its feedback earlier than i and that the delay $d_{i,j}$ between receiver i and receiver j is such that:
   $$d_i + z_i > d_j + z_j + d_{i,j}$$

3. On the receipt of the FBMs, the sender computes an estimate $\hat{R}$ for the number of receivers, using the knowledge about the timer settings of all receivers i that returned feedback: z_i, λ, T, see section 4.7.

4. The sender computes T and λ for the next **request for feedback** based on $\hat{R}$ and its requirement for the feedback latency and the mean number of FBMs it wants to receive.

The SRM protocol [15] uses a similar mechanism for the sending of NAKs, but has two differences: First, SRM uses a **uniformly distributed timer choice** z_i from an interval that depends on the sender-receiver delay d_i. Second, SRM prevents loss of FBMs by scheduling a second request via an exponential back-off in a larger interval in the future.

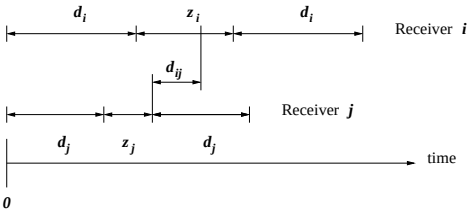

Figure 4.1: The timing for the feedback and the suppression of receiver i's FBM.

In the following, we analyze the expected number $E[X]$ of FBMs returned to the sender from R receivers and the expected feedback latency $E[M]$ due to timers, when FBMs are not subject to loss. In section 4.4 we investigate the performance under loss of FBMs. First, we introduce the following random variables:

D_i	- one-way delay between the sender and receiver i. The delay paths are symmetric and D_i expresses also the one-way delay between receiver i and the sender.
Z_i	- time receiver i delays its feedback.
$V_i = D_i + Z_i$	- the time between the sending of the request for feedback and the time the timer expires at i.
$D_{i,j}$	- one–way delay between receiver i and receiver j. The delay paths are symmetric and $D_{i,j} = D_{j,i}$.
$W_{i,j} = V_j + D_{i,j}$	- time between the sending of the request for feedback and the reception of j's feedback at i.
X_i	- Bernoulli r.v., describes the number of FBMs from receiver i, either 0, or 1.
$X = \sum_{i=1}^{R} X_i$	- total number of FBMs received at the sender from the group of receivers.

The densities $f_{D_i}(d_i)$ and $f_{D_{i,j}}(d_{i,j})$ describe the delay d_i of receiver i to the sender and the delay $d_{i,j}$ between two receivers i, j. Different timer choices and timer choices dependent on the source-receiver delay d_i can be compared in their performance when the density for the timer choice is kept general:

$$f_{Z_i|D_i}(z_i|d_i) \tag{4.1}$$

Then, the density of $V_i = D_i + Z_i$ can be calculated by a transform changing variables [57, ch. 6.3], resulting in:

$$f_{V_i}(v_i) = \int_{-\infty}^{\infty} f_{D_i}(s_i) \cdot f_{Z_i|D_i}(v_i - s_i|s_i) ds_i \tag{4.2}$$

The same way the density of $W_{i,j} = D_{i,j} + V_i$ can be derived. Since $D_{i,j}$ and V_i are independent the joint density is given by:

$$f_{D_{i,j},V_i}(d_{i,j}, v_i) = f_{D_{i,j}}(d_{i,j}) \cdot f_{V_i}(v_i)$$

Such that the density of $W_{i,j}$ using the transform in [57, ch. 6.3] is given by:

$$f_{W_{i,j}}(w_{i,j}) = \int_{-\infty}^{\infty} f_{D_{i,j},V_i}(s_{i,j}, w_{i,j} - s_{i,j}) ds_{i,j} \tag{4.3}$$

We assume delays D_i, and $D_{i,j}$ to be independent among receivers. When only the first timer setting is considered, the Bernoulli random variable X_i describes, whether the FBM from receiver i is sent ($X_i = 1$) or not ($X_i = 0$). Receiver i sends feedback only when no other receiver j suppresses the feedback of i. The probability for receiver i sending feedback is:

$$P(X_i = 1) = \int_0^{\infty} f_{V_i}(v_i) \prod_{j=1, i \neq j}^{R} (1 - F_{W_{i,j}}(v_i))\, dv_i \tag{4.4}$$

The analysis of the timer settings given above is valid for arbitrary delay distributions of D_i and $D_{i,j}$.

For a better understanding of the timer mechanism and the feedback suppression we will first consider the case where the *delays are homogeneous*: All receivers $i = 1, \ldots, R$ have the same

delay $d_i = c$ from the sender and the same delay $d_{i,j} = c$ to any other receiver j:

$$f_{D_i}(d_i) = \delta(d_i - c) \qquad f_{D_{i,j}}(d_{i,j}) = \delta(d_{i,j} - c) \tag{4.5}$$

In section 4.4.2 we analyze the timer mechanism for heterogeneous delays.

We consider the case where *all* receivers $i = 1, \ldots R$ choose a timer out of an interval $[0, T]$ – independent of the delay d_i between sender and receiver:

$$f_{Z_i|D_i}(z_i|d_i) = f_{Z_i}(z_i) \qquad , z_i \in [0, T] \tag{4.6}$$

We are especially interested in the minimal timer, which is the one expiring first. Let $M = \min_{i=1}^{R}\{Z_i\}$ be the random variable describing the minimal timer. Since the Z_i are identically and independently distributed, the distribution of the minimal timer is given by [58, ch 2]:

$$F_M(m) = P(M \leq m) \quad = \quad 1 - (1 - F_{Z_i}(m))^R$$

Our performance measures for evaluating the timer mechanisms are:

- **The expected feedback latency** $E[M]$ **due to the timer mechanism**, given by the minimal timer:

$$E[M] = \int_0^T (1 - F_M(m))\, dm \tag{4.7}$$

- **The expected number** $E[X]$ **of FBMs** at the sender given as:

$$E[X] = \sum_{i=1}^{R} E(X_i) = RP(X_i = 1) \tag{4.8}$$

Using these two performance measures, three different distributions for the timer choice are examined in terms of feedback suppression and feedback latency: The **uniform** distribution, the **beta distribution**, and the **exponential** distribution.

4.2.1 Uniformly Distributed Timers

A uniformly distributed timer choice out of the interval $[0, T]$ for receiver i is given by the density:

$$f_{Z_i}(z_i) = \begin{cases} \frac{1}{T} & , 0 \leq z_i \leq T \\ 0 & , otherwise \end{cases} \tag{4.9}$$

The expected number $E[X]$ of FBMs is:

$$E[X] = \begin{cases} R & , c \geq T > 0 \\ 1 + \frac{c}{T}R - \left(\frac{c}{T}\right)^R & , 0 < c < T \end{cases} \tag{4.10}$$

The expected feedback latency $E[M]$ due to the uniform distributed timer choice is:

$$E[M] = \frac{T}{R+1} \tag{4.11}$$

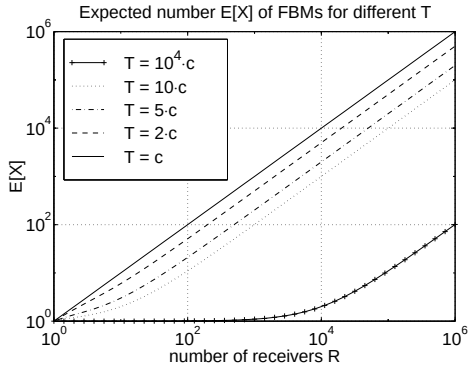

Figure 4.2: Expected number $E[X]$ of FBMs for **uniform distributed timer** choice from intervals of size $T = c, 2c, 5c, 10c, 10^4c$ for R receivers.

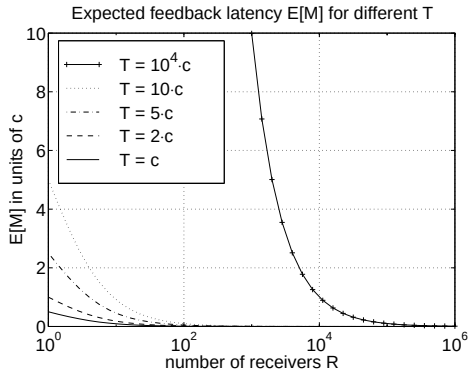

Figure 4.3: Expected feedback latency $E[M]$ for **uniform distributed timer** choice from intervals of size $T = c, 2c, 5c, 10c, 10^4c$ for R receivers.

Let the interval size T be a multiple of the delay c between receivers. For a large number R of receivers, the expected number of FBMs is $E[X] \approx \frac{c}{T}R$ and thus increases linearly with the number of receivers, see figure 4.2. The feedback latency Eq. (4.11) on the other hand decreases with R, see figure 4.3. As already reported in [15], this means that there exists a tradeoff between suppression and latency. The approximation $\frac{c}{T}R$ for $E[X]$ and the feedback latency Eq.(4.11) show the occurrence of a reasonable tradeoff between the two considerations around $T = Rc$.

Figure 4.4 illustrates how suppression works: All receivers set independently their timer in the interval $[0, T]$. All k receivers that set their timer in the interval $[m, m + c]$ will send feedback. The other $R - k$ receivers with timers $z_i > m + c$ will suppress their feedback sending, since the FBM of the receiver with the minimum timer m reaches them before their timer expires.

For a uniform timer choice, the *only* way to adapt the feedback mechanism to the number R of receivers is to change the interval size T, which makes the scheme's performance dependent on the accuracy of the receiver estimate:

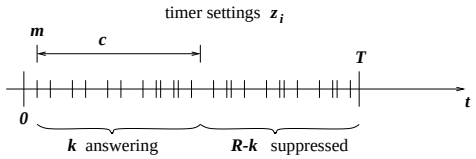

Figure 4.4: Timer Setting.

- If the number R of receivers is overestimated, the interval size T will be chosen too large and a high feedback latency will be encountered.
- If the number R of receivers is underestimated, the small interval size T will lead to a feedback implosion.

An alternative to the *uniform* distributed timer choice and to the intricacies arising from the need to carefully choose the interval size T is to change the shape of the distribution. Fixing the interval size gives a bound on the feedback delay. In order to also achieve a low number of FBMs, the minimal timer needs to be separated as far as possible from the mass of the timer settings. Therefore, the following properties are desirable for the density f_{Z_i} determining the timer choice:

- The minimal timer is separated from other timers by enabling a few timers to be set in a broad range and by grouping most timer settings in a small range.
- Feedback suppression is not sensitive to errors in the receiver estimate.

We investigate two other distributions f_{Z_i} for the timer choice: the **beta distribution** and the **exponential distribution**. Both have parameters that allow us to change the distribution.

4.2.2 Beta Distributed Timers

The beta distribution [59] has two parameters a and b. For parameters $b = 1, a \geq 1$ is a beta distributed timer choice on the interval $[0, T]$ given by the density:

$$f_{Z_i}(z_i) = \begin{cases} \frac{a}{T}\left(\frac{z_i}{T}\right)^{a-1} & 0 \leq z_i \leq T, \\ 0, & otherwise \end{cases} \tag{4.12}$$

For $a = 1$ the beta distribution equals the uniform distribution. The weight of the density shifts towards T with an increasing a and results in a dense timer setting at high values.

The expected number $E[X]$ of FBMs for a beta distributed timer choice is:

$$\begin{aligned} E[X] &= R && , c \geq T > 0 \\ E[X] &= R\left(\frac{c}{T}\right)^a && , 0 < c < T \\ &+ Ra\int_{c/T}^{1} x^{a-1}\left(1-\left(x-\frac{c}{T}\right)^a\right)^{R-1} dx \end{aligned} \tag{4.13}$$

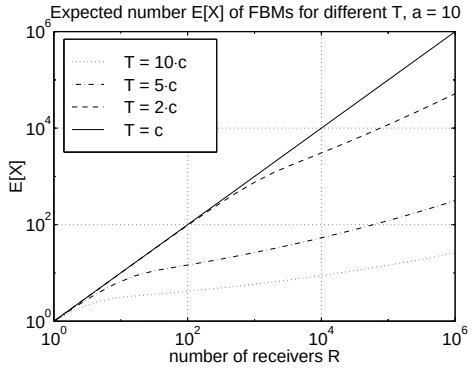

Figure 4.5: Expected number $E[X]$ of FBMs for **beta distributed timer** with parameter $a = 10$ from intervals of size $T = c, 2c, 5c, 10c$ for R receivers.

The feedback latency of Eq. (4.7) is given as:

$$E[M] = T \int_0^1 (1 - m^a)^R \, dm \tag{4.14}$$

Figure 4.5 shows the suppression performance of the beta distribution with parameter $a = 10$ for different interval sizes $T = c, 2c, 5c, 10c$. First, we observe that suppression is achieved by beta distributed timers for a wide range of numbers of receivers R. Second, a moderate interval size $T = 10c$ is sufficient to keep the expected number of FBMs $E[X] < 15$ for up to 10^5 receivers. As a consequence, feedback suppression with beta distributed timer choice is, compared with uniform distributed timers, less sensitive to an error in the estimate of the number of receivers. Also the feedback latency of beta distributed timers, shown in figure 4.6, is relatively insensitive to an error in the estimate of R: For $T = 10c$, the feedback latency varies only by $4c$ for the range from 100 to 10^6 receivers.

As in the case of uniformly distributed timers, a tradeoff exists between the number $E[X]$ of FBMs and the feedback latency $E[M]$: the price to pay for good feedback suppression is an increase of the feedback latency.

Next, we study the performance of different beta distributions by varying parameter a. Figure 4.7 shows the impact of parameter a on suppression for $R = 10^4$ receivers, the corresponding feedback latency is shown in figure 4.8. We observe from figure 4.7 that the expected number $E[X]$ of FBMs is convex in a with a minimum at some a_o. For $a > a_o$ the number of FBMs is increasing with a, since the timer settings are forced on a narrow range close to T. The feedback latency $E[M]$ indicates that the minimal timer m also moves towards T with an increasing $a > a_o$. As a result, the timer settings of an increasing number of receivers fall in the interval $[m, m + c]$ and the number $E[X]$ of FBMs increases.

For $a < a_o$ the minimal timer is close to 0 and the other timers are not well separated from the minimal timer, resulting in feedback implosion.

We further observe from figure 4.7 that the minimal $E[X]$ at a_o does not depend on the interval size T, when T is large enough. Therefore optimal suppression is achieved by minimizing $E[X]$ for a given number R of receivers, not taking the interval size T into account. Once a_o is determined

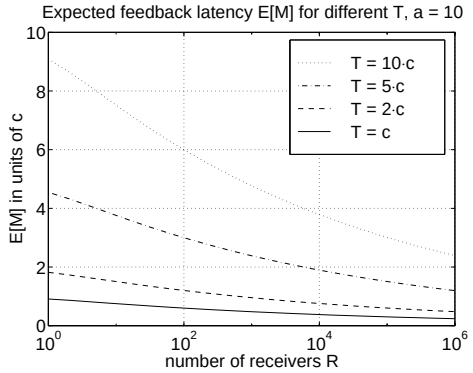

Figure 4.6: Expected feedback latency $E[M]$ for **beta distributed timer** with parameter $a = 10$ from intervals of size $T = c, 2c, 5c, 10c$ for R receivers.

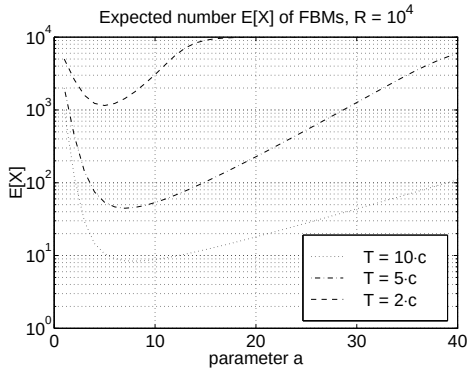

Figure 4.7: Expected number $E[X]$ of FBMs, dependence on parameter a for intervals of size $T = 2c, 5c, 10c$ for $R = 10^4$ receivers.

for optimal suppression, the interval size T can be used to trade-off feedback latency (Eq. (4.14)) against suppression (Eq. (4.13)).

Now, the remaining question is if better suppression is achieved by beta distributed timers or by uniform distributed timers, when the feedback latency is the same in both cases. This question will be answered in section 4.3. We now investigate the exponential distribution.

4.2.3 Exponentially Distributed Timers

The exponential distribution has one parameter λ and is defi ned from $-\infty$ to ∞. A truncated exponentially distributed timer choice in the interval $[0, T]$ is given by the density:

$$f_{Z_i}(z_i) = \begin{cases} \dfrac{1}{e^{\lambda}-1} \cdot \dfrac{\lambda}{T} e^{\frac{\lambda}{T} z_i} & , 0 \le z_i \le T \\ 0, & , otherwise \end{cases} \tag{4.15}$$

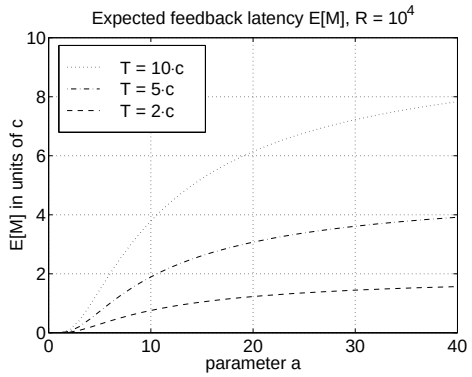

Figure 4.8: Expected feedback latency $E[M]$, dependent on parameter a from intervals of size $T = 2c, 5c, 10c$ for $R = 10^4$ receivers.

