MASTER'S THESIS

Reliable Multicasting Based on Air Caching for Flat Hierarchy Networks

by Kyriakos Manousakis
Advisor: John S. Baras

CSHCN MS 2002-2
(ISR MS 2002-2)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

ABSTRACT

Title of Thesis: RELIABLE MULTICASTING BASED ON AIR CACHING FOR FLAT HIERARCHY NETWORKS

Degree candidate: Kyriakos Manousakis

Degree and year: Master of Science, 2002

Thesis directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

The evolution of satellite networks in the commercial and military world has pushed the research community towards the solution of important problems related to this kind of networks, which are characterized from the lack of physical hierarchy. One of those important problems is how to design an efficient reliable multicasting protocol for one-hop (flat) networks where the link may present characteristics like high propagation delay and high BER. The existing reliable multicasting protocols are inefficient when applied to flat networks, since those are based on intermediate receivers and local recovery techniques. We introduce the Air Cache, which serves as a fast access memory that is realized on the air and contains packets for the recovery of corrupted or erroneous data packets at the receivers. In this thesis we present two classes of reliable multicasting protocols, which are based on the combination of FEC and ARQ with air caching. The non-adaptive class of protocols (UDPAC, RDPAC, PPAC) is characterized by the static nature of the Air Cache in term of size and content and the second class of protocols

(ACDAC, ASPAC, HADAC) is characterized by the dynamic nature of the Air Cache. The objective is to achieve better performance than the existing reliable multicasting protocols in flat hierarchy networks. We evaluate the proposed protocols in terms of the delay, robustness, scalability and the hardware requirements that they pose. The results that we collected prove the effectiveness of air caching when it is combined with traditional techniques like FEC and ARQ.

RELIABLE MULTICASTING BASED ON AIR CACHING

FOR FLAT HIERARCHY NETWORKS

by

Kyriakos Manousakis

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Sciences
2002

Advisory Committee:

Professor John S. Baras, Chair
Professor Nick Roussopoulos
Professor Mark A. Shayman

DEDICATION

To my mother Athanasia, to my father Kostantinos

and to my two sisters Ariadni and Artemis

for their continual love and support

ACKNOWLEDGEMENTS

I am grateful to my advisor Dr. John S. Baras for his advice, support and encouragement. I would also like to thank Dr. Nick Roussopoulos and Dr. Mark A. Shayman for agreeing to serve on my committee and to review this thesis.

Special thanks are due to Maria Striki, for her continual love and support and the numerous hours that we spent together in the lab. Also, I would like to thank my friends from the Greek student organization DIGENIS for creating such a joyful and refreshing environment. Last but not least, I would like to thank the system administrators of the Center for Satellite and Hybrid Communication Networks lab for the maintenance of the hardware equipment and their assistance.

The research reported in this thesis was supported through collaborative participation in Advanced Telecommunications/Information Distribution Research Program (ATIRP) consortium sponsored by the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL 01-96-2-0002. This support is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1: Introduction

1.1 Introduction

In the recent years the deployment of hybrid and satellite networks has been evolved dramatically. This is something that we had being expected to happen for two reasons. The first one is the large increase of network users, which created new demands for network speed and bandwidth and the second one is the launch of new commercial applications and services from the various network companies in order to attract the increased number of network users. Obviously, the dialup connections are unable to carry the load of the network due to small bandwidth. So, one alternative solution for someone is to take advantage of the bandwidth carried from the satellite links. A lot of the network companies that wanted to participate in this promising evolution of the high speed networking market launched a large number of commercial satellites. The number of applications and services that are being offered to every household and business through satellite media has been increased, and nowadays those services vary from digital television and digital music to multiparty video conferencing tools. The concepts of high speed networking, large available bandwidth and the offering of some traditional copper wire and RF services (e.g., television, radio, video, teleconferencing) through the network seems a huge step in the transition towards the digital world where we demand high quality video and audio to be delivered to a large number of subscribers. On the other hand this transition creates new requirements for the efficient utilization of network resources in order to achieve high Quality of Service (QoS) and also creates difficult

challenges to the network engineers that have to deal with the new problems and redesign the larger part of the traditional network and extend it to other interconnection media.

Apart from the large bandwidth that the satellite links offer, there are also disadvantages related to them like large propagation delay and high bit error rate (BER). The latter two important characteristics of the satellite links need to be taken into consideration when we design protocols that operate in such a network environment. But this is not the only crucial point in the design of network protocols. Another very important property that these protocols are required to have is the service of very large number of users who must receive the highest possible quality of service (QoS). The first step in order to accomplish this is the efficient utilization of network resources and specifically in satellite links we have to use as efficiently as possible the available bandwidth. One obvious solution towards this step is the use of multicasting as a way of delivering services to group of users. The nice characteristic of multicasting is that it treats the recipients of information as groups and not as individuals, which is the case in unicasting. So, with a single flow of data you can service more than one user under the assumption that these users are willing to receive the same data. This assumption is valid in today's services like satellite television, radio and other multiparty applications. Intuitively, the savings in bandwidth usage are expected to be very promising and as an extension to this, the scalability of services based on multicasting is expected to be one of their advantageous characteristics.

Based on the previous discussion, multicast networking and its corresponding applications are becoming increasingly popular solutions for new suites of Internet products and services and especially for the satellite-based communications since

satellites are naturally a broadcast medium, a fact that favors the multicast communications. In multicast networking, data is transmitted from either one sender to many recipients or many senders to many recipients. Multicast applications offer many appealing benefits when trying to disseminate information to a large group of users. One of the most important benefits of multicasting is the efficient use of bandwidth compared to unicasting and broadcasting. There are two broad classifications of multicast applications:

- Those that require reliable delivery of data to the multicast group – every member in the multicast group should receive all the transmitted packets correctly, in other words those applications can not tolerate erroneous or corrupted packets
- Those that do not require reliable delivery of data to the multicast group – the members of the multicast group can tolerate some erroneous or corrupted packets

The Protocols, which favor both classes of multicast applications need to be studied in a variety of different network topologies.