As with the beta distribution, the weight of the density shifts towards T with an increasing λ and results in a dense timer setting at high values. The expected number $E[X]$ of FBMs is:

$$\begin{aligned} E[X] &= R && , c \geq T > 0 \\ E[X] &= R\frac{e^{\lambda\frac{c}{T}} - 1}{e^{\lambda} - 1} && , 0 < c < T \\ &- e^{\lambda\frac{c}{T}}\left(\left(\frac{1 - e^{-\lambda\frac{c}{T}}}{1 - e^{-\lambda}}\right)^{R} - 1\right) \end{aligned} \tag{4.16}$$

The feedback latency is:

$$E[M] = T\int_0^1 \left(1 - \frac{e^{\lambda m} - 1}{e^{\lambda} - 1}\right)^{R} dm \tag{4.17}$$

Figure 4.9 shows the suppression performance of an exponentially distributed timer choice with parameter $\lambda = 10$. We observe a *constant* suppression performance for a wide range of number of receivers. For an interval size $T = 10c$ suppression results in an expected number of FBMs $E[X] < 3.5$ for up to 10^4 receivers. Therefore, exponentially distributed timers outperform uniform and beta distributed timers: their suppression performance is less sensitive to a poor estimate of R. This can be seen by comparing fi gure 4.2 and fi gure 4.5.

For more than 10^4 receivers, $\lambda = 10$ is too small to separate the minimal timer from all other timers. The feedback latency shown in fi gure 4.10 goes to zero and an increasing number of receivers fall in the interval $[m, m + c]$, resulting in an increasing number of FBMs, as indicated in fi gure 4.9.

For the uniform and the beta distribution we observed a trade-off between suppression and feedback latency with the interval size T. This trade-off exists also for exponentially distributed timers, as shown in fi gures 4.9 and 4.10.

The impact of parameter λ on suppression is shown in fi gure 4.11. As for beta distributed timers, $E[X]$ is again a convex function with a minimum at some λ_o. We observe that the minimal number of FBMs with exponentially distributed timers is lower than the minimal number of FBMs with beta distributed timers for the same interval size T. This is seen by comparing fi gure 4.11 and fi gure 4.7.

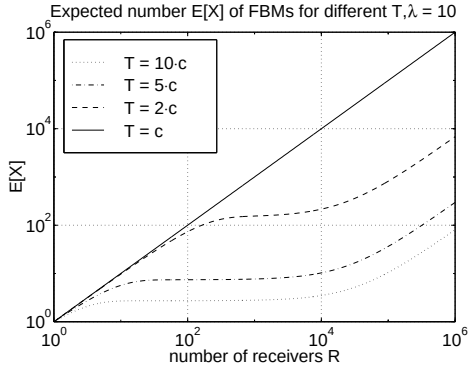

Figure 4.9: Expected number $E[X]$ of FBMs for **exponentially distributed timer** choice with parameter $\lambda = 10$ from intervals of size $T = c, 2c, 5c, 10c$ for R receivers.

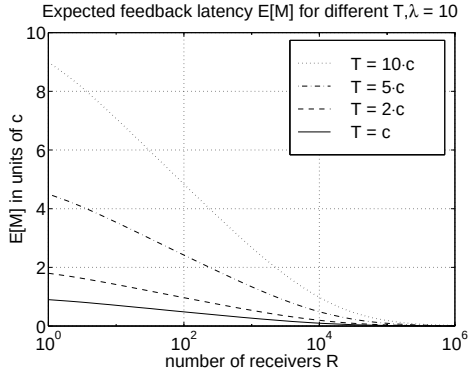

Figure 4.10: Expected feedback latency $E[M]$ for **exponentially distributed timer** choice with parameter $\lambda = 10$ from intervals of size $T = c, 2c, 5c, 10c$ for R receivers.

As with beta distributed timers, the minimal $E[X]$ is nearly independent of the interval size T, if T is large enough (see figure 4.11). The feedback latency dependency on λ, shown in figure 4.12 exhibits the same behavior: For different interval sizes, T, the feedback latency converges to 0 around the same λ. Therefore, λ_o for optimal suppression can be determined with the number of receivers, regardless of the interval size T. In section 4.5 the choice of the parameters λ and T is further investigated.

We can draw the following conclusions regarding feedback suppression dependency on the distribution:

- It is possible to avoid feedback implosion with probabilistic timers by a parametric distribution for the timer choice, while keeping the interval size T small. As a consequence, the feedback latency is small.
- Beta and exponential distribution are less sensitive to poor estimates of the number of receivers than is the uniform distribution:

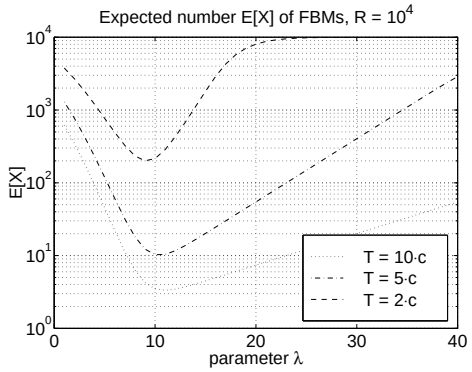

Figure 4.11: Expected number $E[X]$ of FBMs for **exponentially distributed timer** choice, dependent on parameter λ from intervals of size $T = 2c, 5c, 10c$ for $R = 10^4$ receivers.

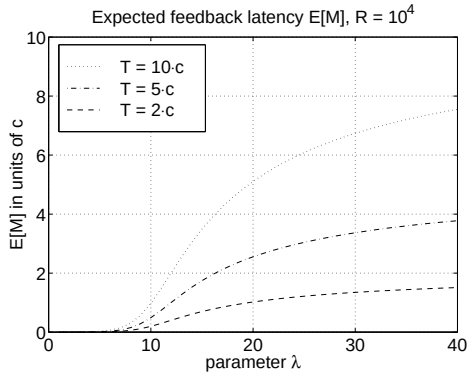

Figure 4.12: Expected feedback latency $E[M]$, dependent on parameter λ from intervals of size $T = 2c, 5c, 10c$ for $R = 10^4$ receivers.

Dynamic changes in the number of receivers by orders of magnitude do not lead to feedback implosion and have only a minor effect on feedback latency with beta and exponential distributions.

- The parameter of the beta and exponential distribution can be adjusted for a desired suppression behavior in a tradeoff with feedback latency.

- Exponentially distributed timers outperform uniform and beta distributed timers for feedback suppression.

In the next section we evaluate the three timer schemes on their performance in the context of reliable multicast feedback and will take a close look on the trade-off between latency and suppression of the three distribution functions.

4.3 Reliable Multicast Feedback

In reliable multicast communication, negative acknowledgments (NAK) are shown to achieve higher throughput performance than positive acknowledgments (ACK) [16], if retransmissions are multicast.

The subgroup of receivers that are potential NAK senders depends on the loss of data packets. The subgroup consists of all receivers that detect a loss and subsequently want to send a NAK. Without a priori knowledge of loss, the number R_l of receivers in this subgroup is unknown and may vary from 0 to R. Feedback implosion must be avoided for the *worst case* where *all* R receivers want to send a NAK. Loss measurements [30] on the Internet have shown that this worst case is not unusual.

Reliable multicast is not the only scenario where feedback should be solicited fast from a subgroup of unknown size. Other scenarios include:

- A server selection process. From a large number R of servers only those being idle should respond to a request for a task assignment.
- Multicast congestion control. From R receivers, only the R_l receivers that can not keep up with the sending rate should respond.
- Access Control. A large number R of stations are connected to a medium that is limited in access. A monitor controls the access to the medium and polls all R stations for the interest in access. Only the subgroup of R_l stations wishing to access the medium responds.

We focus on reliable multicast feedback. Let R_l be a fixed number of receivers out of all R receivers that lost data. In the following we evaluate feedback latency and choose R_l to be 1% of all R receivers, corresponding to a packet loss probability of $p = 10^{-2}$ and an average number of pR potential NAK senders out of R receivers.

We examine the timer distributions for:

- *NAK implosion* in the worst case: All R receivers are potential NAK senders.
- *NAK latency* in the average case: R_l receivers are potential NAK senders.

For each distribution, we evaluate the tradeoff between the expected number $E[X]$ of NAKs in the *worst case* where R receivers want to send a NAK and the expected feedback latency $E[M_p]$ in the *average case* where only R_l receivers want to send a NAK.

For both cases, the same interval size T is used. For the uniform distribution, $(E[M_p], E[X])$ is uniquely determined by T. The exponential and beta distribution have another parameter λ or a. This parameter is adjusted to the worst case, where all R receivers are willing to send a NAK: $E[X]$ is minimized for a group of R receivers and the corresponding λ_o or a_o is used to evaluate the tradeoff in T.

The **expected NAK latency** $E[M_p]$ is the feedback latency in the average case. It is obtained by substituting R by R_l in $E[M]$. The expected NAK latency $E[M_p]$ is higher than $E[M]$, since the feedback latency increases with a decreasing number of receivers. The expected number $E[X]$ of NAKs is given as before.

Figure 4.13 shows the expected NAK latency $E[M_p]$ versus the expected number $E[X]$ of NAKs for $R = 10^2$ receivers. This shows that on the average just one receiver will see a loss and send

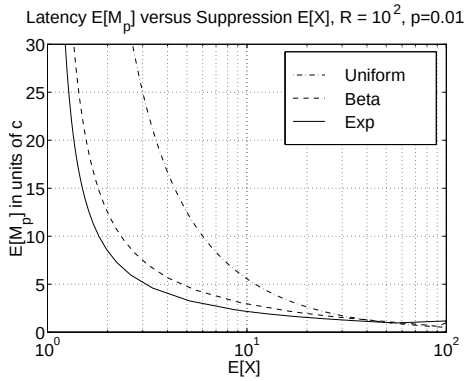

Figure 4.13: NAK latency $E[M_p]$ for optimal implosion avoidance with $R = 10^2$ receivers.

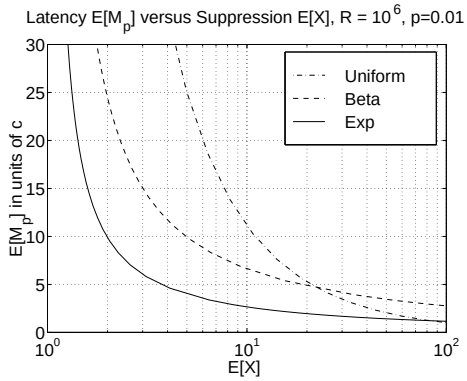

Figure 4.14: NAK latency $E[M_p]$ for optimal implosion avoidance with $R = 10^6$ receivers.

a NAK. The exponential distribution outperforms the other two distributions for up to $E[X] < 30$ NAKs in the worst case: For the same expected number $E[X]$ of NAKs in the worst case the single NAK of the average case is solicited faster with the exponential distribution than with the other distributions. For a larger group of $R = 10^6$ receivers, the benefit of using the exponential distribution is even higher, compare figure 4.14.

Figure 4.14 shows that it is possible to adjust the exponential distribution for $R = 10^6$ receivers such that in the worst case an average of 4 NAKs are returned and in the average case, the first NAK is delayed by only 5 one-way delays c.

We adjusted the three timer distributions for the same performance in feedback suppression for the case where all R receivers want to send feedback and examined the feedback latency for the case where only a subgroup of $R_l < R$ receivers want to send feedback.

Exponentially distributed timers result in faster feedback from the subgroup than with the other two timer distributions.

Due to the superior performance of exponentially distributed timers we will henceforth just consider those. In the following section we investigate the robustness of exponentially distributed timer

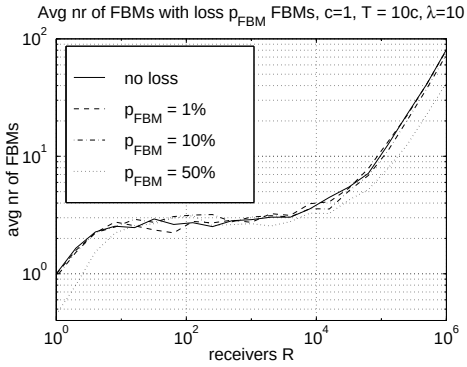

Figure 4.15: Average number of FBMs with loss p_{FBM}, $\lambda = 10$, $T = 10c$.

feedback with respect to loss and heterogeneous network delays.

4.4 Robustness of Exponentially Distributed Timers

4.4.1 Impact of Loss of FBMs

A lost FBM will not suppress sending of FBMs by other receivers. While one might expect that loss of FBMs will result in feedback implosion, we show in the following that this is not the case.

We consider the worst case, where a FBM is lost directly at the feedback sender and is therefore not received by all the other receivers. We simulated 100 feedback rounds and used parameters $\lambda = 10$ and $T = 10c$ in order to achieve simulation results that correspond to the former analytical results (see figure 4.9). FBMs were lost with different probabilities $p_{FBM} = 1\%, 10\%, 50\%$ and compared to the case of loss-free conditions. Figure 4.15 shows that the suppression performance of the timer mechanism is not sensitive to loss of FBMs for loss rates up to $p_{FBM} = 10\%$. We experienced a similar robustness also for the average feedback delay. For the very high loss rate of $p_{FBM} = 50\%$, the average number of FBMs is decreased compared to loss free conditions and the average feedback latency is slightly increased. The reason for this behaviour is twofold. First, the FBM due to the minimal timer m is lost with a probability of only p_{FBM}. Second, if the FBM due to the minimal timer is lost, the FBM of the next smallest timer $m' > m$ where the FBM is not lost jumps in and performs suppression. The number of timers expiring in $[m', m' + c]$ is higher than in $[m, m + c]$ due to the exponential distribution. However, feedback implosion does not happen since these unsuppressed FBMs themselves are subject to loss.

We can conclude that feedback suppression using exponentially distributed timers is very robust with respect to the loss of FBMs.

4.4.2 Impact of Heterogeneous Delays

In a real network, receivers have different delays to the sender and different delays between each other. In order to understand the influence of heterogeneous delays on the timer mechanism, we examine the following two cases:

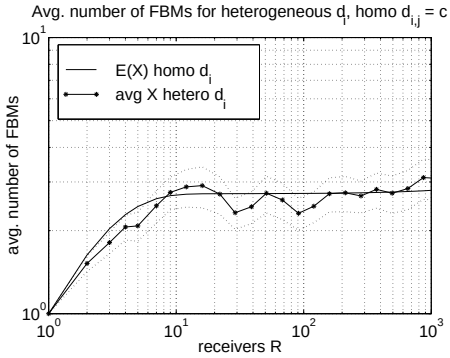

Figure 4.16: Expected number $E[X]$ of FBMs for heterogeneous sender-receiver delays $d_i \in (0, 2c)$, $d_{i,j} = c$, interval size $T = 10c$, $\lambda = 10$.

- Heterogeneous sender-receiver delays d_i, but homogeneous delays $d_{i,j} = c$ between receivers.
- Homogeneous sender-receiver delays $d_i = c$, but heterogeneous delays $d_{i,j}$ between receivers.

Both cases are compared to the case where the delays between sender and receivers and between receivers are homogeneous, i.e., $d_{i,j} = d_i = c$.

Heterogeneous delays d_i, or $d_{i,j}$ are in both cases beta distributed (see [59]) on the interval $[0, 2c]$ with parameters $a = 2$ and $b = 2$. The interval size for the timer choice is $T = 10c$. This means that the average heterogeneous delay (either $\bar{d}_i = c$, or $\bar{d}_{i,j} = c$) equals the point valued homogeneous delays c.

We simulated the FBM suppression by exponentially distributed timers with $\lambda = 10$ for this heterogeneous case for $R = 1, \ldots, 10^3$ receivers and used 95% confi dence intervals.

Heterogeneous delays to the sender

Let us consider the case where the delays between the sender and the receivers are heterogeneous and the delay between any pair of receivers i, j is homogeneous, $d_{i,j} = c$.

Figure 4.16 illustrates that FBM suppression performs better for small groups, $R < 10$, in the case of heterogeneous sender-receiver delays, than for homogeneous sender-receiver delays. This is caused by a wider spread of timer settings over $[0, 2c + T]$ due to the heterogeneous reception times d_i of the request for feedback, instead of a more narrow setting in $[c, c + T]$ with homogeneous sender-receiver delays $d_i = c$.

As the group size, R, increases, FBM suppression does not increasingly benefi t anymore from heterogeneous sender-receiver delays, since the impact of the number R of receivers on the density of the timer settings, and therefore on suppression, is higher than the small difference in the interval sizes.

Heterogeneous delays between receivers

Let us now consider a homogeneous sender-receiver delay, $d_i = c$, but heterogeneous delays $d_{i,j}$ between receivers, with $d_{i,j} \in [0, 2c]$. Therefore, the request for feedback is received at all receivers

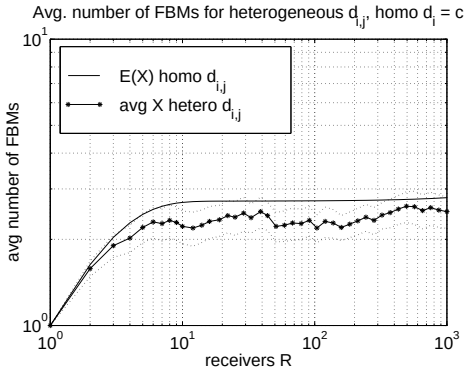

Figure 4.17: Expected number $E[X]$ of FBMs for heterogeneous inter-receiver delays $d_{i,j} \in (0, 2d)$, interval size $T = 10c, \lambda = 10$.

at the same time and all receivers set a timer in the interval $[0, T]$.

This is, for instance, the case for a forward channel via a satellite, where receivers are additionally connected among each other and to the sender via a terrestrial multicast feedback channel. The request for feedback is sent via the satellite (homogeneous $d_i = c$) while the delay between receivers via the terrestrial multicast feedback channel is heterogeneous $d_{i,j}$.

Figure 4.17 shows that for all values of R, suppression benefits from heterogeneous delays between receivers. The reason is that not only does the minimal timer FBM perform suppression, but FBMs triggered by other small timers also perform suppression. For example, the FBM due to the 2nd smallest timer may suppress the feedback sending of the 3rd smallest timer. Heterogeneous delays between receivers therefore result in the suppression of FBMs that would have been sent in the homogeneous case.

From this section we can conclude that feedback suppression by **exponentially distributed timers** is:

- **not** sensitive to loss of feedback messages.
- **not** sensitive to heterogeneous delays between sender and receiver.
- **not** sensitive to heterogeneous delays between receivers.

Instead, these cases contribute to feedback suppression with probabilistic exponential timers and so lead to even better suppression performance.

4.5 Controlling the Feedback Bandwidth

Given limited resources, the bandwidth available for feedback is limited. With the feedback mechanism from section 4.2, the feedback bandwidth is determined by the amount of feedback returned in the time between two successive feedback rounds. For a fixed FBM size of P bytes, the amount of feedback is given by $P \cdot X$, where X is the number of FBMs returned. Therefore, control over the feedback bandwidth is provided, if the number of FBMs of all receivers can be adjusted.

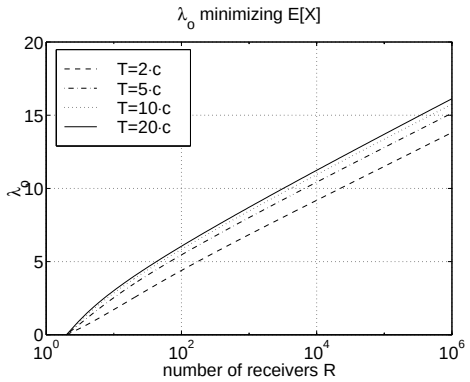

Figure 4.18: The λ_o minimizing the number $E[X]$ of FBMs dependent on R.

In the following we consider a **desired number** N of feedback messages and show how the parameters λ and T can be tuned to obtain, on average, N feedback messages with low feedback latency. To keep the sender implementation simple, we give closed-form expressions for λ and T.

First, assume that the number R of receivers is known. In section 4.2.3 it is shown that $E[X]$ is a convex function with a minimum at λ_o that is nearly independent of T. This allows us to determine a λ_o for optimal suppression - dependent only on the number of receivers: $R \mapsto \lambda_o$. Figure 4.18 shows the λ_o obtained for a given interval size, T, by minimization of $E[X]$ based on a golden section search and parabolic interpolation [60]. This is one way to adjust λ_o to the number of receivers. Another possibility is to approximate λ_o by a closed-form expression.

From figure 4.18 we observe that λ_o depends almost linearly on $\log_e(R)$. We further observe that the dependency of λ_o on the interval size T is minor. Taking this observations into account, λ_o is approximated by λ'_o for a given R:

$$\lambda'_o \sim a \cdot \log_e(R) + b$$

Parameters a and b are found by numerically fitting the polynomial $\lambda'_o(x) = a \cdot x + b$ to $\lambda_o(x)$ for $e^x = R = 10, \ldots, 10^6$ receivers.

	$T = 2c$	$T = 5c$	$T = 10c$	$T = 20c$
a	1.0383	1.0740	1.1000	1.185
b	-0.4214	0.4651	0.7326	0.8563

Table 4.1: Polynomial fitting of λ'_o to λ_o.

Table 4.1 shows the fitted parameters a and b for different interval sizes $T = 2c, 5c, 10c, 20c$. The value of a is stable between $1.0383 < a < 1.185$, while b deviates for a small interval size $T = 2c$ from the other values of b. Such small interval sizes do not allow for good suppression for most of the numbers of receivers used in the fitting process – with a, b of $T = 2c$ for 10^3 receivers already 62.3 FBMs are expected. Therefore, the deviation for small interval sizes is ignored and the

parameters are chosen as $a = 1.1$ and $b = 0.8$. The adjustment of λ_o is then given by:

$$\lambda_o = 1.1 \cdot \log_e(R) + 0.8 \tag{4.18}$$

Given λ_o, the tradeoff between the expected number of FBMs Eq. (4.16) and the feedback latency Eq. (4.17) is determined solely by the interval size T. The expected number $E[X]$ of FBMs is decreasing with T, the expected feedback latency due to timers is linearly increasing with T. Therefore, T is chosen as the smallest value for which $E[X] = N$, where N is the desired number of FBMs for R receivers. The expected number of FBMs (4.16) can be approximated, since a large number R of receivers makes the following term converge against 0 for $T > c$:

$$\lim_{R \longrightarrow \infty} \left(\frac{1 - e^{-\lambda \frac{c}{T}}}{1 - e^{-\lambda}} \right)^R = 0$$

Thus, $E[X]$ is approximated by:

$$E(X) \approx R \frac{e^{\lambda \frac{c}{T}} - 1}{e^{\lambda} - 1} + e^{\lambda \frac{c}{T}} \tag{4.19}$$

If more FBMs are desired than there are receivers ($N \geq R$), the interval size is set to $T = 0$ and every receiver sends feedback immediately. If suppression is needed ($N < R$), λ_o is used and T set such that the minimum of $E[X]$ equals the desired number, N, of FBMs. By solving Eq. (4.19) for T we obtain the expression for the adjustment of the interval size T:

$$T = \begin{cases} 0 & N \geq R \\ \dfrac{\lambda_o \cdot c}{\log_e \left(N + R \frac{1}{e^{\lambda_o} - 1} \right) - \log_e \left(1 + R \frac{1}{e^{\lambda_o} - 1} \right)} & N < R \end{cases} \tag{4.20}$$

The error incurred by the approximation via (4.18) and (4.20) is evaluated in the following. Figure 4.19 shows the expected number $E[X]$ of FBMs for $N = 3, 10, 100$ desired FBMs. We observe that the desired number N of FBMs is approached very fast. The discontinuity in the curves come from the fact that for $N \geq R$ all receivers send immediately feedback. It can be observed that the adjustment of λ and T in the given fashion works well for widely differing N.

The corresponding feedback latency shown in figure 4.20 is low and does not vary significantly with the number of receivers. Even in the case where $N = 3$ FBMs are desired from $R = 10^6$ receivers ($999,997$ suppressions) on average, the first FBM is delayed only for $2c$, which corresponds to one round trip time.

We gave closed-form expressions in Eqs. (4.18) and (4.20) for the adjustment of parameters λ and T to a desired mean number N of FBM. Parameter λ is chosen such that the number of feedback messages is minimized for a given number of receivers. Parameter T is chosen such that the desired number N of FBMs equals this minimum. Due to the tradeoff between number of FBMs and feedback latency this adjustment yields low feedback latency.

Throughout this section we assumed that the number, R, of receivers is either known exactly, or that there exists an estimate $\hat{R}$ for the number of receivers. In the following we investigate the robustness of the parameter adjustment in case of an error in the receiver estimate.

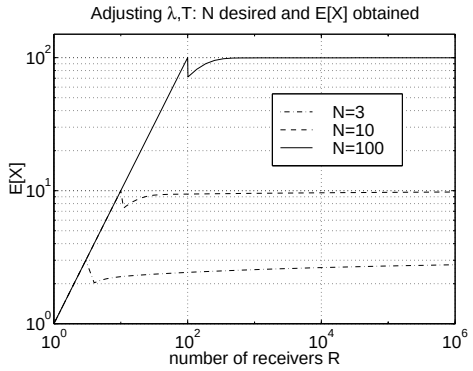

Figure 4.19: Error in adjustment of parameters λ and T to a desired mean number of FBMs $N = 3, 10, 100$.

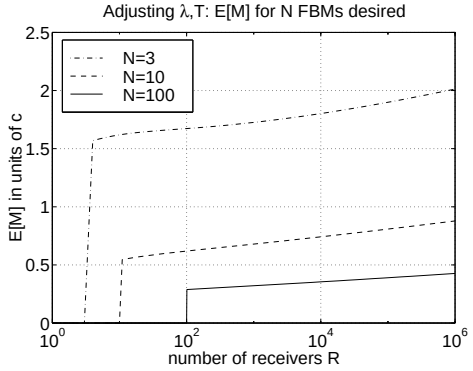

Figure 4.20: Feedback latency for the adjustment of parameters λ and T to a desired mean number of FBMs $N = 3, 10, 100$.

Erroneous Receiver Estimate

The number of receivers might change, or the estimate of the number of receivers might be erroneous. We examine the danger of feedback implosion if the actual number R of receivers is different from the estimate $\hat{R}$. Parameters λ and T are adjusted via (4.18) and (4.20) for $N = 10$ desired feedback messages and for estimates of the number of receivers $\hat{R} = 10^2, 10^3, 10^4$. From figure 4.21 we observe that the parameter adjustment results in the desired number of FBMs obtained just at the end of the flat segment of $E[X]$, right before the expected number $E[X]$ of FBMs starts slowly to increase when the actual number of receivers is higher than the estimate $\hat{R}$. First, at this point the feedback latency is low, compare figure 4.9 and figure 4.10. Second, we observe that the parameter adjustment is robust against a poor receiver estimate. If the real number of receivers is one order of magnitude higher than estimated, the number of FBMs only doubles. If the real number of receivers is one order of magnitude lower than estimated, the number of FBMs stays constant.

To assure that feedback implosion is avoided, we propose to adjust the parameters λ and T using

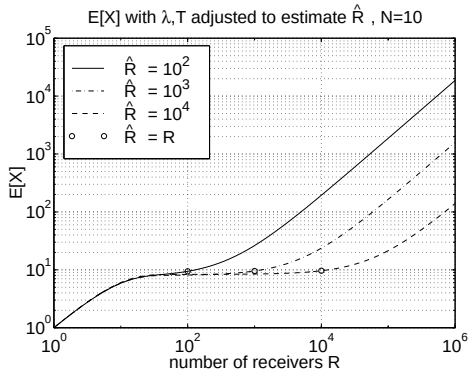

Figure 4.21: Impact of a wrong receiver estimate $\hat{R}$.

a worst case receiver estimate $\hat{R}_{max} > \hat{R}$. If the receiver group is known to be stable, $\hat{R}_{max}$ can be chosen close to $\hat{R}$ to decrease feedback latency.

4.6 Unicast Feedback

Feedback suppression as introduced in section 4.2 requires a multicast feedback channel for *every* receiver. In this section we show how the same mechanism works in the presence of unicast feedback channels from the receivers back to the sender.

Unicast feedback has several advantages:

- The state in network nodes is reduced if only sender-based multicast routing algorithms, such as DVMRP [61], are supported. In such networks a separate multicast tree is built for every multicast sender, even if the senders belong to the same group. Receivers that multicast feedback *are* senders and the state in the network is therefore proportional to the number of receivers.

- Feedback suppression is possible for satellite networks, with terrestrial unicast feedback channel.

Feedback suppression requires receivers to be aware of feedback sent by other receivers.

For unicast feedback, the missing multicast feedback channel is *emulated*. On the receipt of the first unicast feedback message, the sender multicasts information to all receivers to indicate that the feedback round is closed. On the reception of this message, receivers suppress feedback for this round.

For a multicast feedback channel a natural robustness against FBM loss exists for feedback suppression, see section 4.4, since multiple FBMs are sent, each of which able to suppress other feedback sendings. To achieve a similar robustness to FBM loss for unicast feedback, the sender must indicate to the receivers the end of the feedback round several times. Several possibilities exist:

- The sender forwards every FBM received.

- The sender indicates several times using the forward multicast channel the end of feedback round I.
- The sender starts a new feedback round $I+1$. Receivers that have pending feedback for round $J < I+1$ suppress this feedback.

The advantages of unicast feedback are offset by a larger feedback delay. This larger feedback delay, in turn, must be taken into account, when determining timer intervals. The round trip of the feedback via the sender results in a delay $d_{i,j}$ between two receivers i and j that is given by the sum of the symmetric delays d_i and d_j to the sender:

$$d_{i,j} = d_i + d_j$$

For unicast feedback and homogeneous delays $d_i = d_j = c$, the distance $d_{i,j}$ between receivers becomes $d_{i,j} = 2c$, as opposed to the case of multicast feedback, where $d_{i,j} = c$. The interval size T adjusted with Eq. (4.20) in proportion to the distance between receivers; therefore T also doubles. Since the feedback latency (Eq. (4.17)) is proportional to T, it will also double. The expected number $E[X]$ of FBMs in Eq. (4.16) will not change, since it is determined by the ratio of the distance between receivers and the interval size: c/T for the case of multicast feedback and $2c/2T = c/T$ for unicast feedback. As a consequence, the results from previous sections hold also for the case of unicast feedback, except that the expected feedback delay $E[M]$ due to timers will double.

4.7 Estimating the Number of Receivers

Up to now, we have assumed that either the number R of receivers is known exactly, or we assumed the presence of an estimate $\hat{R}$ on the number of receivers. This section shows how such an estimate can be obtained based on feedback with exponentially distributed timers.

Since the sender determines the parameters λ and T, it has knowledge about the timer distribution. Remember that every receiver sending feedback returns its timer z_i and the identifier I for the feedback round. The sender has associated with the feedback round the parameters of the distribution for the timer choice. With the density $f_Z(z)$ from Eq. (4.15) the distribution $F_Z(z)$ of a single timer z set in the interval $[0, T]$ is given by:

$$F_Z(z) = \frac{e^{\lambda \frac{z}{T}} - 1}{e^{\lambda} - 1} \tag{4.21}$$

Let X be the number of FBMs returned from the receivers, the sender obtains a sample $z_{1:R} \leq z_{2:R} \leq \cdots \leq z_{X:R}$ of X timers out of the R timers set. Assuming a constant delay c between receivers and between any receiver and the sender, this sample gives the smallest X timers set among all R receivers. Let $m = z_{1:R}$ denote the minimal timer, then m is the first order statistic [58]. Since receivers with a timer $z_i > m + c$ suppress feedback sending, the X timers returned are all set in the interval $[m, m + c]$.

Figure 4.22 illustrates how this knowledge is used to obtain an estimate $\hat{R}$ on the number of receivers:

- Having the first X timers set in $[m, m+c]$ corresponds to the probability $F_Z(m+c) - F_Z(m)$.

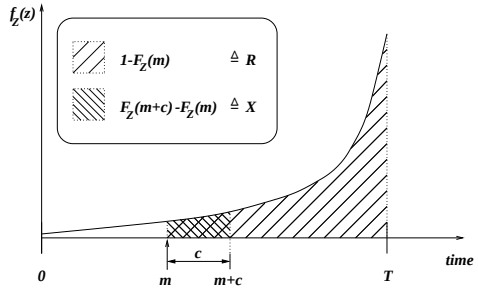

Figure 4.22: Estimation of the number of receivers.

- Having all R timers set in $[m, T]$ corresponds to the probability $1 - F_Z(m)$.

The ratio between these two cases allows to estimate the number of receivers by:

$$\hat{R}_I = X \frac{1 - F_Z(m)}{F_Z(m + c) - F_Z(m)} \tag{4.22}$$

For a single feedback round I, Eq. (4.22) gives an estimate $\hat{R}_I$. This estimate $\hat{R}_I$ as an outcome of round I is in turn used to adjust the parameters λ and T for the next feedback round $I + 1$ via expressions (4.18) and (4.20).

Due to the probabilistic timer choice some variance is encountered for the minimal timer m and the number X of returned feedback messages, even for a constant number of receivers. Since m and X vary from round to round, the receiver estimate also varies. In order to get a more stable receiver estimate for round I, an average over several receiver estimates is taken. In order to adapt the receiver estimate to changes in the receiver population an exponential weighted moving average is used:

$$\hat{R}_{I,\alpha} = \begin{cases} 1 & I = 1 \\ (1 - \alpha)\hat{R}_I + \alpha \hat{R}_{I-1,\alpha} & I > 1 \end{cases} \tag{4.23}$$

We propose to choose $\alpha = 0.8$ to achieve a fast convergence and a reasonably smooth estimate over the feedback rounds.

Note that for the parameter adjustment of λ and T, $\hat{R}_I$ is used, instead of $\hat{R}_{I,\alpha}$ to react faster to changes in the receiver population, thereby avoiding feedback implosion.

We simulated the feedback mechanism based on exponential timers from section 4.2 using the parameter adjustment from section 4.5, Eqs. (4.18) and (4.20), and the receiver estimation of Eq. (4.22). The one-way-delay is chosen to be $c = 300$ms, $N = 20$ FBMs are desired on average and $\alpha = 0.8$ was used to smooth the receiver estimate $\hat{R}_{I,\alpha}$. Without a priori knowledge of the number of receivers, feedback implosion is avoided for the first round $I = 1$ by initially adjusting parameters λ and T to a maximum number of $R_{max} = 10^6$ receivers.

After sending a **request for feedback (**I, λ, T**)**, the sender sets a timeout $T + 2c$ and collects FBMs for round I until its timeout expires, then starts the next round $I + 1$.

We evaluate the receiver estimation algorithm for a dynamic receiver population changing from $R = 100$ to $R = 10000$ receivers and back to $R = 100$ receivers. The duration of the 300 feedback rounds simulated was 7min 8.1696s. After 2min 5s (round $I = 100$) the number of receivers

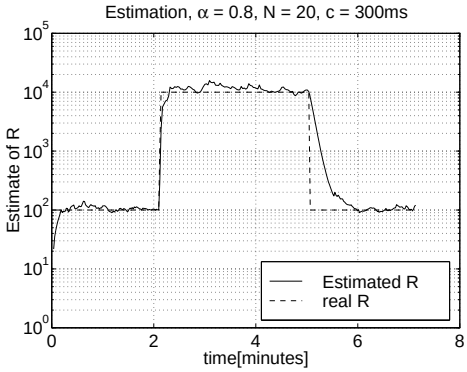

Figure 4.23: The receiver estimate $\hat{R}_{I,\alpha}$ for a dynamic population, $c = 300$ms, $N = 20$, $\alpha = 0.8$.

increased within 1.2333s up to 1000 receivers (for round $I = 101$) and after another 1.2378s up to 10000 receivers.