In this thesis, we focus on applications that require reliable delivery of data. The problem of reliability in multicast communications, which are happening on top of unreliable underlying media with high propagation delay characteristics (satellite links), is difficult and more complex than the unicast case. When we consider the propagation delay penalties, which the satellite links pose to the reliable multicasting protocols, the issue of error recovery becomes extremely crucial. In the Automatic Repeat Request (ARQ) error recovery method, a feedback channel is required for proper protocol function. However, due to the high latency over satellite links, such protocols experience significant throughput degradation (e.g. the TCP degradation observed over high latency

links [1]). For this reason, the schemes that are based on the Forward Error Correction (FEC) error recovery method appear to be the more promising of the two approaches. Unfortunately, pure FEC does not require a feedback channel and therefore does not guarantee reliability. A hybrid approach that uses parity packets to reduce the residual packet error probability and a feedback mechanism to ensure reliability combines the advantages of both FEC and ARQ schemes to form a more robust protocol. There are variations of hybrid error control (HEC) protocols that offer additional potential benefits. One important variation uses local recovery to limit the amount feedback to and the number of transmissions from the source.

The HEC along with the local recovery schemes are the ones traditionally used for the design of reliable multicasting protocols for satellite networks. Although those schemes present very promising latency characteristics there is space for improvement. Most of corresponding studies [40] assume that the transmission over the satellite links is error-free and they focus on applying the reliable multicast schemes on the terrestrial networks, which contain satellite enable receivers. So, there is a necessity to study the latency performance of reliable multicasting protocols focusing on the satellite link. Furthermore, the robustness and scalability of the reliable multicasting protocols over the high BER and large propagation delay satellite link needs to be explored as well.

In this thesis we propose a number of reliable multicasting protocols having as goal the performance improvement of the reliable multicast transmission over the satellite links. Those schemes use a combination of ARQ and FEC [34], as is the case for the majority of successful reliable multicasting protocols ([17], [15], [36]). Furthermore and in combination with the ARQ and FEC we introduce technique, which is new in the area

of reliable multicasting protocols and is called Air Caching ([2], [7]). Air Caching is described from the continuous dissemination of the most requested or the most important data for fast delivery. Intuitively we expect this technique to benefit more the reliable multicasting protocols designed for satellite networks because of the broadcast characteristics of the satellite links, which favor Air Caching and the large propagation delay that those links present, which can be reduced because of the proactive characteristics of Air Caching. Obviously, Air Caching could be used for the fast recovery of the erroneous or corrupted packets. How this is done and what is the performance of protocols based on the combination of those techniques (e.g., ARQ, FEC and Air Caching) is described in this thesis through the design and evaluation of a variety of reliable multicasting protocols.

In this thesis we study the performance behavior in terms of latency, scalability, robustness and efficiency of a number of protocols that we will propose, whose functionality is based on Air Caching and combinations of FEC and ARQ. Originally, we propose these protocols for satellite networks, which is the assumed environment for their evaluation. On the other hand, these reliable multicasting protocols could be used in wireless environments without change in their functionality. The principal operation of these protocols is based on the utilization of two transmission channels C_1 and C_2. C_1 is used for the transmission of data packets, and C_2 is used for the transmission of the Air Cache contents. The C_2 channel contains packets chosen with a specific algorithmic way based on the protocols' functionality (e.g., most requested for retransmission data packets, parity packets, etc.), for the quick recovery of the data packets that are transmitted in C_1 and they have been corrupted.

The protocols that we proposed are classified in two categories:

- The non-adaptive Air Cache protocols
- The adaptive Air Cache protocols

based on the way we update the Air Cache per transmission round, and if we utilize the feedback from the multicast group for the update of the Air Cache. The adaptation process of Air Cache can happen potentially in two dimensions, where the first dimension is the content and the second one is the size. The adaptation process can happen to either one of those dimensions but also to both of them simultaneously. If no adaptation is involved, the Air Cache size remains constant and the Air Cache content is not based on the feedback from the multicast group.

In chapter 2 we will give an overview of the existing reliable multicasting protocols, and of the error recovery methods, which are preferred in the design of those protocols. A more detailed overview of the Air Caching, the functionality and the principals of operation of the adaptive and non-adaptive protocols is given in chapter 3. Then, we present in more detail, the three non-adaptive Air Cache reliable multicasting protocols and the three adaptive Air Cache reliable multicasting protocols that we propose along with their performance evaluation and comments about their practical usefulness of these, their pros and their cons. Also, comparison of these protocols with already existing reliable multicasting protocols will be included, in order to get a better idea of their relative performance. The non-adaptive protocols are presented in chapter 4 and the adaptive ones in chapter 5. The last chapter (i.e., chapter 6) provides us with an overview of the conclusions that we reached throughout this study and there are some proposals for future research.

Chapter 2: Related Work and Air Caching

2.1 Background

Network communications can be broadly classified based upon the number of senders, the number of receivers, and the association between these two types of network entities. Point-to-point communications occur between one sender and one receiver. Classical unicast services belong to the point-to-point communication class. When communications occur between one sender and a set of receivers, it is referred to as point-to-multipoint. Point-to-all communications occur between a single receiver and all associated partners. An example of point-to-all communication is a broadcast service over a local area network. Communication between a group of network entities able to both send and receive data is known as a multipoint-to-multipoint system. [4]

Whereas unicast services enable point-to-point communications, multicast services enable both point-to-multipoint and multipoint-to-multipoint communications. Some multicast services do not require reliable communication such as distributed gaming, distance learning applications, or distributed interactive simulations[15]. Other applications require reliable multicast service. These applications can be categorized based upon their data content (i.e. data-only or multimedia) and their delay constraints (real-time or non-real-time). For example, video-replication is a data-only, non-real-time application. Financial stock quote dissemination is considered a data-only, real-time application. One data-only, non-real-time application is database replication [15].

The number of applications that can benefit from reliable multicast transport services is growing as the number of Internet applications increases. The military also has

the need for a reliable multicast transport service that can deliver important information to many users/units deployed upon a battlefield. This information could include terrain maps, intelligence information, and orders of battle.