Figure 4.23 shows that the receiver estimate $\hat{R}_{I,\alpha}$ tightly follows the real value of R, even for sudden changes by orders of magnitude: The sudden increase of the number of receivers from 100 to 10000 receivers within less than 2.5s is captured very fast by the receiver estimate $\hat{R}_{I,\alpha}$. The convergence of the estimate after the sudden drop from 10000 back to 100 receivers is slower. This behavior is due to the weighted moving average, where high values of R_I keep their influence for a longer time than small values of R_I.

From figure 4.23 we further observe that the receiver estimate shows a relatively small deviation from the real number of receivers. If a more accurate receiver estimate is desired several solutions exist:

- The number N of desired FBMs can be increased, this is equivalent to increasing the feedback bandwidth. Increasing N always results in a more accurate receiver estimate.
- More sophisticated methods from order statistics can be applied. One possibility is to use maximum likelihood estimation of $\hat{R}$ with the joint distribution of the X returned timer values (the first X order statistics).

Parameters λ and T are adjusted every round to $N = 20$ desired FBMs on average for the dynamically changing number of receivers. Figure 4.24 shows the number of FBMs returned in each of the 300 rounds at the points in time the round is finished ($T + 2c$ after the request for feedback). While the number of FBMs returned in each round varies quite a bit (max. = 73, min. = 1) the average number of FBMs over the entire trace (dashed line) is very close to the desired number of $N = 20$ messages.

We further evaluated the required bandwidth for feedback at the sender.[1] Assuming feedback packets of size $P = 50$bytes the feedback bandwidth BW_I needed for round I is given by the ratio of the number X_I of returned FBMs in round I and the duration $T_I + 2c$ of the round:

$$BW_I = P \cdot \frac{X_I}{2c + T_I}$$

[1] In the case of multicast feedback, the bandwidth BW_I at the sender is the bandwidth consumed on *every* link of the multicast tree.

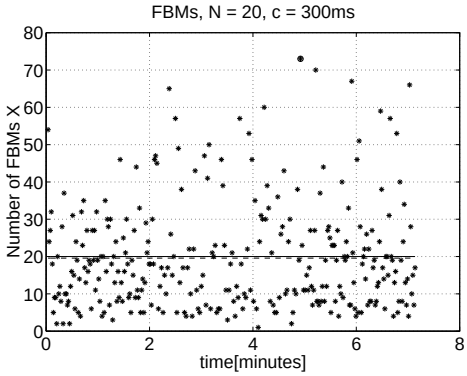

Figure 4.24: The number X of FBMs returned.

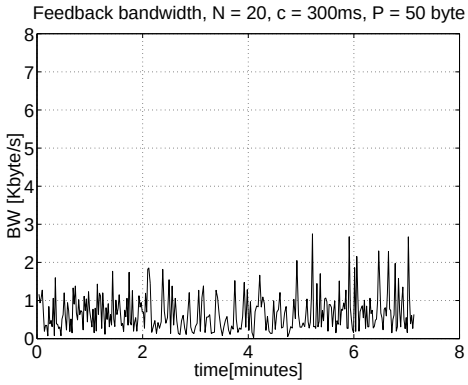

Figure 4.25: The feedback bandwidth used at the sender.

Figure 4.25 shows the bandwidth in Kbyte/s over time. The feedback bandwidth has a mean of 0.7 Kbyte/s and never exceeds 2.7 KByte/s. Furthermore, we observe that the dynamic change of the receiver population has no effect on the feedback bandwidth. The reason is that the interval size T_I increases with the increased number $\hat{R}_I$ of estimated receivers for the period when the real number of receivers is 10000.

We gave a method to estimate the size of a dynamically changing population of receivers. A low bandwidth feedback channel was also shown to be suffi cient to provide for a fast adapting receiver estimate. The feedback bandwidth was unaffected for changes by orders of magnitude in the number of receivers.

Using this method, multicast and broadcast senders, including TV and radio, have the possibility of continuously estimating the number of receivers. Required is only a unicast feedback channel (e.g. dial-up line) from the receivers back to the sender. Such an estimate of the number of receivers enables senders to:

- Determine the popularity of emitted content within seconds.

- Advertise when the number of receivers is high.

The given estimation method can also be used to determine the size of a specific subgroup of all receivers. The request for feedback must contain a discriminator that allows a receiver to determine if it belongs to the subgroup requested. Estimating the size of dynamic subgroups is also useful for multicast flow and congestion control, where the sender adapts its send rate dependent on the reception status of the receivers. Distinct subgroups for the reception status are given e.g. by ranges of experienced loss rates [62]. Receivers measure loss over some time and determine the corresponding subgroup. The sender estimates the size of the different subgroups and reacts correspondingly.

4.8 Discussion and Related Work

Ammar has defined the feedback problem as response collection via several cost functions [63]. Most research on the feedback implosion problem has been driven by reliable multicast feedback.

Two major classes of feedback mechanisms exist that provide a solution to the *feedback implosion* problem:

- *Hierarchical approaches* [48, 12, 47, 64, 65]: Are an inherent solution to the *feedback implosion* problem and ensure a limited number of FBMs by accumulation/filtering in subgroups.
- *Approaches based on MAC protocols* [66, 46, 15, 56]: The feedback problem in multicast communication is related to the problem of Medium Access Control [67]: The multicast channel constitutes the shared medium and messages sent on the multicast channel are seen by every connected group member. A token mechanism as in token ring is proposed in [66] and random timers with exponential back-off as in CSMA/CD [68] are used in XTP [46] or the SRM protocol [15, 56].

Both classes of solutions are not without disadvantages: Hierarchical approaches require the expensive setup of the hierarchy of subgroups and can not be employed in a scenario like satellite distribution with unicast backward channels. Approaches based on MAC protocols suffer from scalability problems. Tokens lead to high feedback latencies and random timers in [15, 56] are based on a uniform distribution. The analysis in [55] compares multicast feedback with random uniform timers to unicast feedback with respect to the cost in terms of number of control packets per link. The authors conclude that unicast control packets outperform multicast control packets for a small number of receivers.

SRM [15] exploits heterogeneous delays for a deterministic suppression, but needs a delay estimate $\hat{d}_i$ to the sender. This involves at least one packet sending from every receiver i, resulting for large groups of R receivers in a high amount of control traffic proportional to R. The optimal deterministic timers setting of Grossglauser [69] ensures only one NAK. However, the scheme requires the knowledge of the delay and network support for the timer setting.

Our mechanism does not suffer from any of these drawbacks, since it is a pure end-to-end mechanism. It does not rely on a full table of delay estimates to all receivers and its complexity is independent of the number of receivers. It does not need any network support except for data delivery and it does not need topological information. It can be employed in any kind of multicast capable network, also in networks where the feedback channel is only unicast.

Another end-to-end solution based on probabilistic feedback with exponential steps is the probing method of Bolot [62] that proceeds in discrete rounds. Using discrete rounds leads to very good

performance for suppression but incurs a higher feedback latency than our scheme that uses a single round.

4.9 Conclusions

We investigated probabilistic feedback for multicast groups of up to 10^6 receivers by analysis and simulation. Our main results are:

- Exponentially distributed timer settings lead to a lower feedback latency and better feedback suppression than existing schemes based on uniform distributed timer settings.
- Probabilistic feedback with exponential timers is scalable with the number of receivers and avoids feedback implosion while assuring moderate feedback latency.

Based on these results we proposed a new timer-based feedback scheme that requires very little state, does not need any network support other than data delivery, and adapts to the number of receivers:

- It avoids feedback implosion and feedback arrives fast.
- It is robust under loss of feedback messages.
- It works for heterogeneous and homogeneous delays between multicast group members and can therefore be employed on nearly any kind of network including satellite-based networks.
- It allows one to control the feedback bandwidth by adjusting the parameters dependent on the trade-off between average number of feedback messages returned and the latency for the feedback.
- It allows one to estimate the number of receivers.
- It is robust against an erroneous receiver estimate.
- It can operate on networks that only provide unicast feedback channels.

Chapter 5

Protocol Comparison

5.1 Introduction

The requirements for reliable multicast communications vary widely and several different protocol approaches have been proposed to provide reliable multicast delivery. Therefore, it cannot be expected that a single approach will be used for many different application and network scenarios. Instead, it can be expected that alternative approaches will coexist. A large number of protocols providing reliable multicast services have been presented which feature, among other differences, a large variety of error control mechanisms. Several taxonomies were presented to classify the large number of different multicast protocols (see [70, 16, 71, 72, 73]).

Multicast error recovery can be classified, depending on the participation of group members, as:

- **Centralized error recovery (CER)** allows retransmissions to be performed exclusively by the multicast source, referred to also as **Source-based recovery**.
- **Distributed error recovery (DER)** allows retransmissions potentially to be performed by all multicast members. The burden of recovery is decentralized over the whole group.

Distributed error recovery can further be sub-classified (see figure 5.1), since the multicast group may be partitioned into multiple **local**[1] **groups**. In such a case, we refer to **grouped DER**, where retransmissions are performed within a local group. The absence of local groups is described by **ungrouped DER**, where retransmissions are performed by *any member* to the *global* multicast group.

Existing protocols and classifications can be mapped to our classification scheme in agreement with their authors' classification. Further there are no conflicts with other classifications [71], [72]. RMTP [12] is a protocol based on a hierarchical structure with local groups, each with a designated receiver that performs retransmissions. RMTP is a grouped DER protocol. SRM [15] allows retransmissions potentially by all nodes and proposes extensions for local recovery. Hence, SRM is an ungrouped DER protocol in our classification. In the case of the extension it is a grouped DER protocol. In NP [19] only the multicast source can perform retransmissions, so NP can be classified as CER. MESH [74] is presented as a DER protocol to which we conform with our classification. MESH incorporates both local and global recovery.

[1] Local in the sense of neighboring in the multicast tree

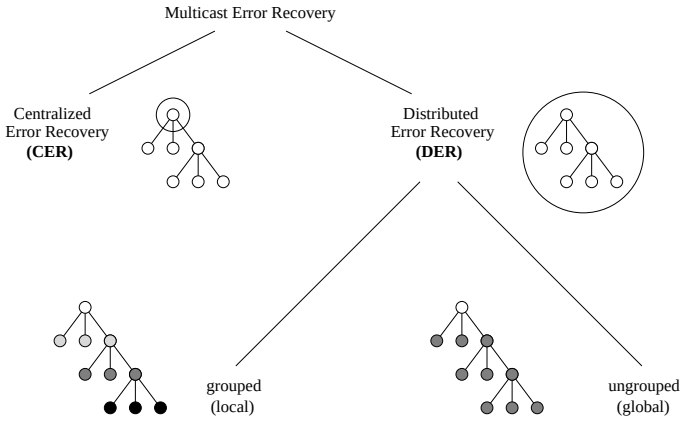

Figure 5.1: Classification of multicast error recovery techniques

Retransmission mechanisms can further be distinguished according to whether original data or parity is retransmitted for loss repair. Retransmission of parity, also referred to as **type 2 hybrid ARQ** has excellent scaling properties for large groups, as different losses at different receivers can be repaired by a *single* parity packet. It leads to a significant reduction of the number of transmissions compared to retransmission of original data as we have seen in Chapter 3. Since we consider here only retransmissions, we refer to parity retransmission also as FEC[2].

Several comparisons between generic protocols of the DER class and the CER class exist. In [71] is shown that DER protocols are superior to CER protocols concerning throughput. In [75] a grouped DER and a modified ungrouped DER protocol are compared and better performance is shown for the grouped DER protocol.

Our results from Chapter 3 now allows to reconsider CER protocols. In the following we will compare a CER protocol based on hybrid ARQ type 2 to a grouped DER protocol with respect to the performance in bandwidth consumption and completion time for a reliable transfer.

We further combine the two successful approaches to a grouped DER protocol with parity retransmissions and compare it to the others.

The rest of this chapter proceeds follows. Section 5.2 presents our model for the comparison and describes the protocols. Section 5.3 gives the analysis of bandwidth consumption of the different approaches. Section 5.4 compares the different approaches with respect to different loss scenarios. Section 5.5 compares the protocols for completion time. Finally, section 5.6 presents conclusions.

5.2 Model

We are looking at $1 : R$ communication and assume the multicast routing tree to be created by some multicast routing algorithm. We consider temporally independent data packet loss due to buffer overflows in network nodes of the tree. Due to the tree structure we can assign every node a link and refer in the following also to link loss. The spatial loss correlation among receivers is given by the topology of the tree model shown in figure 5.2. The first tree level consists of one link, the **source**

[2]Usually FEC means that parity is transmitted pro-actively with the originals

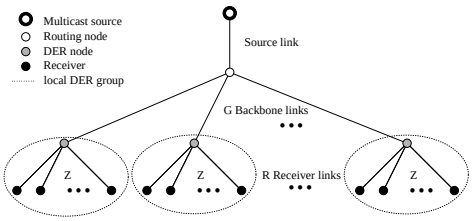

Figure 5.2: Tree model.

link, connecting the multicast source to a backbone router. Loss on the source link is experienced by all receivers (**shared loss**). In the second tree level we have G **backbone links**, each leading via Z **receiver links** to the receivers that are located at the leaves of the tree. Therefore the tree connects $R = G \cdot Z$ receivers to the source.

The tree is similar to the one in [75], which is based on loss measurements for Internet multicast [30] that showed that loss occurs mainly on the source link and on the receiver links and that backbone loss is negligible. Our tree model allows us to model such loss, by assigning no loss to backbone links.

Figure 5.2 shows the tree model for DER, where Z receivers connected to the same backbone link belong to one **local group**. Each local group constitutes of a separate multicast group and the **DER node** at the end of a backbone link can perform retransmissions.

For CER the topology is the same, but only one multicast group exists that connects all receivers to the source. Local groups do not exist and DER nodes are just internal nodes that only perform routing of multicast packets.

To show the influence of loss on the different tree levels, we will examine different loss scenarios:

- *homogeneous independent loss* only on the receiver links (last hop) with packet loss probability p.
- *heterogeneous independent loss* only on the receiver links; In each of the G local groups a fraction $f_h\%$ of the Z receivers experience high loss with probability p_h, all other receivers experience loss with probability p.
- *shared source link loss* with a homogeneous loss probability p' on the source link and all receiver links.

Let d describe the constant time it takes to send a data packet over any link. With our tree model the RTT between receiver i and the source is $d_i = 6d$, between receivers i and j of the same local group is $d_{i,j} = 2d$ and the RTT between receivers i and j of different local groups is $d_{i,j} = 4d$.

5.2.1 Protocols

Since a large number of variations is possible within the classes of CER and DER protocols, we examine a few generic protocols from each class, with characteristics that have been shown to allow the highest performance for this class, up to date. For our comparison we defi ned one CER protocol that features hybrid ARQ type 2, one DER protocol that features hybrid ARQ type 2 and one DER

protocol that features plain ARQ. We did not look at a CER protocol that features ARQ, since existing work already showed that DER/ARQ is superior to CER/ARQ [71].

For all three protocols we assume receiver-based loss detection and negative acknowledgment (NAK). Retransmissions are multicast. All protocols transmit an ADU consisting of N packets that are split into TGs of size k packets. The transmissions and retransmissions can be interleaved. Interleaving means that packets of different TGs can be transmitted intermittently. This improves the protocol throughput, since the source can use the time while it waits for feedback to transmit new packets (see section 5.3).

We assume that there is network support for hierarchical filtering of feedback messages for the DER protocols. This way the G feedback messages from the DER nodes will be filtered by the backbone router below the source, such that redundant feedback messages will not arrive at the source. We consider the processing amount for those feedback messages at the source to be negligible. However, the Z feedback messages from the group members in one DER group have to be processed by the DER nodes. In our later Latency analysis we will neglect feedback processing both for CER and for DER, since we scale the amount of feedback messages in the CER case down to the number of feedback messages that has to be processed by a DER node.

Protocol C

Protocol **C** is a CER protocol based on hybrid ARQ type 2 and feedback suppression with exponential timers that we studied in Chapter 4. Parity packets are not pre-encoded, but are coded on demand using the Reed-Solomon coder presented in [76].

Receiver-based loss detection assumes in-sequence delivery of packets, to be able to perform gap-based loss detection.

The parameters λ and T of the feedback suppression mechanism are given with Eq. (4.18) and Eq. (4.20) such that the expected number of feedback messages arriving at the source is, in the worst case, equal to the number of receivers in a local group, $E[X] = Z$, in our tree model.

The transmission of a TG of k packets is done the following way:

The multicast source:

1. Sends the k original packets of the TG; a poll for feedback is piggybacked with the last transmitted packet to indicate the end of the TG.

2. If it is indicated by feedback from the receivers that less than k packets are received by any receiver, a_{max} new parity packets are generated and retransmitted, where a_{max} is the maximum number of packets missing out of the total number of k packets. Again, a poll for feedback is piggybacked.

3. Step 2 is repeated until no feedback about missing packets is received within a certain timeout interval.

The receiver:

1. Original and parity packets of a TG are buffered.

2. If k or more packets have been received, the k originals are decoded and sent to a higher layer.

3. If less than k packets have been received and a poll for feedback for the TG is received, the receiver calculates the number of required additional parity packets. If the feedback suppression algorithm decides that the receiver sends feedback, it will multicast its feedback with the number of missing packets (NAK).