2.1.1 Multicast versus Unicast

If a unicast service were used to provide point-to-multipoint communications, then the source must generate a copy of a particular packet to be sent to each individual receiver. Thus the source is required to copy and address each duplicate packet with a different address. Rather than generate a duplicate packet for each receiver, multicast services first establish a cycle-free routing tree connecting the sender with the set of receivers. When sending a packet, the source addresses the packet to the multicast group connected by the multicast tree. Every node in the tree forwards an arriving packet to its children. Therefore, the source only sends a packet to its direct children. Upon receipt of a packet, each node forwards a copy to each of its children. Two major advantages of multicast transmission are simplicity of addressing and more efficient use of bandwidth.

As multicast services enable point-to-multipoint and multipoint-to-multipoint communications, they have a different set of requirements than unicast services. These requirements can be subdivided into five different areas; addressing, group membership, routing, network heterogeneity, and error control mechanisms. Multicasting requires an aggregated address per group enabling many anonymous members to participate in a session; whereas unicasting requires that the sender knows only the receiver's address. In multicast communications, there is a concept of group membership. Group membership can remain static or may dynamically change over a multicast session. The complexity of

routing in multicast environments is generally higher than in their unicast counterparts. As group membership dynamically changes, the complexity of routing further increases. The routing protocol's efficiency impacts multicast service performance. Multicast groups often contain receivers that experience different network characteristics. Network heterogeneity impacts the quality of service guarantees, the negotiation of any session parameters, and the flow and congestion control. Both multicast and unicast communications need error control mechanisms to recovery from erroneous data. However, these mechanisms are especially important when the multicast service requires reliable delivery of data.

There are five basic issues involved when considering reliable multicast protocols [23]. The first issue, the request implosion problem, occurs when the loss of a packet results in many simultaneous repair requests. Such a phenomenon can overwhelm the sender and possibly other receivers depending upon how the feedback is transmitted to the group. The second issue, duplicate replies, occurs when multiple repairs are sent from group members responding to a request (this happens primarily in local recovery situations). The third issue, recovery latency is the time required for a repair to be received after a loss is detected. Recovery isolation is very important in correlated-loss situations where a loss on an upstream link causes a group of receivers to lose the same packet. The fourth issue, recovery isolation, is the desire to isolate non-local nodes from receiving local repairs. Finally, the issue of adaptability to dynamic membership changes addresses the time required to return to a steady state after group membership or topological changes.

There has been a great deal of research focused on the many aspects of multicast and reliable multicast protocols. More recently, there has been an explosion of proposed protocols and schemes to provide reliable multicast services. This thesis primarily focuses on the design of reliable multicasting protocols for one-hop networks (e.g., flat hierarchy networks). The protocols that we will propose are based both on existing error recovery techniques (e.g., ARQ and FEC) but also on recently introduced methods (e.g., air caching). We will study, how the protocols that we designed behave in error prone environments where the links are characterized from large propagation delay and the lack of hierarchy eliminates the advantages of local recovery, caching and suppression of NACKs. Prior to studying the designed protocols, a detailed overview of the main thrusts of research on error control mechanisms and architectures is presented in the following sections.

2.1.2 Error Control Mechanisms and Architectures

This section deals with the mechanisms and architectures used by reliable multicast protocols to recover from packet errors or packet losses. Error control architectures can be classified based upon group member participation in error recovery. In centralized-error recovery (CER) architectures all retransmissions are sent from the original source; whereas in distributed error recovery (DER) architectures other intermediate nodes are allowed to participate in the retransmission of packets [17]. Within each of these architectures, there are two mechanisms that can be used to provide error recovery; Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). Additionally, these two mechanisms can be combined to form hybrid schemes that provide more robust

reliability. This section discusses error control mechanisms and explores these mechanisms within different error control architectures.

2.1.2.1 Error Control Mechanisms

2.1.2.1.1 Automatic Repeat Request (ARQ)

Automatic Repeat Request is a mechanism that recovers lost packets using retransmissions initiated by feedback. There are two basic forms of ARQ protocols; sender-initiated and receiver-initiated [26]. In sender-initiated protocols, the sender must ensure the reliable delivery of data to the entire multicast group via a multicast tree. Upon sending a particular packet, the sender starts a timer. Each receiver that correctly receives the packet sends a positive acknowledgement (ACK) to the sender. For each transmitted packet, the sender maintains a list of the receivers from which it has received an ACK. When the timer expires, the sender determines if all of the receivers in the group have acknowledged the packet. If at least one receiver is missing the packet, the packet is retransmitted and the timer restarted.

There are several disadvantages that make sender-initiated protocols undesirable for multicast environments. First, the sender must maintain packet state information for each receiver resulting in increased processing requirements for large multicast groups. Second, updating this state information for each receiver requires a significant amount of feedback and requires greater processing requirements at the source. This feedback results in additional network congestion on links surrounding the sender. Therefore,

sender-initiated protocols are limited in that they do not scale well to support large multicast groups. [26]

Whereas sender-initiated protocols place the responsibility of reliable delivery upon the sender, receiver-initiated protocols place most of this responsibility upon the receivers [26]. In receiver-initiated protocols, the sender continually transmits new data packets over a multicast tree until a negative acknowledgement (NAK) is received. A NAK is generated when a receiver detects a lost packet. Packet loss detection is performed by observing “gaps” in the received packets sequence numbers. If a “gap” is detected, then a NAK is generated. When sending a NAK, the receiver will start a NAK retransmission timer. If this timer expires before the correct reception of the requested packet, then the receiver resends a NAK to the sender.

Generating NAKs in such a manner results in the request (or feedback) implosion problem. This phenomenon occurs when a significantly large number of negative acknowledgements are transmitted to the sender. These feedback messages must traverse links immediately surrounding the sender and thus result in network overload and congestion. Therefore, a NAK suppression mechanism is desirable so that the number of NAKs actually reaching the sender is significantly decreased (ideally down to 1). Most suppression mechanisms use timers in conjugation with multicast NAKs. In such a mechanism, any receiver detecting a loss waits for a random amount of time prior to sending a NAK. Although, the characteristics of the timer are not discussed, they are important to the performance of the suppression mechanism [19]. If no other NAK for the same packet is received within the time interval, then the receiver proceeds and multicasts the NAK to the entire multicast tree (includes the sender and all other

receivers). If a receiver receives a NAK prior to its timer expiration, then it suppresses its NAK. This particular receiver sets its NAK retransmission timer so that the feedback cycle resets if the request is not fulfilled. Such a mechanism enables the receivers to coordinate NAK generation and thus reduce the number of feedback messages received at the source. Some form of NAK suppression mechanism is assumed in subsequent discussions.