4. Step 3 is repeated until $\geq k$ packets have been received.

Generic CER Protocol (C_{noFEC}) We define a generic CER protocol with the same characteristics as protocol C but instead of parity retransmission it uses data retransmission. This protocol is merely used to show that CER protocols have a higher performance gain when employing parity retransmission than DER protocols. We will not provide any further analysis for this protocol, since it can easily be derived from the analysis of C and D1.

Protocol D1

We define D1 as grouped DER protocol that uses just ARQ. The source is a group leader for all the internal DER nodes in the tree model (figure 5.2). The internal nodes in turn are group leaders for all the receivers at the leaves. The first transmission is done to all receivers. Retransmissions are performed locally. A grouped DER scheme reduces the maximum number of feedback messages to be handled by any group leader to the number of group members. This holds in our model for the lower tree level, as well in the upper tree level, we assume an optimal delay-less feedback suppression mechanism to scale the number of feedback messages to be processed by the source to the number of receivers in the local groups in the lower tree level.

Protocol D1 works in a store-and-forward manner. All data first has to be received by all DER nodes on the first tree level. Then it will be forwarded in parallel from all DER nodes to the receivers at the leaves.[3]

The transmission of a TG of k packets is done the following way, either between source and DER nodes, or DER node and receivers:

The multicast source/ DER node:

1. Sends the k original packets of the TG to all receivers and all DER nodes in the tree (global multicast). A Poll for feedback from the DER nodes is sent piggybacked to all the DER nodes (local multicast). , again with a piggybacked Poll for feedback from the DER nodes.

2. Step 2 is repeated until no missing packets are indicated anymore by the DER nodes.

The DER node/ receiver:

1. Original packets of a TG are buffered.

2. On the detection of a loss and reception of a Poll for feedback a NAK is sent.

3. Step 2 is repeated until the TG is fully received. In the case of a DER node, a Poll for feedback is now sent to the receivers.

[3] For delay considerations of reliable delivery to all receivers this is a worst case for distributed recovery, since it is assumed that maximum delays on both tree levels occur on one path.

Protocol D2

Protocol D2 is a grouped DER protocol using hybrid ARQ type 2. The groups are set up the same way as in D1. Protocol D2 transmits a TG of k packets in the same store-and-forward manner as Protocol D1. In both steps parities, rather than missing data packets, are retransmitted.

5.3 Bandwidth Analysis

We consider bandwidth in terms of a cost B for a multicast packet[4] on an average link in the multicast tree [55]. The cost of a multicast packet in a multicast group i is the product of the number M_i of transmissions per packet (original and retransmissions) and the number H_i of links that connect the group sender to the group's receivers. Over all local groups i and $H = R + R/Z + 1$ links in total, our bandwidth measure, the **average cost of a multicast packet per link** is:

$$E[B] = \frac{1}{H} \sum_i E[M_i] \cdot H_i \tag{5.1}$$

To show the relative bandwidth savings of DER protocols over CER protocols, the **relative performance** $E[B_D]/E[B_C]$ of a DER protocol D and a CER protocol C is used.

5.3.1 Protocol C

For the CER protocol C, we have only one multicast group and all transmissions are multicast over all links. Thus we get:

$$E[B] = E[M_C] \tag{5.2}$$

In the following $E[M_C]$ is derived for the different cases of loss.

For homogeneous independent loss Let L_r describe the number of additional packet transmissions required by a random receiver to receive a complete TG with integrated FEC. Let L describe the number of additional packet transmissions required to have all receivers receive the complete TG. Then the distribution of L and L_r and the expectation of L and M_C is given as in Chapter 3 by Eq. (3.6):

$$F_{L_r}(l) = \sum_{i=0}^{l} \binom{k+i-1}{k-1} p^i (1-p)^k \qquad , l = 0, \ldots \tag{5.3}$$

$$F_L(l) = F_{L_r}(l)^R \tag{5.4}$$

$$E[L] = E[L] = \sum_{l=0}^{\infty} (1 - F_L(l)) \tag{5.5}$$

$$E[M_C] = 1 + E[L]/k \tag{5.6}$$

[4] We do not consider feedback packets, due to their small size.

For heterogeneous independent loss at receivers, we assume a fraction f_h of receivers to experience a higher loss p_h and the rest of the receivers to experience the lower loss p. We can directly derive from equations 5.3 and 5.4:

$$F_L(l) = (F_{L_{rh}}(l))^{R \cdot f_h} \cdot (F_{L_r}(l))^{R \cdot (1-f_h)} \tag{5.7}$$

where $F_{L_{rh}}(l)$ is $F_{L_r}(l)$ given by equation 5.3 with p_h substituted for p. $E[M_C]$ is then given by (5.5) and (5.6).

For shared source link loss in our multicast tree, we get the value of $E[M_C]$ by simulation. The loss with probability p perceived by a receiver is due to loss on the source link and the receiver link. Let p' be the loss probability on the source link and the receiver links. Then

$$p = 1 - (1 - p')^2 \tag{5.8}$$

5.3.2 Protocol D1

The reliable transmission of a packet from the multicast source to the G DER nodes is done via $G + 1$ links with $M_{D1,G}$ transmissions. From each DER node $M_{D1,Z}$ transmissions over Z links are needed to reliably transmit a packet to the receivers of the local group. The bandwidth cost for D1 is given by:

$$E[B] = \frac{1}{H} \left(E[M_{D1,G}] \cdot (1 + G) + E[M_{D1,Z}] \cdot R \right) \tag{5.9}$$

For independent homogeneous loss each packet is transmitted once over all links and retransmissions are limited to the local group, such that we get:

$$E[B] = 1 + \frac{1}{H}(E[M_{D1,Z}] - 1)R \tag{5.10}$$

Since retransmitting originals corresponds to the retransmission of parities, if the TG size is $k = 1$ (a repetition code), equations (5.3) - (5.6) allow us to calculate $E[M_{D1,Z}] = E[M_C]$, using $k = 1$ and $R = Z$. The distribution of the number $M_{D1,Z}$ of transmissions per packet in the local groups is:

$$F_{M_{D1,Z}}(m) = (1 - p^m)^Z \tag{5.11}$$

For heterogeneous independent loss a local group consists of a fraction f_h of receivers with high loss p_h and the rest of the receivers with low loss p. The same way as above we derive:

$$F_{M_{D1,Z}}(m) = (1 - p_h^m)^{Z \cdot f_h} \cdot (1 - p^m)^{Z \cdot (1-f_h)} \tag{5.12}$$

We calculate $E[M_{D1,Z}]$ again the same way as in (5.5) and $E[B]$ from (5.10).

For shared source link loss , the loss probability p' (5.8) is the same for source link and the receiver links. Since the number of transmissions for G DER nodes behind the single lossy source link is the same as for only one DER node behind the lossy source link, we get:

$$F_{M_{D1,G}}(m) = (1 - p'^m) \tag{5.13}$$

$$F_{M_{D1,Z}}(m) = (1 - p'^m)^Z \tag{5.14}$$

We calculate $E[M_{D1,G}]$ and $E[M_{D1,Z}]$ similar to (5.5) and $E[B]$ from (5.9).

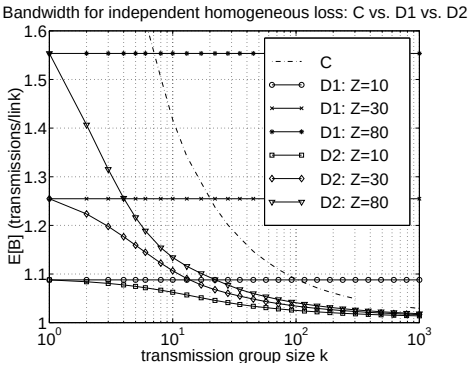

Figure 5.3: Bandwidth dependent on TG size k for independent homogeneous loss: C vs. D1 vs. D2, $R = 10^6$, $p = 0.01$.

5.3.3 Protocol D2

For D2, the bandwidth can again be calculated by separating the transmission into two independent steps. The bandwidth analysis follows the same equations as for protocol D1, except that for $M_{D2,G}$ and $M_{D2,Z}$, parity retransmission has to be considered as for protocol C. For details see [77].

5.4 Bandwidth Comparison

In the following, the three protocols D1, D2 and C are compared for the three loss scenarios. Unless stated otherwise a packet loss probability of $p = 0.01$ is used and $R = 10^6$ receivers are in the global multicast group.

First, the case of *homogeneous, independent loss* on the receiver links is considered. The performance of the protocols C and D2 with parity retransmission, depends on the TG size k, as shown in figure 5.3 for different local group sizes $Z = \{10, 30, 80\}$. The performance improves for both protocols D2 and C with an increasing TG size k. This is due to the fact that a parity packet can repair the loss of any packet out of the TG *and* that therefore a parity packet can repair the loss of different packets at different receivers. An effect that becomes more powerful with an increasing TG size k.

Figure 5.3 shows that the protocol D2 performs better than D1 for all transmission group sizes, a result that we observed also for a wide range of loss probabilities and a wide range of local group sizes Z. It can be seen that the performance of D1 and D2 improves with decreasing Z, since retransmissions are limited to within a local group.

Further is shown that even the CER protocol C achieves better performance than the DER protocol D1, if the TG size k is large enough. The reason is again parity retransmission.

In [71] the throughput performance of generic CER and grouped DER protocols is compared. From the results, it is concluded that grouped DER protocols have better scalability due to their hierarchical structure. Further it is stated that any technique employed in a CER protocol can also be employed in a local group and thus would not change anything in the relative performance. We show that this is not the case for the application of parity retransmissions and examine the relative perfor-

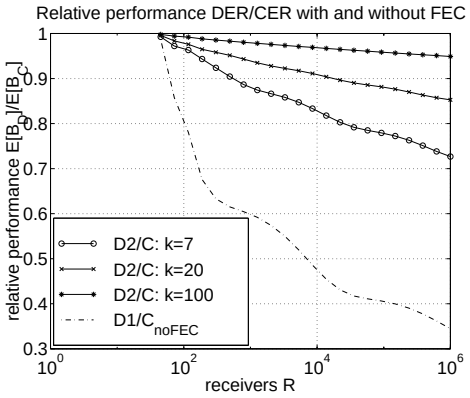

Figure 5.4: Relative performance $E[B]_{DER}/E[B]_{CER}$ for independent homogeneous loss with and without parity retransmission (FEC), $p = 0.01$, $Z = 30$.

mance DER/CER with and without parity retransmissions. The additional CER protocol C_{noFEC} is examined, which is the same as protocol C, but does not employ parity transmission.

Figure 5.4 shows that the relative bandwidth savings $E[B_{D2}]/E[B_C]$ of DER compared to CER are lower if parity retransmission is used, than without parities ($E[B_{D1}]/E[B_{C_{noFEC}}]$). This is due to the fact that protocol C performs very well due to parity retransmission; each parity packet can repair different losses at different receivers, an effect that is not exploited to the same extent in the DER case, where retransmissions are limited to a local group.

Since protocol D2 outperforms protocol D1 for all parameters, in the remainder only D2 will be considered and compared to the CER protocol C.

Next we will examine the effect of *heterogeneous independent loss* on the performance of D2 and C. 10% of all receivers experience a high loss probability of $p_h = 0.25$, while the other receivers experience loss with probability $p = 0.01$.

Figure 5.5 shows that D2 achieves higher bandwidth savings than C for heterogeneous loss, compared to homogeneous loss (see figure 5.4).

In the worst case for a TG size of $k = 7$ the performance of C relative to D2 decreases by almost 20%. This is due to the fact that high loss receivers dominate the required bandwidth, since retransmissions are multicast. D2 achieves better performance by restricting the influence of the high loss receivers to the local groups. The performance of D2 remains superior for all numbers of receivers.

Figure 5.6 compares D2 and C for the case of *shared source link loss*, where loss happens also on the first link from the source to the backbone.

It can be seen that, compared to the homogeneous case (figure 5.4, C improves relative to D2. As shown in 3.4.1, shared loss for a group of R receivers can be modeled by considering a smaller group $R_{indep} \leq R$ of independent loss receivers for protocols employing FEC. Since for protocol D2 the number of receivers in a group is small already, it profits very little from shared loss. In spite of the improvement of the performance of C, D2 remains superior over the whole range of R.

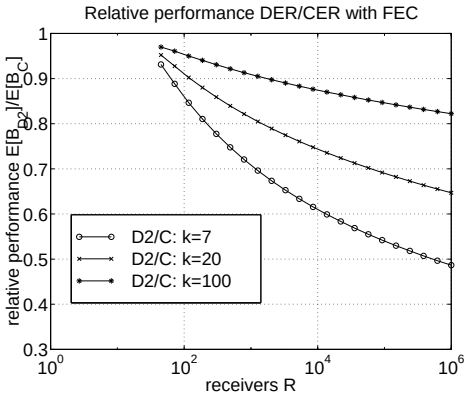

Figure 5.5: Relative performance $E[B]_{D2}/E[B]_C$ for independent heterogeneous loss with FEC, $p = 0.01$, $Z = 30$, $p_h = 0.25$, $f_h = 0.1$.

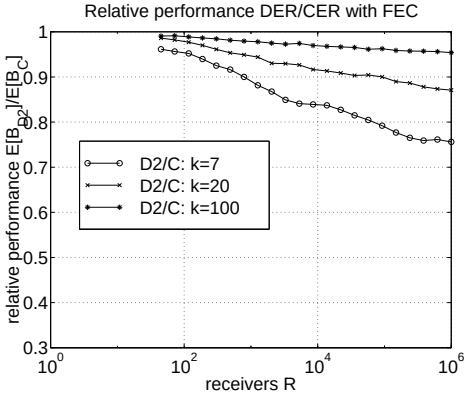

Figure 5.6: Relative performance $E[B]_{D2}/E[B]_C$ for shared source link loss with FEC, $p = 0.01$, $Z = 30$.

5.5 Latency

In the following we will give a brief overview over the latency analysis in [77]. Further, protocols C and D2 will be compared regarding the required **completion time** for the transmission of a *short ADU* [5] of length k, the TG size. The completion time is the time that is required to fully and successfully transmit the ADU from the sender to *all* receivers. To compare different sizes of k we use the average completion time per packet $E[D]/k$ in multiples of RTT as a measure. We choose the transmission group sizes $k \in \{7, 20, 100\}$, which correspond to typical sizes of pages in the WWW. The comparison is done for the three defi ned loss scenarios in which the scalability of the protocols for the number of receivers is examined.

The different contributions to the total completion time D are denoted by the following random

[5] Additional results for the transfer of large ADUs consisting of $N > 10^4$ packets were derived and can be found in [77]. The results for the comparison are largely similar to the results for the bandwidth measure

variables:

- The total gross packet **transmission delay**, denoted by D_t: this accounts for queuing delay due to flow and congestion control at the sender/DER node and is obtained from the constant packet throughput Λ,
- the **feedback delay**, denoted by D_f: this accounts for feedback suppression delay and additional round trip times through retransmissions rounds.
- the FEC **coding delay**, denoted by D_c
- the propagation delay, D_p.

such that we get:

$$E[D] = E[D_t] + E[D_f] + E[D_c] + D_p \tag{5.15}$$

For more details about the latency analysis see [77].

The numerical results for latency are ordered by loss scenario (see section 5.2). The scalability of the protocols is shown for all loss scenarios.

A constant transport layer packet size of $P = 2kB$ will be assumed. We did measurements with the FEC coder introduced in [20] on a SUN SPARC-20 workstation to calculate the FEC coding delay. The packet throughput is set to $\Lambda = 25pkts/s$. [6] We set $RTT = 0.1s = 6d$. The packet loss probability that a receiver sees is $p = 0.01$. The TG size will be chosen as $k \in \{7, 20, 100\}$. The local group size is $Z = 30$.

All results were calculated analytically according to the analysis in [77] and with additional simulations, using the topology given in section 5.2.

5.5.1 Homogeneous Independent Loss

We now look at the scalability of the protocols. Figure 5.7 shows the per-packet latency $E[D]/k$ in RTT for protocols C and D2 on the ordinate, with respect to the number of receivers R.

In figure 5.7 it can be seen that protocol D2 performs better than protocol C over the whole range of R for corresponding TG sizes k. Both protocols scale very well with the number of receivers. The performance difference between protocol C and D2 is very small for large k. The smaller performance difference for large k is due to the fact that for larger k, a larger number of different losses can be repaired with one parity packet in the CER case and thus the number of packets to be multicast is reduced. In the DER case, the number of packets to be multicast is reduced already through the partitioning of receivers into local groups and the effect of larger k is not as pronounced.

5.5.2 Heterogeneous Independent Loss

We will now examine the effect of heterogeneous loss patterns on our results from the homogeneous independent loss scenario. Figure 5.8 shows the per-packet latency $E[D]/k$ in RTT for protocols C and D2 on the ordinate, with respect to the number of receivers R.

In figure 5.8, compared to figure 5.7, it can be seen that the relative performance of protocol C to protocol D2 for corresponding TG sizes k is hardly influenced by heterogeneity of loss. In absolute

[6] A throughput of $\Lambda = 25$ packets per second has been reported by Bolot [53] for a loaded IP path between Sophia Antipolis (INRIA) and London (UCL).

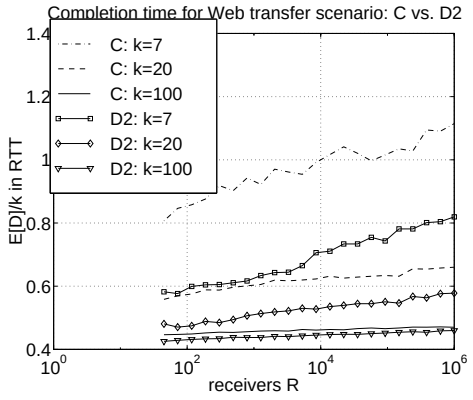

Figure 5.7: Completion time for independent homogeneous loss: C vs. D2, $Z = 30$, $p = 0.01$, $\Lambda = 25/s$, $RTT = 0.1s$, $P = 2kB$

performance, both protocols have a poorer performance due to heterogeneous loss. This is because the high loss receivers dominate the delay (the slowest receiver is decisive) through a higher number of transmissions. Protocol D2 improves slightly in relation to protocol C, since fewer multicast retransmissions are necessary in G parallel local groups with a small number of high loss receivers each (D2), than in one large group, including all high loss receivers (C).