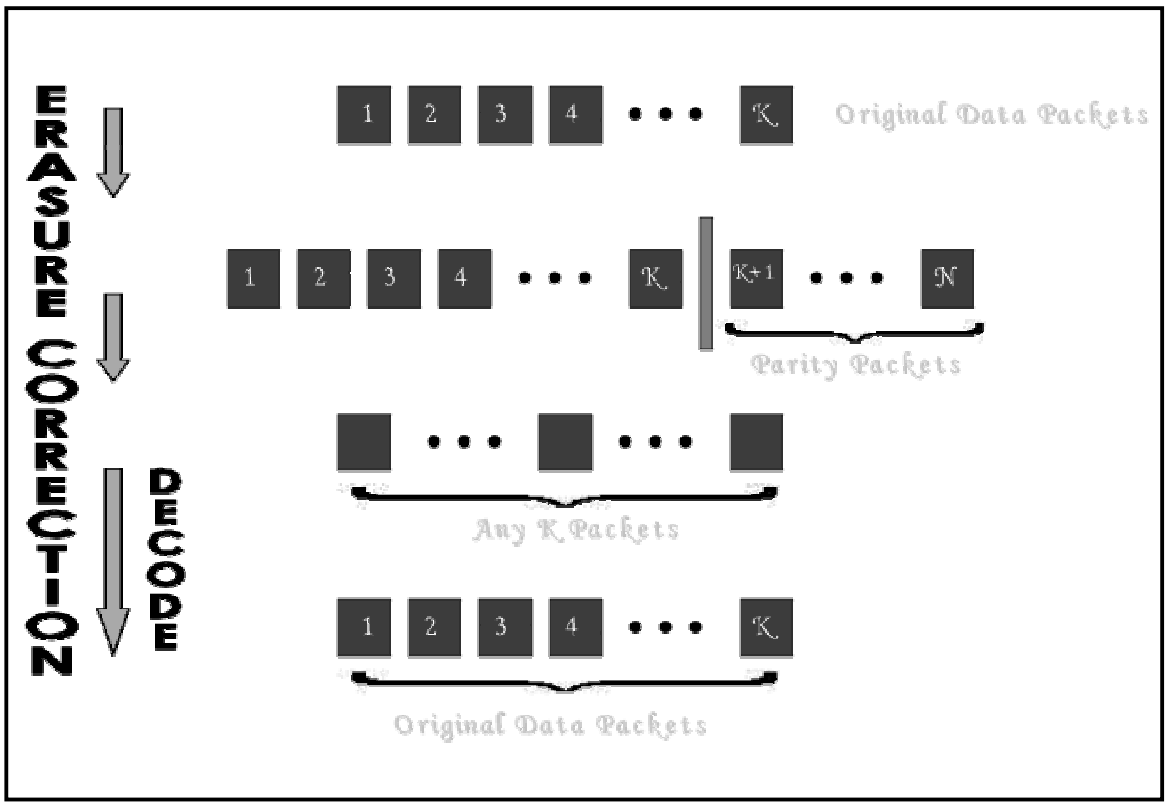
2.1.2.1.2 Forward Error Correction

Whereas ARQ is a reactive mechanism activated by packet losses during transmission, FEC is a proactive mechanism that transmits redundant data so receivers can reconstruct the original message even in the presence of communication errors. This statement offers some intuitive feel of the tradeoffs present when considering ARQ or FEC as error control mechanisms. Since ARQ schemes only retransmit data when errors occur, they waste little bandwidth at the cost of the delay incurred by waiting for feedback. FEC mechanisms transmit redundant data in anticipation of errors thereby reducing feedback delay. However, this redundancy may waste bandwidth over relatively error-free links.

The process of generating redundant data (or parity) requires the processing of an entire data stream. Such an encoding process is computationally expensive. The high computational cost of FEC implementations is not a concern in bit level communication systems where the encoder/decoder is usually implemented in dedicated hardware. At this level hardware implementation is usually much cheaper than having a feedback channel. In computer communications, the feedback channel often requires little overhead and

FEC requires a noticeable processing overhead for the host systems. However, as shown by Rizzo in [24], packet level FEC can be effectively implemented with software.

In coding theory there are two types of errors; corruption and erasure. Corruption of data occurs when bits are corrupted; whereas the erasure of data occurs when whole packets are lost [17]. In multicast protocols, the use of FEC mechanisms is mainly restricted to the recovery from erasures thereby reducing the effect of packet loss at different receivers. In other words, as long as a receiver collects a sufficient number of different packets, reconstruction of the original data is possible independent of the received packets' identity. For example, k data packets can be encoded to produce n packets (where n > k) so that if any k-subset of the n packets is correctly received, then



the data packets can be reconstructed.

Figure 2.1: Operation of Forward Error Correction (FEC) Method

By producing h = n – k parity packets and sending these packets with the *k* data packets, the residual error rate is reduced. This property of FEC mechanisms improves reliable multicast protocols scalability irrespective of the actual loss pattern at each receiver [17]. Additionally, the reduction in the residual loss rate (after decoding) largely reduces the need to send feedback to the sender, thus minimizing the use of the channel and simplifying feedback handling.

There are a variety of different ways to perform FEC encoding on a packet stream. The following description of the RSE schemes mirrors the one presented in [17]. Other coding descriptions are developed in [14] and [24]. In most cases, a Reed-Solomon erasure (RSE) code is used to generate the redundant packets. As previously stated, *k* represents the number of data packets of length P bits. These data packets are represented as $d_1, d_2, \ldots, d_k$. The RSE encoder takes $d_1, d_2, \ldots, d_k$ and produces parities $p_1, p_2, \ldots, p_{n-k}$. For the purpose of coding, consider the vector $\bar{d} = [d_1, d_2, \ldots, d_k]$ of data packets as elements of the Galois field $GF\left(2^P\right)$. Given the primitive element α of $GF\left(2^P\right)$, the (n,k) matrix $G' = \left[g_{i,j}\right]$ with elements in $GF\left(2^P\right)$ is defined as

$$g_{i,j} = \alpha^{i \cdot j} \quad 0 \le i \le 2^P - 2, \quad 0 \le j \le k-1.$$

The basic RSE encoder can produce up to $n = 2^P - 1$ FEC packets as components of

$$\bar{y}' = [y_1', y_2', \ldots, y_k'] = G' \bar{d}^T$$

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

Since the basic RSE scheme is not a systematic code, the data packets $d_1, d_2, \ldots, d_k$ are not part of $\bar{y}'$. Consequently, the RSE decoder must always solve k simultaneous linear equations to retrieve the data packets from k components of $\bar{y}'$. This decoding complexity can be avoided by using Gaussian elimination on the matrix G' prior to encoding. This operation modifies the first k row vectors of G' into a (k,k) identity matrix and creates a new matrix G. Using this G in the encoding $\bar{y} = G\bar{d}^T$, the first k components of $\bar{y}$ are copies of $d_1, d_2, \ldots, d_k$. The remaining $n-k$ components of $\bar{y}$ are the parities $p_1, p_2, \ldots, p_{n-k}$. These parity packets are the last $n-k$ packets in $\bar{y}$ (e.g., $p_i = y_{k+i}$ for $i \in \{1, \ldots, n-k\}$).

An example presented in [12] demonstrates how a Reed-Solomon erasure code is used in actual reliable multicast protocols. In this example, a shortened RS code was designed around 8 bit symbols and used an RS(255,k) code as its basis. First, *x* data symbols at the encoder and decoder are zero-filled so that a shortened RS codeword can be created. This zero-filling results in the transmission of *k-x* codewords over the link. By choosing k=235 and x=215, a shortened RS(40,20) code is formed that enables the generation of 20 parity packets per 20 data packets. These parity packets are transmitted as determined by the reliable multicast protocol. The following section discusses the different ways in which the parity packets are transmitted.

There are multiple benefits using the parity packets for loss recovery instead of retransmitting the lost 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.

- Improved scalability in terms of group size:

  In an ARQ scheme retransmitting the original packets that are lost the sender needs to know the sequence numbers of the lost packets. Using parity packets for loss repair, the sender needs only to know the maximum number of packets lost by any receiver 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, result in unnecessary receptions at all receivers that do not need the retransmission. Such duplicate packets waste transmission bandwidth and processing capacity.

2.1.2.1.3 Hybrid Error Control (HEC)

Even though FEC reduces the residual error rate, it cannot provide full reliability since there is no mechanism by which data can be retransmitted. Therefore, FEC needs to be augmented with an ARQ scheme so information can be retransmitted if network conditions deteriorate beyond a certain threshold. This threshold is normally determined as the maximum allowable number of packet errors occurring over the communication links or the total number of transmitted parity. ARQ mechanisms can reliably deliver data but require the retransmission of individually lost packets. FEC strengthens ARQ mechanisms because a single parity packet can correct different packet errors at different

receivers. Therefore, the combination of these two schemes results in a more robust mechanism that can guarantee reliability.

These two schemes can be combined in two ways; layered and integrated. In the layered approach [17], the reliable multicast (RM) layer sits atop an FEC layer in the protocol stack. The RM layer generates k data packets (called a transmission group, TG), which are sent to the FEC layer. Then the FEC layer generates h parity packets using the k data packets. Both the data and parity packets (a total of $n = k + h$ packets – called an FEC block) are multicast to the receivers in the group. If fewer than k of these n packets are received, then the data cannot be reconstructed at the receiver. The receiver discards the received parity packets and requests the lost originals from the sender. These lost originals are transmitted as part of a new FEC block. For many situations [17], this scheme makes more efficient use of network resources than ARQ schemes since the numbers of repair requests and numbers of retransmissions are reduced.

The integrated approach fuses the RM and FEC layers into one protocol stack layer. Through this integration, parity is used more efficiently. For example, a parity packets (known as autoparity packets) may be sent in the initial transmission of k data packets. If the receivers lose more that a of the $k + a$ packets, then they must request new parity so that the k data packets can be reconstructed. Upon receiving parity requests, the sender multicasts parity packets until all parity packets have been used. When the parity has been used, packets requiring retransmission are placed in the next TG. A performance comparison of the pure ARQ, layered FEC, and integrated FEC schemes was based upon the average number of transmissions needed to reliably send a packet to all receivers in a multicast group [17]. This comparison showed that both integrated and layered FEC

outperform the ARQ scheme for a large number of receivers (over 100). For smaller numbers of receivers, the ARQ scheme outperformed the layered FEC scheme because there were fewer than *h* errors and thus bandwidth was unnecessarily wasted. When *a* equals zero, the integrated FEC scheme performed better than both other schemes over all group sizes.

By integrating FEC with ARQ mechanisms, the number of transmissions and therefore the bandwidth usage is reduced. This reduction occurs because the resource usage is shifted from the network to the source and the receivers in the form of parity encoding and/or decoding. Through the encoding and decoding of parity packets, the error-control feedback is reduced. Due to lower amounts of feedback and more efficient bandwidth usage, integrated FEC protocols have good scalability properties (i.e. – the protocol performs acceptably for up to 1 million receivers) [17].

There are other ways to implement hybrid FEC-ARQ schemes. Generally, most differences are with the methods used to deal with the depletion of fresh parity. The above scheme placed any packets requiring retransmission in the next FEC block when parity was depleted. However, for large number of receivers (much larger than the size of the TG), there is a high probability that all packets within the TG will need to be retransmitted. There are other schemes that propose the use of an explicit ARQ repair phase when the parity is depleted [6]. Regardless how depleted parity is handled, hybrid FEC-ARQ schemes have been shown to yield better bandwidth performance than strictly ARQ schemes.