5.5.3 Shared source Link Loss

We examine the influence of shared source link loss. Figure 5.9 shows the per-packet latency $E[D]/k$ for protocols C and D2 on the ordinate, with respect to the number of receivers R. The packet loss probability that each receiver sees is $p = 0.01$, such that the loss probability on each link is $p' = 1 - \sqrt{1-p}$.

In figure 5.9 it can be seen that protocol D2 performs better than protocol C for the whole range of R. Both protocols benefit from shared source link loss. This is due to the fact that for shared loss, even with retransmission of original packets, losses at different receivers can be recovered by retransmission of one original packet (see section 5.3). Protocol D2 profits more from shared source link loss than protocol C. The benefit through shared source link loss for both protocols is smaller for large TG sizes k than for small k.

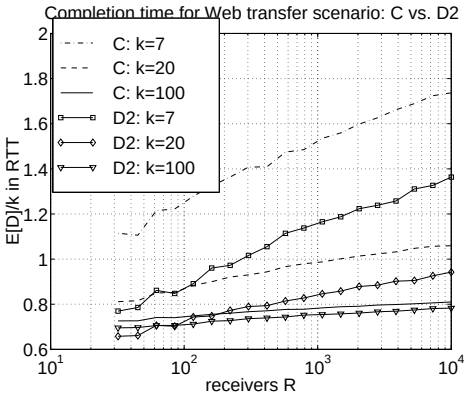

Figure 5.8: Completion time for independent heterogeneous loss: C vs. D2, $Z = 30$, $p = 0.01$, $p_h = 0.25$, $f_h = 0.1$, $\Lambda = 25/s$, $RTT = 0.1s$, $P = 2kB$

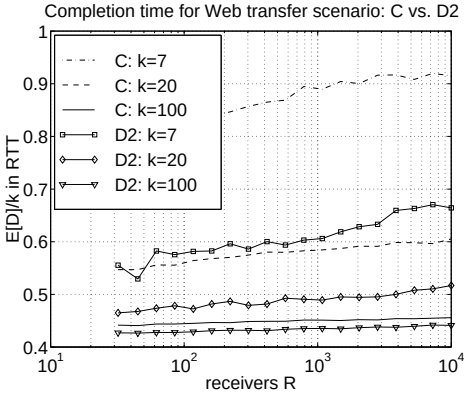

Figure 5.9: Completion time dependent on R for shared source link loss: C vs. D2, $Z = 30$, $p = 0.01$, $\Lambda = 25/s$, $RTT = 0.1s$, $P = 2kB$

5.6 Conclusion

In this chapter, we compared three generic reliable multicast protocols. Two of them (D1 and D2) have additional structure that allows them to limit retransmission to a local scope. One protocol, C, allows only retransmissions from the source. C and D2 were protocols with parity retransmissions, while for D1 originals were retransmitted. Our conclusions from the comparison with respect to completion time of a reliable transfer and the bandwidth needed are as follows:

- D2 outperforms D1 and C in terms of completion time and bandwidth.
- With heterogeneous loss, including certain high loss receivers, the performance of C decreases more than the performance of the DER protocols.
- Applying hybrid type 2 ARQ to protocols with local groups does not yield as high a performance gain, as applying it to protocols without local groups.

- For large transmission group sizes k, the performance of C comes close to the performance of D2.

Chapter 6

Conclusions

6.1 Summary of the Thesis

The thesis covers reliable multicast transmission from one sender to multiple receivers, where the number of receivers is potentially very large. The thesis focus is on the design of protocol mechanisms with respect to the scalability with the number of receivers and with respect to effi ciency. We consider protocol entities at the sender and the receivers only, where every receiver communicates exclusively with the sender or other receivers. The potential of such an end-to-end approach is the ease of deployment and the independence of the network. The proposed protocol mechanisms are not dependent on support from the network, which allows reliable multicast to be employed over any network: the Internet, satellite networks and ATM networks. Also combinations of networks are possible to achieve reliable multicast transmission, as long as the combination provides a multicast forward channel and a unicast feedback channel.

6.2 Contributions

We showed that it is possible to achieve scalable reliable multicast transmission for very large numbers of receivers up to one million via a protocol between sender and receivers only. The contributions of the thesis to the fi eld of reliable multicast are in the area of modeling and performance evaluation, in fi ndings for protocol design issues, and in new methods for retransmission and feedback for reliable multicast.

The contributions of the thesis to reliable multicast are outlined in the following.

6.2.1 Modeling

Throughout the thesis, performance is evaluated with analytical means and simulation for very large numbers of receivers. Analytical performance evaluation can be understood as a contribution to the fi eld of reliable multicast.

Scalability with the number of receivers is a key requirement for reliable multicast protocols. Evaluation of a multicast protocol via simulation, or experiment is feasible for a small number of receivers, but is infeasible in a limited amount of time for a large number of receivers. Therefore,

analytical modeling is required to evaluate performance and scalability of reliable multicast protocols. However, analytical performance evaluation has been rarely applied [31, 16, 7] in the field of reliable multicast.

We develop analytical tools that allow us to evaluate the following aspects:

- The *evaluation of the loss correlation* of receivers dependent on the multicast tree topology. We give formulae that allow one to calculate the probability of having k receivers losing the same packet, given a certain multicast tree topology and a homogeneous link loss probability.

- The *number of multicast transmissions* required to ensure that a packet is successfully received at all receivers. The number of transmissions determines the network bandwidth needed for a reliable multicast, the completion time of a reliable multicast, and the processing load at the sender and at the receivers. We gave expressions for the expected number of transmissions with integrated FEC, with layered FEC and with no FEC. In the case of no FEC we gave formulae for the expected number of transmissions in the case of burst loss.

  We discovered the step-like behavior in the expected number of transmissions with the logarithm of the number of receivers and show that the expected number of transmissions in the case of spatially correlated loss among the receivers can be obtained by considering a smaller receiver population with independent loss.

- *Feedback implosion* is expressed by the number of feedback messages returned in a certain time interval. For timer-based feedback mechanisms, we gave an analysis of the expected number of feedback messages and the expected feedback latency. The analysis allows one to compare several distributions for the timer choice on the performance in feedback suppression and feedback delay performance.

In chapter 2 about modeling we evaluated the impact of the multicast routing algorithm on reliable multicast and showed that a multicast routing algorithm that optimizes delay (SPT) results in better performance of reliable multicast than if the multicast tree is constructed by a routing algorithm that optimizes cost (KMB).

The loss correlation among receivers is dependent on the topology of the multicast tree. We compared SPT and KMB to several generic multicast trees for the cases of spatial loss correlation, the number of transmissions and the link share of the tree. We showed that the full binary tree (FBT) can serve as a good multicast tree model, since its loss characteristics are close to the loss characteristics of multicast trees that are constructed by SPT and KMB.

For a small number of receivers, we presented a simple approximation of the expected number of retransmissions to achieve reliability for all receivers. The approximation is independent of the multicast tree topology and is just the product of the number of links in the multicast tree and the homogeneous link loss probability: $E \approx q \cdot L$.

6.2.2 Loss Recovery

For a high number of receivers, reliable multicast protocols suffer from a high aggregate loss seen by the sender. As a consequence, a high number of transmissions by the sender are needed to repair loss at the receivers.

We investigated how FEC (Forward Error Correction) can be combined with ARQ (Automatic Repeat Request) to achieve scalable reliable multicast transmission. We considered the two scenarios

where FEC is introduced as a transparent layer underneath a reliable multicast layer that uses ARQ, and where FEC and ARQ are both integrated into a single layer that uses the retransmission of *parity* data to recover from the loss of original data packets.

To evaluate the performance improvements due to FEC, we considered different loss rates and different types of loss behaviors (spatially or temporally correlated loss, homogeneous or heterogeneous loss) for up to 10^6 receivers. Our results showed that introducing FEC as a transparent layer below ARQ can improve multicast transmission efficiency and scalability. However there are substantial additional improvements when FEC and ARQ are integrated.

Integrating FEC in a reliable multicast protocol does not come for free. The benefits in terms of a reduced number of transmissions are paid by coding at the sender and decoding at the receivers. Our evaluation of the impact of software coding and decoding on the throughput of the protocol showed that the sender processing rate limits the throughput of the protocol. We showed that releasing the sender from the burden of coding by pre-computation of parities or hardware support for coding results in a reliable multicast protocol that achieves higher throughput than a protocol that retransmits original data packets.

6.2.3 Feedback

The amount of feedback returned to the sender increases with the number of receivers. As the number of receivers grows, the sender runs the danger of being overwhelmed by feedback messages. We investigated the scalability of feedback in multicast communication and proposed a new way to avoid feedback implosion by probabilistic feedback based on exponentially distributed timers.

By analysis and simulation for up to one million receivers we showed that feedback implosion is avoided while feedback latency is low. The mechanism is robust against the loss of feedback messages and works well in case of homogeneous and heterogeneous delays. Our mechanism achieves lower feedback latency for the same performance in feedback suppression than existing timer-based feedback schemes. It is scalable, since the amount of state at every group member is independent of the number of receivers and the number of feedback messages is nearly constant for wide ranges of numbers of receivers. No topological information of the network is used and data delivery is the only support required from the network. It adapts to the number of receivers and leads therefore to a stable performance for implosion avoidance and feedback latency.

We showed that an estimate of the number of receivers can be given, based on exponentially distributed timer feedback. The estimation of the number of receivers adapts within seconds to a receiver population that changes in size by orders of magnitude.

6.2.4 Comparison

Finally, we quantified the difference in performance of an end-to-end reliable multicast protocol that uses parity retransmission by the sender and exponentially distributed timers at the receivers to a reliable multicast protocol that has support for retransmission from within the network.

Our performance measures were network bandwidth used for the reliable multicast and the completion time. Our findings indicate that the performance of a reliable multicast protocol without support from the network is very close to the performance of a reliable multicast protocol that is supported by the network, when the amount of data to transfer is large. For small amounts of data reliable multicast protocols with support from the network achieve better performance.

We further showed the benefits of combining the two approaches and considered a reliable multicast protocol that has both, support from the network, and integrated FEC. We showed that such a protocol based on local parity retransmissions performs best in terms of bandwidth and latency.

In this context, we found that loss recovery techniques from protocols that do not have support from the network may not yield the same benefits when applied in a reliable multicast protocol with local recovery: the benefits of parity retransmission are higher for source-based protocols than for local recovery protocols.

6.3 Outlook

The thesis has focused on aspects of reliable multicast. In order to be successfully deployed a few more open issues need to be resolved.

6.3.1 Multicast Congestion Control

One of the most important open issues for multicast networks is congestion control for multicast traffic. Congestion control is the means to avoid a network collapse by adjusting the amount of traffic emitted through an overloaded network node, referred to as the bottleneck. The congestion control algorithm of TCP handles congestion for *unicast* traffic over Internet bottlenecks. Several new issues arise with congestion control for multicast:

Heterogeneity A multicast tree can include several bottlenecks. Receivers located behind different bottlenecks have different bandwidth available through the different bottlenecks. Multicast congestion control therefore must deal with a set of several heterogeneous bottlenecks. Layered transmission schemes are a first approach towards a scalable solution to this problem: The sender transmits data via several multicast layers and a receiver decides on the number of layers it is able to receive. This technique, already demonstrated for video [78, 79] and audio [23], is a first step towards multicast congestion control for a heterogeneous set of bottlenecks. While for audio and video a different number of layers results in a different quality audio or video, reliable multicast requires a receiver to receive all data [22].

We showed that parity retransmissions leads to highly efficient reliable multicast. Another advantage of parities exists in the context of multicast congestion control. Parities spread the original information over a longer period of time and therefore give receivers a certain flexibility by receiving some time with a lower number of layers (a lower rate) than at other times. In order for a receiver to receive the whole content and to limit duplicate receptions, a sophisticated method is needed at the source to stripe data across layers.

Dynamics The bandwidth available through a bottleneck changes over time. Arriving and departing traffic, a changed network topology, or traffic controlled by other congestion control algorithms results in varying bottleneck conditions. New bottlenecks appear, old bottlenecks disappear and the traffic through each bottleneck is varying over time. Therefore, a challenge for multicast congestion control is to handle such dynamics.

Fairness Multicast traffic is copied inside the network towards the receivers. Therefore, multicast traffic uses the network bandwidth more efficiently than several unicasts to the same number of receivers. As a consequence, one can argue that multicast traffic should get some reward and should not be treated the same as unicast traffic in a bottleneck.

Defining congestion as the loss of utility to a multicast receiver due to high traffic loads, the task for congestion control is to maximize the sum of the receivers utilities at high traffic loads with the fairness constraint that all receivers of all multicast groups get a fair share of the bottleneck bandwidth. This is a definition that would favor a congestion control algorithm that weights multicast traffic through a bottleneck with the number of receivers that are reached by this traffic. Several methods to share the bottleneck bandwidth are possible and the subject of fairness is an open question that needs to be investigated.

Bibliography

[1] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs", *Proc. ACM SIGCOMM '88*, pp. 55–64, Stanford, CA, August 1988.

[2] S. Deering, "Host Extensions for IP Multicasting", *Internet Request for Comments*, RFC 1112, Network Information Center, August 1989.

[3] H. Eriksson, "MBONE: The Multicast Backbone", *Communications of the ACM*, 37(8):54–60, August 1994.

[4] M. M. R. and D. P. Brutzmann, "MBone Provides Audio and Video Across the Internet", *IEEE Computer*, April 1994.

[5] S. Keshav, *Congestion Control in Computer Networks*, Ph.D. Dissertation, EECS, University of Berkeley, CA 94720, USA, September 1991.

[6] Y.-D. Yao and S.-X. Cheng, "An ARQ Scheme for Point-to-Multipoint Satellite Communication Systems", *Proceedings of INFOCOM'87*, pp. 527–532, Department of Radio Engineering, Nanjing Institute of Technology, Nanjing, China, 1987, IEEE.

[7] D. Towsley, "An Analysis of a Point-to-Multipoint Channel Using a Go-Back-N Error Control Protocol", *IEEE Transactions on Communications*, 33(3):282–285, March 1985.

[8] K. Mase, T. Takenaka, H. Yamamoto, and M. Shinohara, "Go-Back-N ARQ schemes for point-to-multipoint satellite communications", *IEEE Transactions on Communications*, 31:583–589, 1983.

[9] E. J. Weldon, "An improved selective-repeat ARQ strategy", *IEEE Transactions on Communications*, 30:480–486, 1982.

[10] N. Shacham and D. Towsley, "Resequencing Delay and Buffer Occupancy in Selective Repeat ARQ with Multiple Receivers", *Proceedings of IEEE INFOCOM'87*, pp. 512–520, 1987.

[11] D. Towsley and S. Mithal, "A Selective Repeat ARQ Protocol for a Point to Multipoint Channel", *Proceedings of IEEE INFOCOM'87*, pp. 521–526, San Francisco, CA, USA, April 1987.

[12] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)", *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, 15(3):407 – 421, April 1997.

[13] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation", *Proceedings of SIGCOMM'95*, volume 25, ACM, October 1995.

[14] C. Papadopoulos and G. M. Parulkar, "Implosion Control for Multipoint Applications", *Proceedings of the 10th Annual IEEE Workshop on Computer Communications*, IEEE, September 1995.

[15] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[16] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, 1997.

[17] B. Levine., D. Lavo, and J. J. Garcia-Luna-Aceves, "The Case for Concurrent Reliable Multicasting Using Shared Ack Trees", *Proc. ACM Multimedia 1996*, pp. 365–376, Boston, MA, November 1996.

[18] S. Kasera, J. Kurose, and D. Towsley, "Scalable Reliable Multicast Using Multiple Multicast Groups", UM-CS-1996-073, UMASS, October 1996.

[19] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", *SIGCOMM '97*, pp. 289–300, Cannes, France, September 1997.

[20] L. Rizzo, "Effective erasure codes for reliable computer communication protocols", *Computer Communication Review*, 27(2):24–36, April 1997.

[21] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", *SIGCOMM 96*, pp. ??–327, August 1996.

[22] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[23] T. Turletti, S. F. Parisis, and J. Bolot, "Multicast Transmission of Layered Audio over the Internet", submitted to SigComm'97, INRIA, B.P.93, Sophia-Antipolis Cedex, France, February 1997.

[24] L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms", *Proceedings of ICCCN'94*, San Francisco, CA, USA, Sept 1994.

[25] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing", *Proceedings of IEEE INFOCOM'93*, volume 1, pp. 82–89, 1993.

[26] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicasting for Multimedia Applications", *Proceedings of IEEE INFOCOM'92*, pp. 2078–2085, 1992.

[27] C. A. Noronha and F. A. Tobagi, "Evaluation of Multicast Routing Algorithms for Multimedia Streams", *Proceedings of IEEE ITS'94*, Rio de Janeiro, Brazil, August 1994.

[28] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels", TR-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.

[29] T. Billhartz, J. B. Cain, E. Farrey-Goudreau, and D. Fieg, "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols in DIS Environments", *Proceedings of IEEE INFOCOM'96*, volume 1, pp. 85–93, San Francisco, CA, USA, March 1996.

[30] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network", *Proceedings of IEEE Global Internet*, London, UK, November 1996.