2.1.2.1 Error Control Architectures

In the preceding discussion, it was implicitly assumed that only the original source is allowed to generate retransmissions. Therefore, the above mechanisms fit within the CER (or global recovery) architecture. In DER (or local recovery) architectures, the responsibility of error recovery is distributed amongst members of the multicast group by allowing non-source nodes to aid in recovery. Distributed error recovery can further be divided into two sub-classifications; ungrouped DER and grouped DER. In ungrouped DER, any member of the global multicast group has the ability to perform retransmissions. Therefore, any particular node can potentially send retransmissions to the entire multicast group. Like ungrouped DER, grouped DER allows non-source nodes to retransmit data. However, in grouped DER, these retransmissions are performed within the local group (or local neighborhood). Intuitively, DER architectures reduce the amount of network-wide bandwidth consumed and contain errors to the locality in which they occurred. Since lost packets are recovered by local retransmissions, these architectures have the potential to provide significant performance gains in terms of end-to-end delay and higher system throughput. In the remainder of this section, several DER architectural examples are discussed.

2.1.2.2.1 Local Recovery Approaches

In this section, server-based local recovery and receiver-based local recovery are described. In the receiver-based approach, a receiver attempts to locally recover packets from the end hosts within its local neighborhood. In other words, any receiver that has correctly received a requested packet has the ability to send retransmissions during local

recovery. An example of this type of approach is Scalable Reliable Multicast (SRM) with local recovery enhancements [3]. The SRM protocol proposes a general framework and requires additional specifications to be specialized for a particular application. This protocol allows any receiver that has the correct data to generate repairs. Such a concept increases scalability by reducing administrative feedback to the source. To suppress duplicate repair requests, receivers requiring data wait a random period of time prior to issuing their requests. Repairs are made following a similar process in which a random timer is set. For both the requests and the repairs, the timer is a function of the closeness between the receiver in need of the packet and the receiver transmitting the repair. Although the use of local repairs relieves the NAK implosion problem at the sender, there are no limitations upon traffic flows within the group. The absence of limitations could potentially lead to the inefficient use of network resources in some localities [15].

The server-based approach makes use of specially designated hosts called repair servers to perform local recovery. Only these repair servers can send retransmissions during the local recovery phase. An example of the server-based approach is the Reliable Multicast Transport Protocol (RMTP) [22]. This protocol increases scalability by creating a hierarchy that enables Designated Receivers to collect feedback messages and to provide any available repairs to nodes within its local domain. Using the global multicast tree, the sender sends every packet over the multicast tree to the entire multicast group. Rather than sending its status information directly to the source, receivers send this information to their corresponding DR over a local multicast tree. A DR does not consolidate the feedback, but rather sends its own status information to the source. [22]

The generic reliable multicast version of the receiver-based protocol presented in [10] performs ARQ error recovery in a DER grouped architecture. The source sends all transmissions over a multicast tree. When a receiver detects a loss, it performs the local retransmission phase. During this local retransmission phase, the receiver waits a random amount of time prior to sending its own NAK. In this time interval, if no other receiver's NAKs are received, then a NAK is multicast to its local neighborhood peers. When sending a NAK, the receiver sets a local retransmission timer. This timer determines the interval that the receiver waits for the request to be locally fulfilled. Upon receiving a local packet retransmission request that can be fulfilled, the receiver multicasts the packet to the entire neighborhood. However, prior to sending this packet, the receiver waits for a random amount of time and suppresses its own transmission if another "requested" packet is received. If no members in the local neighborhood respond prior to the expiration of the local retransmission timer, the receiver begins the global recovery phase. In this phase, the receiver transmits a global NAK over the entire multicast tree using a NAK suppression mechanism. If the requested packet is not received, then the receiver enters another local recovery phase. This process of alternating between local and global retransmissions repeats until the receiver has correctly received all desired packets. When the source receives a global NAK, it remulticasts the packet to the entire group. This process is followed for all packets multicast from the source to the multicast group.

The generic version of a sender-based protocol presented in [10] performs ARQ error recovery in a DER grouped architecture. This protocol uses repair servers to process the retransmission requests of receivers in the local neighborhood. Initially, the source

sends a packet to all receivers and repair servers via a multicast tree. When a receiver detects a loss, it employs a NAK suppression mechanism to reduce the amount of feedback. However, rather than multicasting a NAK to the entire group, the receiver only multicasts a NAK to its repair server and to the receivers within its local neighborhood. A local neighborhood is defined as the set of receivers on the same subtree rooted at the nearest backbone router (see Figure 2.2). When a repair server receives a NAK for a particular packet from a member of its group, it determines the availability of the packet. If the repair server has the packet, then it is multicast to the local neighborhood. Otherwise, the repair server must retrieve the packet from the source or its upstream repair server. This retrieval is accomplished using the same NAK suppression mechanism that was used by the receivers to retrieve the packets from the repair server. When the source receives a NAK, it remulticasts the packet to all receivers and repair servers. This process repeats for each packet multicast from the source to the receivers.