[31] P. Bhagwat, P. P. Mishra, and S. K. Tripathi, "Effect of Topology on Performance of Reliable Multicast Communication", *Proceedings of IEEE INFOCOM'94*, volume 2, pp. 602–609, Toronto, Ontario, Canada, June 1994.

[32] R. H. Deng, "Hybrid ARQ Schemes for Point-to-Multipoint Communication Over Nonstationary Broadcast Channels", *IEEE Transactions on Communications*, 41(9):1379–1387, September 1993.

[33] S. Pingali, D. Towsley, and J. F. Kurose, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *Proceedings of ACM Sigmetrics*, pp. 221–230, Santa Clara, CA, USA, May 1994.

[34] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees", *Acta Informatica*, 15:141–145, 1981.

[35] P. Winter, "Steiner problem in networks: a survey", *Networks*, volume 17, pp. 129–167, 1987.

[36] B. M. Waxman, "Routing of Multipoint Connections", *IEEE JSAC*, 6(9):1617–1622, December 1988.

[37] J. Kadirire, "Minimising Packet Copies in Multicast Routing by Exploiting Geographic Spread", *ACM Computer Communication Review*, 24(3):47–62, ACM Press, July 1994.

[38] J. Nonnenmacher and E. W. Biersack, "Reliable Multicast: Where to use FEC", *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, pp. 134–148, INRIA, Sophia Antipolis, FRANCE, October 1996, Chapman & Hall.

[39] A. Jean-Marie, "On the Average Number of Retransmissions in Multicast Applications", available from the author, INRIA, Sophia Antipolis, France, Aug. 1997.

[40] J. Nonnenmacher, "Ressource sharing in Multicast Connections", , Institut EURECOM, 2229 route des Crêtes, B.P. 193, 06904 Sophia Antipolis Cedex, FRANCE, February 1996.

[41] S. Lin and D. J. Costello, *Error Correcting Coding: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[42] S. Lin, D. J. Costello, and M. J. Miller, "Automatic-repeat-request error-control schemes.", *IEEE Commun. Magazine*, 22(12):5–17, 1984.

[43] J. Metzner, "An Improved Broadcast Retransmission Protocol", *IEEE Transactions on Communications*, COM-32(6):679–683, June 1984.

[44] C. Huitema, "The case for packet level FEC", *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, pp. 110–120, INRIA, Sophia Antipolis, FRANCE, October 1996, Chapman & Hall.

[45] S. Floyd, V. Jacobsen, S. McCanne, C.-G. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *Proceedings of SIGCOMM'95*, volume 25, pp. 342–356, ACM, October 1995.

[46] T. W. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP - THE XPRESS TRANSFER PROTOCOL*, Addison-Wesley, 1992.

[47] M. Hofmann, "A Generic Concept for Large-Scale Multicast", B. Plattner, Ed., *Proc. International Zuerich Seminar*, volume 1044 of *LNCS*, pp. 95–106, Springer Verlag, February 1996.

[48] R. Yavatkar, J. Griffoen, and M. Sudan, "A reliable Dissemination Protocol for Interactive Collaborative Applications", *Proceedings of ACM Multimedia*, pp. 333–344, San Francisco, CA USA, 1995, ACM.

[49] K. Sakakibara and M. Kasahara, "A Multicast Hybrid ARQ Scheme using MDS Codes and GMD Decoding", *IEEE Transactions on Communications*, 43(12):2933–2939, December 1995.

[50] R. H. Deng, "Hybrid ARQ Schemes for Point-to-Multipoint Communication over Nonstationary Broadcast Channels", *IEEE Transactions on Communications*, COM-41(9):1379–1387, September 1993.

[51] A. J. McAuley, "Reliable Broadband Communications Using a Burst Erasure Correcting Code", *Proc. ACM SIGCOMM 90*, pp. 287–306, Philadelphia, PA, September 1990.

[52] P. Morse, *Queues, Inventories, and Maintenance*, John Wiley, 1958.

[53] J. C. Bolot, "Analysis and control of audio packet loss in the Internet", T. D. C. Little and R. Gusella, Eds., *5th Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1018 of *LNCS*, pp. 163–174, Springer Verlag, Heidelberg, Germany, April 1995.

[54] J. Rosenberg and H. Schulzrinne, "Timer Reconsideration for Enhanced RTP Scalability", *Proceedings of IEEE INFOCOM*, March 1998.

[55] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error Control Using Retransmission Schemes in Multicast Transport Protocols for Real-Time Media", *IEEE/ACM Transactions on Networking*, 4(3):413–427, June 1996.

[56] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson, "Scalable Timers for Soft State Protocols", *Proc. INFOCOMM 97*, April 1997.

[57] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, Inc., 3rd edition, 1991.

[58] E. Castillo, *Extreme Value Theory in Engineering*, Academic Press, Inc., 1988.

[59] P. E. Pfeiffer, *Probability for Applications*, Springer Verlag, New York, Berlin, Heidelberg, 1990.

[60] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, 1976.

[61] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol; RFC 1075", *Internet Request for Comments*, (1075), Network Information Center, SRI International, Menlo Park, CA, nov 1988.

[62] J. Bolot, T. Turletti, and I. Wakeman, "Scalable Feedback Control for Multicast Video Distribution in the Internet", *Proceedings of SigComm*, ACM, September 1994.

[63] M. H. Ammar and G. N. Rouskas, "On the Performance of Protocols for Collecting Responses over a Multiple-Access Channel", *Proceedings of INFOCOM'91*, volume 3, IEEE, 1991.

[64] J. Bonnin and J. Pansiot, "A Scalable Collect", *HIPPARCH Workshop*, Sweden, June 1997.

[65] D. DeLucia and K. Obraczka, "Multicast Feedback Suppression Using Representatives", *IEEE Infocom97*, 1997.

[66] J.M.Chang and N.F.Maxemchuk, "A Broadcast Protocol for Broadcast Networks", *Proceedings of GLOBECOM*, Dec. 1993.

[67] I. Cidon and M. Sidon, "Conflict Multiplicity Estimation and Batch Resolution Algorithms", *IEEE Transactions on Information Theory*, 34(1):101–110, January 1988.

[68] Institute of Electrical and Electronics Engineers, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifi cations", *IEEE Std 802.3-1985*, IEEE, New York, NY, 1985.

[69] M. Grossglauser, "Optimal Deterministic Timeouts for Reliable Scalable Multicast", *IEEE Journal on Selected Area in Communications*, 15(3):422–433, April 1997.

[70] S. Pingali, *Protocol and Real-Time Scheduling Issues for Multimedia Applications*, Ph.D. Dissertation, UMass, September 1994.

[71] B. Levine and J. J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Protocols,", *Proc. Conference on Network Protocols (ICNP-96)*, Columbus, Ohio, October 1996.

[72] K. Obraczka, "Multicast Transport Mechanism: A Survey and Taxonomy", submitted to IEEE Communications Magazine, 1997.

[73] C. Diot, W. Dabbous, and C. J, "Multipoint Communication: A Survey of Protocols, Functions and Mechanisms", *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, April 1997.

[74] M. T. Lucas, B. J. Dempsey, and A. C. Weaver, "MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast Networks", *International Conference on Communication, ICC'97*, Montreal, Canada, June 1997.

[75] S. K. Kasera, J. Kurose, and D. Towsley, "A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast", *Proceedings of IEEE INFOCOM*, San Francisco, CA, USA, March 1998.

[76] L. Rizzo, "On the feasibility of software FEC", , Univ. di Pisa, Italy, January 1997.

[77] M. Lacher, "Analysis of error recovery techniques of reliable multicast: Centralized Error Recovery vs. Distributed Error Recovery", M.S. Thesis, EURECOM, Dec. 1997.

[78] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", *SIGCOMM 96*, pp. 117–130, Stanford, CA, August 1996.

[79] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast", SSC/1997/001, EPFL, Lausanne, Switzerland, January 1997.

[80] E. Ayanoglu, R. D. Gitlin, and N. C. Oguz, "Performance Improvement in Broadband Networks using Forward Error Correction for Lost Packet Recovery", *Journal of High Speed Networks*, 2:287–303, 1993.

[81] D. Rubenstein, "Using Packet-level FEC with Real-time Data Delivery", available from the author, CS department, University of Massachusetts, Amherst, May 1997.

Appendix A

Modeling

A.1 Lemm a 1

We proof a fundamental relation between routing and error recovery for 1:R – multicast communication. Given a homogeneous link loss probability q in a multicast connection (tree) with L links, where retransmissions are multicast we show that the expected number of retransmissions $E[M_S - 1]$ needed to deliver a packet from the source to *all* receivers can be approximated for *any* topology by qL, as long as $qL < 1$:

$$E[M_S - 1] \approx qL$$

This approximation is based on the following Lemma 1.

Lemma 1

The *CDF* $F_n(i)$ can always be expressed in the following form:

$$F_n(i) = 1 - \sum_{j_n^-} (Q_{j_n^-})^i + \sum_{j_n^+} (Q_{j_n^+})^i \tag{A.1}$$

Where the $Q_{j_n^-}$ and $Q_{j_n^+}$ are polynomials in q: $Q = \sum_k \zeta_k q^k$, with the following properties:

- the smallest exponent in Q is $k_{min} \geq 1$ and the coeffi cient of $q^{k_{min}}$ is a natural number $\zeta_{k_{min}} \geq 1$.
- Let
  - N_n^- be the number of polynomials $Q_{j_n^-}$ in $-\sum_{j_n^-} (Q_{j_n^-})^i$, e.g. the number of polynomials $Q_{j_n^-}$ indexed by j_n^-.
  - N_n^+ be the number of polynomials $Q_{j_n^+}$ indexed by j_n^+.

  Then it is always
  $$N_n^- = N_n^+ + 1$$
- Let

- Σ_n^- be the sum of the coeffi cients ζ_1 of all polynomials $Q_{j_n^-}$ that have a minimal exponent $k_{min} = 1$. It is possible to sum over all $Q_{j_n^-}$: $\Sigma_n^- = \sum_{j_n^-} \zeta_{1_{j_n^-}}$, since for polynomials with a minimal exponent $k_{min} > 1$ is $\zeta_1 = 0$.
- Σ_n^+ be the sum of the coeffi cients ζ_1 of all polynomials $Q_{j_n^+}$ that have a minimal exponent $k_{min} = 1$. As for Σ_n^- we can sum as $\Sigma_n^+ = \sum_{j_n^+} \zeta_{1_{j_n^+}}$.
- L_n be the number of links in the subtree rooted at n.

Then, when there is a link leading to n it is:

$$L_n + 1 = \Sigma_n^- - \Sigma_n^+$$

In the case where n is the source, then there is no link leading to n and it is:

$$L_n = \Sigma_n^- - \Sigma_n^+$$

A.1.1 Proof of Lemma 1

The proof proceeds by induction over the children c in $child(n)$ using Eqs. (2.8), (2.9), (2.10) and the binomial theorem.

$$(a+b)^i = \sum_{u=0}^{i} \binom{i}{u} a^u b^{(i-u)} \tag{A.2}$$

When proving Lemma 1 for node n the induction assumption is that Lemma 1 holds for the children of node n. The induction over the children must distinguish the cases of node n as a leaf, as the source, and as an intermediate node – just as in the defi nition of the *CDF* $F_n(i)$ given in (2.8), (2.9) and (2.10).

Case o)

The case where n is a **leaf** gives the induction basis.

$$F_n(i) =_{(2.8)} 1 - q^i = 1 - \sum_{j_n^-} (Q_{j_n^-})^i + \sum_{j_n^+} (Q_{j_n^+})^i$$

Where j_n^- indexes just one polynomial $Q = q$ with the smallest exponent $k = 1 \geq 1$ and the coeffi cient $\zeta_1 = 1 \geq 1$. j_n^+ does not index any polynomial. Trivially, there is one more polynomial in $-\sum_{j_n^-}(Q_{j_n^-})^i$, than in $+\sum_{j_n^+}(Q_{j_n^+})^i$ and

$$N_n^- = N_n^+ + 1$$

holds. The one polynomial $Q = (\zeta_1 q + \ldots) = q$ with exponent $k = 1$ and $\zeta_1 = 1$, yields $\Sigma_n^- = \sum_{j_n^-} \zeta_{1_{j_n^-}} = 1$ and since there's no polynomial $Q_{j_n^+}$ it is $\Sigma_n^+ = 0$. The number of links in the subtree rooted at l is $L_n = 0$ and there is a link leading to l and the equation

$$L_n + 1 = 1 = \Sigma_n^- - \Sigma_n^+$$

is true.

Case i)

For a node n, not a leaf we must distinguish two cases: the node n is the source S, or n is neither source, nor leaf. In both cases the induction assumption (I.A.) is that Lemma 1 is true for every child $c \in child(n)$ of n.
Case i.1.) n is the source S
Then due to the definition of $F_S(i)$ in (2.10):

$$\begin{aligned} F_S(i) &=_{(2.10)} \prod_{c \in child(S)} F_c(i) \\ &=_{I.A.} \prod_{c \in child(S)} (1 - \sum_{j_c^-} (Q_{j_c^-})^i + \sum_{j_c^+} (Q_{j_c^+})^i) \end{aligned}$$

The proof that $F_S(i)$ can be expressed as:

$$F_S(i) = 1 - \sum_{j_S^-} (Q_{j_S^-})^i + \sum_{j_S^+} (Q_{j_S^+})^i$$

with the given properties in Lemma 1 Eq. (A.1) is relatively straightforward and proceeds via another inner induction over the number $w = 1, \ldots, z$ of children $c \in child(S)$. For the ease of indexing we assume the children of S named 1,...,z.
Case i.1.o) The induction base is one child of source S ($w = 1$):

$$\begin{aligned} F_S(i) &=_{(2.10)} \prod_{c=1}^{w} F_c(i) = F_1(i) \\ &=_{I.A.} 1 - \sum_{j_1^-} (Q_{j_1^-})^i + \sum_{j_1^+} (Q_{j_1^+})^i \\ &= 1 - \sum_{j_S^-} (Q_{j_S^-})^i + \sum_{j_S^+} (Q_{j_S^+})^i \end{aligned}$$

and Lemma 1 Eq. (A.1) is true due to the outer induction assumption, since $c = 1$ indexes one child of n and for a child the induction assumption holds. Therefore:

$$N_S^- = N_S^+ + 1$$

is true for $F_S(i)$, since the number of polynomials indexed by j_1^- and j_1^+ does not change and $N_1^- = N_1^+ + 1$ was true for the child by induction assumption.
For child 1 it is $L_1 + 1 = \Sigma_1^- - \Sigma_1^+$, since there is a link leading to child 1.
There is only one child ($w = 1$) of the source S and the number of links in the tree rooted at S is just the number of links in the tree rooted at the child (L_1) plus one for the link from S to the child: $L_S = L_1 + 1$. It is $\Sigma_S^- = \Sigma_1^-$ and $\Sigma_S^+ = \Sigma_1^+$ for the source S as for the child 1, since $-\sum_{j_S^-} (Q_{j_S^-})^i = -\sum_{j_1^-} (Q_{j_1^-})^i$ and $+\sum_{j_S^+} (Q_{j_S^+})^i = +\sum_{j_1^+} (Q_{j_1^+})^i$. It is therefore

$$L_S = L_1 + 1 =_{I.A.} \Sigma_1^- - \Sigma_1^+ = \Sigma_S^- - \Sigma_S^+$$

Since there is no link leading to S, Lemma 1 is proven for the case of one child of the source.

Case i.1.i) The induction step from w to $w+1$ children of the source S.
This case uses the inner induction assumption, referred to as ($i.I.A.$) that for w children at the source S Lemma 1 is true for S. The induction step is adding one child, called $w+1$, for which Lemma 1 is also true by the outer induction assumption ($I.A.$). An extended notation $S(w)$ will be used to describe the source node S with w children. $F_{S(w+1)}(i)$ can be expressed as:

$$F_{S(w+1)}(i) =_{(2.10)} F_{w+1}(i)F_{S(w)}$$

Where:

$$\begin{aligned} F_{w+1}(i) &=_{I.A.} 1 - \sum_{j^-_{w+1}} (Q_{j^-_{w+1}})^i + \sum_{j^+_{w+1}} (Q_{j^+_{w+1}})^i \\ F_{S(w)} &=_{i.I.A.} 1 - \sum_{j^-_{S(w)}} (Q_{j^-_{S(w)}})^i + \sum_{j^+_{S(w)}} (Q_{j^+_{S(w)}})^i \end{aligned}$$

The product $F_{w+1}(i)F_{S(w)}$ results again in the following form for $F_{S(w+1)}(i)$:

$$\begin{aligned} F_{S(w+1)}(i) = 1 \quad &- \sum_{j^-_{S(w+1)}} (Q_{j^-_{S(w+1)}})^i \\ &+ \sum_{j^+_{S(w+1)}} (Q_{j^+_{S(w+1)}})^i \end{aligned}$$

The polynomials $Q_{j^-_{S(w+1)}}$ and $Q_{j^+_{S(w+1)}}$ are products of polynomials $Q_{j^+_{w+1}}$, $Q_{j^+_{S(w)}}$, $Q_{j^-_{w+1}}$ and $Q_{j^-_{S(w)}}$ as can be seen below.

$$\begin{aligned} -\sum_{j^-_{S(w+1)}} (Q_{j^-_{S(w+1)}})^i = \quad &- \sum_{j^-_{w+1}} (Q_{j^-_{w+1}})^i \\ &- \sum_{j^+_{w+1}, j^-_{S(w)}} (Q_{j^+_{w+1}} Q_{j^-_{S(w)}})^i \\ &- \sum_{j^-_{S(w)}} (Q_{j^-_{S(w)}})^i \\ &- \sum_{j^-_{w+1}, j^+_{S(w)}} (Q_{j^-_{w+1}} Q_{j^+_{S(w)}})^i \end{aligned} \tag{A.3}$$

$$\begin{aligned} +\sum_{j^+_{S(w+1)}} (Q_{j^+_{S(w+1)}})^i = \quad &+ \sum_{j^+_{w+1}} (Q_{j^+_{w+1}})^i \\ &+ \sum_{j^+_{w+1}, j^+_{S(w)}} (Q_{j^+_{w+1}} Q_{j^+_{S(w)}})^i \\ &+ \sum_{j^+_{S(w)}} (Q_{j^+_{S(w)}})^i \end{aligned} \tag{A.4}$$

$$+ \sum_{j^-_{w+1}, j^-_{S(w)}} (Q_{j^-_{w+1}} Q_{j^-_{S(w)}})^i$$

Due to the induction assumption the $Q_{j^+_{w+1}}$, $Q_{j^-_{w+1}}$, $Q_{j^+_{S(w)}}$ and $Q_{j^-_{S(w)}}$ are all polynomials $Q = \sum_k \zeta_k q^k$ in q with a minimal exponent $k_{min} \geq 1$ and the coeffi cient of $q^{k_{min}}$ is a natural number $\zeta_{k_{min}} \geq 1$. A product $Q_1 Q_2$ of two polynomials Q_1 and Q_2 with this property results in a polynomial in q with a minimal exponent $k_{min} \geq 2 \geq 1$ and a coeffi cient of $q^{k_{min}}$ that is again a natural number $\zeta_{k_{min}} \geq 1$. The polynomials $Q_{j^+_{S(w+1)}}$ and $Q_{j^-_{S(w+1)}}$ have therefore also a minimal exponent $k_{min} \geq 1$ and $\zeta_{k_{min}} \geq 1$.
The total number of polynomials $N^-_{S(w+1)}$ in the sum $-\sum_{j^-_{S(w+1)}} (Q_{j^-_{S(w+1)}})^i$ and the total number of polynomials $N^+_{S(w+1)}$ in the sum $+\sum_{j^+_{S(w+1)}} (Q_{j^+_{S(w+1)}})^i$ has again the property:

$$N^-_{S(w+1)} - N^+_{S(w+1)} = 1$$

To prove this, the number of polynomials in the above expressions (A.3) and (A.4) will be evaluated:

$$\begin{aligned} N^-_{S(w+1)} &= N^-_{(w+1)} + (N^+_{(w+1)})(N^-_{S(w)}) \\ &+ N^-_{S(w)} + (N^-_{(w+1)})(N^+_{S(w)}) \\ N^+_{S(w+1)} &= N^+_{(w+1)} + (N^+_{(w+1)})(N^+_{S(w)}) \\ &+ N^+_{S(w)} + (N^-_{(w+1)})(N^-_{S(w)}) \end{aligned}$$

The induction assumption gives

$$\begin{aligned} N^-_{(w+1)} - N^+_{(w+1)} &= 1 \\ N^-_{S(w)} - N^+_{S(w)} &= 1 \end{aligned}$$

and can be applied to $N^-_{S(w+1)} - N^+_{S(w+1)}$:

$$\begin{aligned} N^-_{S(w+1)} - N^+_{S(w+1)} &=_{i.I.A.} (N^-_{(w+1)} - N^+_{(w+1)}) \\ &+ N^+_{(w+1)} - N^-_{(w+1)} + 1 \\ &=_{I.A.} 1 \end{aligned}$$

In order to complete the proof of Lemma 1, we need to show that the number $L_{S(w+1)}$ of links in the tree rooted at the source S with $w+1$ children equals the difference of the sums of the coeffi cients ζ_1 of the polynomials in (A.3) and (A.4):

$$L_{S(w+1)} = \Sigma^-_{S(w+1)} - \Sigma^+_{S(w+1)}$$

The number $L_{S(w+1)}$ of links in the tree rooted at S in the case of $w+1$ children is just the number $L_{S(w)}$ of links in the case of w children at the source plus the number L_{w+1} of links in the tree rooted at child $w+1$ plus 1 for the link leading from the source S to this child:

$$L_{S(w+1)} = L_{S(w)} + L_{w+1} + 1$$

As stated before the products $Q_1 Q_2$ of polynomials Q_1 and Q_2 in Eq. (A.3) and Eq. (A.4) have a minimal exponent $k_{min} \geq 2$, which means that the coeffi cients $\zeta_{k_{min}}$ of these products do not influence $\Sigma^-_{S(w+1)}$ and $\Sigma^+_{S(w+1)}$. Therefore only the polynomials in the expressions $\sum_{j^-_{w+1}} (Q_{j^-_{w+1}})^i$ and $\sum_{j^-_{S(w)}} (Q_{j^-_{S(w)}})^i$ in (A.3) and the expressions $\sum_{j^+_{w+1}} (Q_{j^+_{w+1}})^i$ and $\sum_{j^+_{S(w)}} (Q_{j^+_{S(w)}})^i$ in (A.4) have to be considered. The sum of the coeffi cients ζ_1 in (A.3) and (A.4) is:

$$\begin{aligned}
\Sigma^-_{S(w+1)} &= \sum_{j^-_{S(w+1)}} \zeta_{1_{j^-_{S(w+1)}}} \\
&= \sum_{j^-_{w+1}} \zeta_{1_{j^-_{w+1}}} + \sum_{j^-_{S(w)}} \zeta_{1_{j^-_{S(w)}}} \\
&= \Sigma^-_{w+1} + \Sigma^-_{S(w)} \\
\Sigma^+_{S(w+1)} &= \sum_{j^+_{S(w+1)}} \zeta_{1_{j^+_{S(w+1)}}} \\
&= \sum_{j^+_{w+1}} \zeta_{1_{j^+_{w+1}}} + \sum_{j^+_{S(w)}} \zeta_{1_{j^+_{S(w)}}} \\
&= \Sigma^+_{w+1} + \Sigma^+_{S(w)}
\end{aligned}$$

This yields:

$$\begin{aligned}
\Sigma^-_{S(w+1)} - \Sigma^+_{S(w+1)} &= (\Sigma^-_{w+1} + \Sigma^-_{S(w)}) \\
&- (\Sigma^+_{w+1} + \Sigma^+_{S(w)}) \\
&=_{i.I.A.} \Sigma^-_{w+1} - \Sigma^+_{w+1} + L_{S(w)} \\
&=_{I.A.} L_{w+1} + 1 + L_{S(w)} \\
&= L_{S(w+1)}
\end{aligned}$$

Case i.2.) n **is neither the source** S**, nor a leaf**
We need to consider defi nition (2.9) of $F_n(i)$ for this case:

$$F_n(i) = \sum_{u=0}^{i-1} \binom{i}{u} q^u (1-q)^{i-u} \prod_{c \in child(n)} F_c(i-u)$$

The result from the preceding proof for the source S can be reused, since the expression $\prod_{c \in child(n)} F_c(i-u)$ has the same form as $F_S(i) = \prod_{c \in child(S)} F_c(i)$. Lemma 1 has been proven for the source and $\prod_{c \in child(n)} F_c(i-u)$ can therefore be expressed in the following form:

$$\prod_{c \in child(n)} F_c(i-u) = 1 - \sum_{j^-_m} (Q_{j^-_m})^{i-u} + \sum_{j^+_m} (Q_{j^+_m})^{i-u} \tag{A.5}$$

Here the node variable m instead of n is used, since n is not the source and there is a link leading to n. m describes the node n without the link leading from the parent to n. Because the properties in Lemma 1 have been proven for the source S, m has the same properties:
The $Q_{j_m^-}$ and $Q_{j_m^+}$ are polynomials in q: $Q = \sum_k \zeta_k q^k$.

- The smallest exponent in Q is $k_{min} \geq 1$ and the coeffi cient of $q^{k_{min}}$ is a natural number $\zeta_{k_{min}} \geq 1$.
- $N_m^- = N_m^+ + 1$
- $L_m = \Sigma_m^- - \Sigma_m^+$

Substituting $\prod_{c \in child(S)} F_c(i)$ by (A.5) in $F_n(i)$ and applying the binomial theorem (A.2) yields:

$$\begin{aligned}
F_n(i) \quad =_{(2.9)} \quad & \sum_{u=0}^{i} \binom{i}{u} q^u (1-q)^{i-u} \prod_{c \in child(n)} F_c(i-u) \\
=_{(A.5)} \quad & \sum_{u=0}^{i} \binom{i}{u} q^u (1-q)^{i-u} \\
- \quad & \sum_{u=0}^{i} \binom{i}{u} q^u \sum_{j_m^-} ((1-q) Q_{j_m^-})^{i-u} \\
+ \quad & \sum_{u=0}^{i} \binom{i}{u} q^u \sum_{j_m^+} ((1-q) Q_{j_m^+})^{i-u} \\
=_{(A.2)} \quad & 1 \\
- \quad & \sum_{j_m^-} (q + (1-q) Q_{j_m^-})^i \\
+ \quad & \sum_{j_m^+} (q + (1-q) Q_{j_m^+})^i \\
= \quad & 1 - \sum_{j_n^-} (Q_{j_n^-})^i + \sum_{j_n^+} (Q_{j_n^+})^i)
\end{aligned}$$

Where the indexes are just renamed $j_n^- = j_m^-$ and $j_n^+ = j_m^+$ and cover the same range. The $Q_{j_n^-}$ and $Q_{j_n^+}$ are again polynomials with the following relation to the $Q_{j_m^-}$ and $Q_{j_m^+}$:

$$Q_{j_n^-} = q + (1-q) Q_{j_m^-} \tag{A.6}$$
$$Q_{j_n^+} = q + (1-q) Q_{j_m^+} \tag{A.7}$$

The $Q_{j_m^-}$ and $Q_{j_m^+}$ are polynomials in q: $Q = \sum_k \zeta_k q^k$. The $Q_{j_n^-}$ and $Q_{j_n^+}$ have then the following form:

$$q + (1-q)Q = (\zeta_1 + 1)q + (\zeta_2 - \zeta_1)q^2 + \ldots \tag{A.8}$$

These polynomials are again polynomials in q and have again a minimal exponent $k_{min} = 1 \geq 1$ and the coeffi cient $\zeta_{k_{min}} = \zeta_1 + 1 \geq 1$ is again a natural number, since ζ_1 has this property. The

bijective mapping of polynomials Q_{j_m} to Q_{j_n} in (A.6) and (A.7) means further that the number of polynomials in $-\sum_{j_n^-}(Q_{j_n^-})^i$ is the same as in $-\sum_{j_m^-}(Q_{j_m^-})^{i-u}$ and the number of polynomials in $+\sum_{j_n^+}(Q_{j_n^+})^i$ is the same as in $+\sum_{j_m^+}(Q_{j_m^+})^{i-u}$:

$$N_n^- = N_m^- \qquad N_n^+ = N_m^+$$

Since $N_m^- = N_m^+ + 1$ holds it is also $N_n^- = N_n^+ + 1$.

$$L_m = \Sigma_m^- - \Sigma_m^+$$

is true for m, since m describes node n as the source, without the link leading to n. The link to n exists and the property

$$L_n + 1 = \Sigma_n^- - \Sigma_n^+$$

in Lemma 1 has to be proven. The ζ_1s in the coeffi cients $(\zeta_1 + 1)$ of the fi rst order terms q in (A.8) add up to $\Sigma_m = \sum_{j_m}\zeta_{1_{j_m}}$. The 1s in the coeffi cients $(\zeta_1 + 1)$ of the fi rst order term q in (A.8) add N_m^- (N_m^+) to Σ_n^- (Σ_n^+) – one time for every polynomial $Q_{j_m^-}$ $(Q_{j_m^+})$ of the N_m^- (N_m^+) polynomials. Such that:

$$\begin{aligned} \Sigma_n^- &= \Sigma_m^- + N_m^- \\ \Sigma_n^+ &= \Sigma_m^+ + N_m^+ \end{aligned}$$

This yields:

$$\begin{aligned} \Sigma_n^- - \Sigma_n^+ &= \Sigma_m^- - \Sigma_m^+ + N_m^- - N_m^+ \\ &= L_m + 1 \\ &= L_n + 1 \end{aligned}$$

□

Appendix B

Throughput Analysis

Let us first recall from [16] the equations for the processing rates for protocol N2.

$$
\begin{aligned}
1/\Lambda_s^{N2} &= E[X^{N2}] && \text{(B.1)}\\
&= E[M^{N2}]E[X_p] + (E[M^{N2}]-1)E[X_n] \\
1/\Lambda_r^{N2} &= E[Y^{N2}] && \text{(B.2)}\\
&= E[M^{N2}](1-p)E[Y_p] \\
&\quad +(E[M^{N2}]-1)\left(\frac{1}{R}E[Y_n] + \frac{R-1}{R}E[Y_n^{'}]\right) \\
&\quad +E[(M_r-2)^{+}]E[Y_t]
\end{aligned}
$$

We can derive the processing rates for NP in a similar way to N2, taking into account the time for encoding and decoding and the fact that feedback and retransmissions are performed for entire TGs. We define the following variables:

X_e	time to encode a packet at the sender, which is a function of k and h
c_e	constant encoding time factor
X_p	time to process the transmission of a packet
X_n	time to process a NAK at the sender
Y_p, Y_t	time to process a packet or timeout at a receiver
Y_d	time to decode a packet at a receiver, which is a function of k and l
c_d	constant decoding time factor
Y_n	time to process and transmit a NAK at a receiver
$Y_n^{'}$	time to receive and process a NAK at a receiver
l	number of lost packets in a TG
M_r	number of transmissions necessary for receiver r to successfully receive a packet
M^w	number of transmissions for all receivers to successfully receive a packet, $w \in \{N2, NP\}$
T_r	number of transmission rounds necessary for receiver r to successfully receive a TG
T	number of transmission rounds for all receivers to successfully receive a TG
X^w, Y^w	send and receive per packet processing times of protocol $w \in \{N2, NP\}$

Λ_s^w, Λ_s^w send and receive per packet processing rates of protocol $w \in \{N2, NP\}$
Λ_o^w throughput of protocol $w \in \{N2, NP\}$ for a one-to-many transmission

The parameters p and k remain as before. To simplify the analysis we make the following assumptions:

- Losses among receivers are independent.
- The sender never runs out of parities. Otherwise, receivers requiring more than h parities can be ejected.
- Per transmission round always only one NAK is sent. NAKs are never lost.
- The buffer at the receivers is large enough to store the packets from all the TGs that cannot yet be reconstructed.

$$\Lambda_o^{NP} = min\{\Lambda_s^{NP}, \Lambda_r^{NP}\} \tag{B.3}$$

With

$$\begin{aligned} 1/\Lambda_s^{NP} &= E[X^{NP}] && \text{(B.4)} \\ &= E[X_e] + E[M^{NP}]E[X_p] + \frac{E[T]-1}{k}E[X_n] \\ 1/\Lambda_r^{NP} &= E[Y^{NP}] && \text{(B.5)} \\ &= E[M^{NP}](1-p)E[Y_p] \\ &\quad +((E[T]-1)/k)\left(\frac{1}{R}E[Y_n] + \frac{R-1}{R}E[Y_n^{'}]\right) \\ &\quad +E[(T_r-2)^+]E[Y_t] + E[Y_d] \end{aligned}$$

$E[M^{NP}]$ is computed as $E[M]$ of Eq. (3.7).
For the RSE coder presented in [20] per packet coding and decoding times are:

$$\begin{aligned} E[X_e] &= k \cdot (E[M^{NP}] - 1) \cdot c_e && \text{(B.6)} \\ E[Y_d] &= k \cdot p \cdot c_d && \text{(B.7)} \end{aligned}$$

where $k \cdot p$ denotes the mean number of data packets per transmission group that are lost and need to be reconstructed and c_e and c_d are constants for encoding and decoding that depend on the particular hardware, the symbol size m and the the size P of the packets.
The mean number of transmission rounds is

$$\begin{aligned} E[T_r] &= \sum_{m=0}^{\infty}(1 - P[T_r \leq m]) \\ E[(T_r-2)^+] &= E[T_r] - P[T_r = 1] - 2P[T_r > 1] \\ E[T] &= \sum_{m=0}^{\infty}(1 - P[T \leq m]) && \text{(B.8)} \end{aligned}$$

with $P[T \leq m] = P[T_r \leq m]^R,\ m = 1, 2, 3, ...$
For $P[T_r \leq m]$ we use

$$P[T_r \leq m] \quad \approx \quad (1 - p^m)^k,\ m = 1, 2, 3, ..$$

from the expressions derived in [80], which assumes that the number of parity packets sent during a transmission round is the same as the number of parities needed by receiver r. Since the sender will, however, send the maximum number of parities required by any receiver, this assumption will yield an upper bound on the expected number of transmission rounds.
Figure B.1 shows that the analytical approximation provides an upper bound on the exact value of E[T].
The derivation of the exact number of transmission rounds can be found in [81], however its use is computationally very intensive.

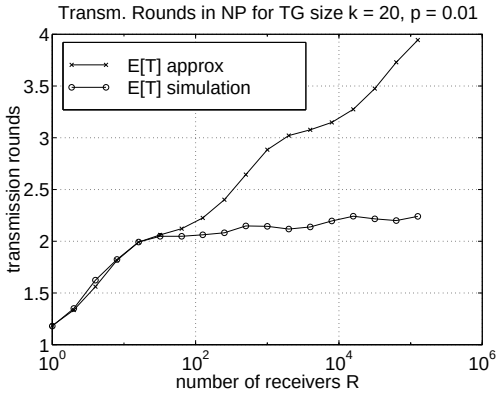

Figure B.1: The number of transmission rounds: Simulation and the analytical approximation for protocol NP with parameters $k = 20, p = 0.01$ for independent loss.