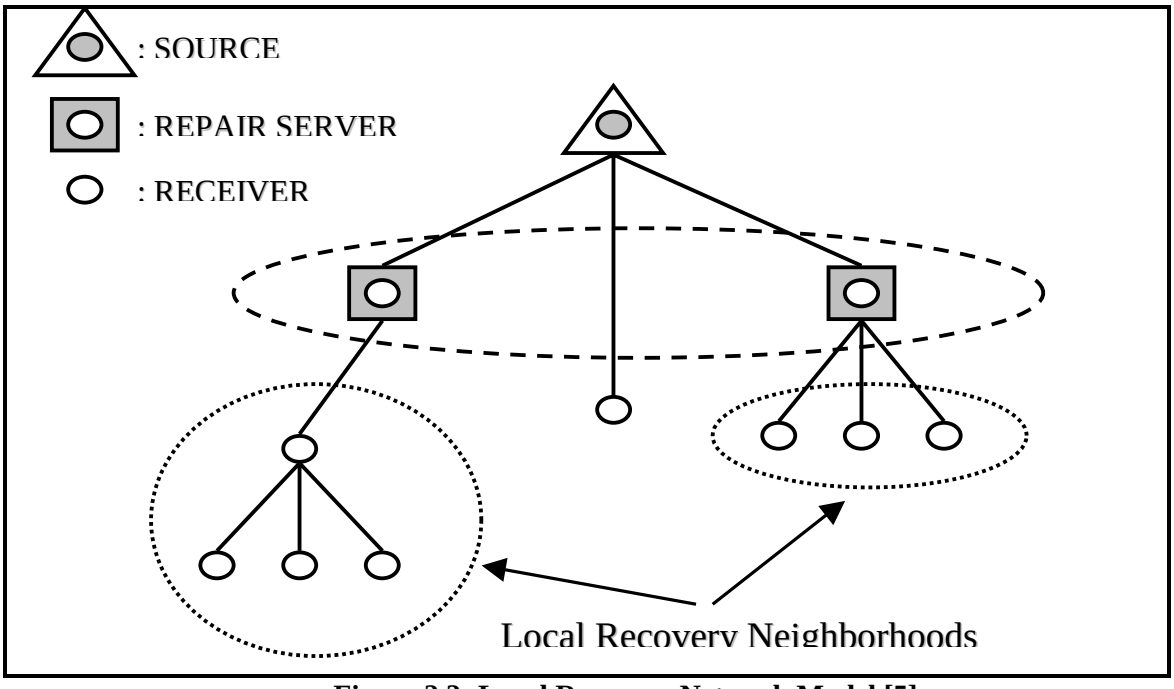


Figure 2.2: Local Recovery Network Model [5]

Per the analysis completed in [10], the server-based approach yielded higher protocol throughput and lower bandwidth usage than the receiver-based one. However, these results required that the repair servers have processing power slightly higher than that of a receiver and several hundred kilobytes of buffer space available per multicast session.

2.1.2.2.2 Local Recovery and Hybrid Error Control Approaches

Now that ARQ mechanisms have been studied with both CER and DER architectures, a discussion of hybrid error control within DER architectures is needed. As was previously discussed, the retransmission of parity has excellent scaling properties as a single parity packet can repair different losses at different receivers. In CER architectures, it was shown that using integrated hybrid error control the number of retransmissions was significantly reduced [17]. Nonnenmacher et al. [18] analyzed integrated hybrid error control within the DER architecture and compared performance with the hybrid error control within the CER architecture.

In the analysis of [17], one CER-based and two DER-based protocols were analyzed. The CER protocol, called C, is based upon the integrated hybrid error control model shown in Section 2.2.1.3. The two DER-based protocols conform to the DER grouped architecture. One of the DER group based protocols, D1, uses an ARQ mechanism. In this protocol, the source is considered the repair server for all internal DER nodes. These internal DER nodes are group leaders for all receivers that are their children. The first transmission is multicast to all receivers, but subsequent retransmissions are handled locally. This protocol basically works in a store-and-forward

manner requiring that all data be received by all internal DER nodes on the first tree level. Then, the data will be forwarded in parallel from all DER nodes to their corresponding receivers. The other DER grouped based protocol, D2, is a variation of the D1 protocol. Rather than using ARQ as an error recovery mechanism, D2 uses integrated HEC. Protocol D2 transmits packets in the same manner used in D1. However, error recovery at both levels uses parity retransmissions rather than an ARQ mechanism.

A bandwidth analysis of these three protocols shows that D2 outperforms D1. This increased performance can be directly attributed to the use of parity packets to repair losses. If the transmission group size (i.e. – number of data packets), k, is large enough, then C also outperforms D1 due to the efficiency use of parity packets. For example, assume that the TG size is relatively large. Each of R receivers requires a different packet to correctly receive a TG. In protocol C only one packet would have to be transmitted by the source, whereas in D1 R packets need to be transmitted from internal DERs. Grouped DER protocols have better scalability than CER protocols due to their hierarchical structure, which limits the scope of retransmissions. When increasing the block size (e.g. increasing k), the performance gain of the D2 protocol over the CER protocol decreases. Additionally, there are some weaknesses in using such a hierarchy for error recovery. In flat network architectures (such as with many satellite networks) there is no place for internal DER nodes because receivers tend to be directly connected to the satellite. In this thesis we are considering this type of flat hierarchy networks so the advantages of hierarchical error control architectures that are based on internal DER nodes disappear.

Bandwidth analyses have shown that local recovery coupled with hybrid error control (protocol D2) performs more efficiently than both local recovery without hybrid

error (protocol D1) control and HEC without local recovery (protocol C). Since HEC appears to be useful in reducing bandwidth usage, it follows that it is interesting to analyze situations in which differing levels of FEC encoding/decoding capabilities are placed at repair servers within the multicast tree. Protocols using this type of capability are known as active parity encoding services (APES) [25]. APES protocols send FEC-based repairs rather than retransmissions. APES protocols can be classified into three generic types:

1) The Store-Data-Build-Repairs (SDBR) Protocol: Once a repair server reliably obtains k source packets, it reproduces the k original data packets that are subsequently buffered. Whenever an additional repair is required, the repair server generates a new distinct repair via FEC encoding. Since these repairs are distinct, any receiver that needs additional repairs can use any combination of *k* original data packets and repairs obtained from the repair server.
2) The Build-Repairs-Store-Repairs (BRSR) Protocol: A repair server decides in advance upon a fixed number of repairs, b, per block to generate via FEC encoding. Once these b parity packets are buffered at the repair server, the remaining data packets are flushed from the cache. If the receivers within a given repair server's repair domain need parity packets, then these packets are reliably transmitted to each of the receivers. By requiring reliable transmission of parity packets (via some mechanism such as ARQ), the repair servers do not have to retain the data so that new parity can be created.
3) The Get-Repairs-Store-Repairs (GRSR) Protocol: Rather than generating the repairs, this protocol requires that the repair servers request b parity packets from

the sender. Once these b parity packets are received and buffered, this protocol behaves exactly like BRSR.

When analyzing the APES protocols, the general network model shown in Figure 2.2 was essentially used in [25]. Each of the repair servers was responsible for their respective repair domains (or local neighborhoods). In the analysis, the assumption that there was no loss between the source and the repair servers was made. The following two assumptions were also made: 1) a TG of size *k* data packets and 2) receivers lose any packet sent to it with probability p. Each repair domain was analyzed separately.

The bandwidth analysis was sub-divided into two parts, the bandwidth used between the source and repair servers, and the bandwidth used between the repair servers and the receivers. With the above assumption of error free transmission between source and repair servers, the difference in the bandwidth over these error-free links is due solely to the design of the protocols. In BRSR and GRSR, a repair server must retrieve needed repairs from the source if the number of errors experienced within its repair domain exceeds b. For sufficiently large values of b, this required bandwidth is negligible. BRSR and GRSR do not use substantially more bandwidth between the repair server and receivers than SDBR for reasonable packet loss rates (p=0.01, p=0.05). Since SDBR uses the minimal number of distinct repairs to provide reliability, it provides a lower bound on the expected bandwidth for BRSR and GRSR. However, for domain sizes and loss rates that one might expect in reality, the difference in bandwidth is negligible. The bandwidth used throughout the network in a CER hybrid error control approach is dominated by the bandwidth required by domains with high loss. Conversely, in networks with repair

servers, this bandwidth consumption can be limited to the domains where it is required rather than penalizing the performance of the entire network.

2.2 New Proposed Control Mechanisms

In the previous paragraphs we gave an overview of the error control techniques that are currently used in the design of reliable multicasting protocols. As we mentioned above, the evaluation of the protocols whose operation is based on error control mechanisms like FEC, ARQ and error control architectures like local recovery, is done in environment were the satellite link is used as the backbone of disseminating information to a large number of users through some satellite reception enable network entities, and is assumed error-free. The last assumption, even though it simplifies the study of the reliable multicasting protocols, does not reflect the reality. The satellite links are high BER links and they are characterized by the large propagation delay. Under the truthfulness of the last statement, and based on the characteristics of the above mentioned error control architectures, we expect that protocols which behave very well in network architectures consisted mainly from wireless 802.11 networks, may fail over satellite links because of the combination of high BER and large propagation delay characteristics. One more characteristic about the satellite networks is that they are flat hierarchy networks (e.g., there is no hierarchy involved in the connection between the participating network entities), so error control architectures based on Dedicated Receivers (DRs) and local recovery (in the case where the group members are only connected through the satellite and not via another connection of terrestrial form).

Based on the last's paragraph discussion, it is obvious that there is no much consideration of designing and evaluating network protocols over satellite links. Because of the propagation delay and high BER, we need to introduce an error control architecture, which acts proactively in the recovery of erroneous or corrupted packets. FEC and ARQ are error control mechanisms that still can be used as the building blocks of the new error control architecture, especially the former one (e.g. FEC) due its pro-activeness.

In this thesis we introduce such an error control mechanism, which is called Air Caching. Air Caching is not a new term, and has been proposed by Stathatos [] for the update of database caches in satellite based networks. Actually, Air Cache is used for the fast delivery of the most wanted information. By continuous broadcasting of the most wanted information, the receivers can access it with the minimum average delay, since the information is on the air before even requesting it. We are going to transfer the notion of the Air Caching in the case of reliable multicasting protocols for flat hierarchy networks as the satellite ones are.

A formal definition of Air Cache is given in the following paragraphs along with the general ideas of how we can use FEC and ARQ along with this newly proposed error control architecture.

2.2.1 Air Caching

We can characterize Air Caching as a continuous broadcast or continuous push of data. The immediate result of having data continuously broadcasted, is that these data can be accessed from the end hosts more frequently and with less average end-to-end latency

compared to the data that do not belong to the Air Cache. Using the latter description of Air Caching we can easily find out why it is called like that and that is because it acts like a cache. The notion of a cache as it appears in computer engineering, is a very fast memory that contains the most popular data. The speed of the cache results in low latency access on its contents. In the case of Air Caching the latter is the basic characteristic, because the data that are continuously broadcasted, can proactively serve a request from any end host.

Let us assume that we broadcast data with period T over a channel of bandwidth B. This broadcast can be considered to form a memory space of size B×T with some special characteristics:

- It can be accessed by any number of clients concurrently, i.e. there is no access contention.
- It can be accessed only sequentially. A direct consequence is that the average access time depends on the size of the memory, which in turn is determined by the period T.
- The server cannot have any information about the effectiveness of this memory space, i.e. which –if any- clients actually use it.

The notion of Air Cache has not been used before in designing a reliable multicasting protocol, and as we are going to show there a significant boost in the performance of the reliable protocol by using slightly more bandwidth. Extending our study to the case of Adaptive Air Cache we can see that using the same amount of bandwidth as in the case without the Air Cache, we can have again significant increase in the performance by paying more in protocol's complexity this time.

2.2.2 Air Caching and Forward Error Correction

A question that arises is how we use the existing error control mechanisms like FEC and ARQ in combination with air caching in order to design effective and efficient protocols for flat hierarchy networks. A brief overview of the answer is given in these paragraphs and a more detailed presentation is given in the next chapter were we give an overview of the functionality of the proposed reliable multicasting protocols.

How FEC is used in combination with Air Caching? The answer is somehow fairly obvious. In the Air Cache we continuously transmit parity packets of the TG (e.g. data packets seeking reliable delivery to a multicast group). The question now is how this could work successfully and boost the performance of a reliable multicasting protocol that is built based on this combination. We answer this question in very general terms here by giving an example. Assume that the TG consists of 20 data packets. We already know that each parity packet can make up for any corrupted data packet in combination with the correctly received data packets. So, assume that a member of the multicast group receives correctly 18 data packets, and the other 2 have been corrupted. Then this member of the multicast group can access fast the Air Cache because is continuously transmitted and get 2 different parity packets, then the TG can be reconstructed without the receiver having to request the 2 corrupted data packets and then receive the parity or the corresponding data packets from the receiver. Imagine what the latency gain will be, since we do not have to access the satellite link multiple times and be penalized with the high propagation delay.

The above is the general idea of how the Air Cache can use the FEC error control technique for the recovery and correct delivery of a number of data packets (TG). More details are given in the next chapter.

2.2.3 Air Caching and Automatic Repeat Request

In accordance with the questions that we arise in the previous paragraph, we ask how Air Caching can use ARQ as an error control mechanism and how this works. We intend to answer those questions in the same fashion as before since the general idea is not changing, even though we change the error control mechanism from FEC to ARQ. In the case of using ARQ as error control mechanism on reliable multicasting scheme that is based on Air Caching, we continuously transmit data packets of the TG. So the Air Cache in this case consists of data packets of the Transmission Group of packets that we want to deliver reliably to a multicast group. In order to get an idea of how this works, we give the following example, as we did in the case of FEC. Assume that the TG consists of 20 data packets that need to be delivered reliable to the participants of a multicast group. When a member of this multicast group receives 18 of the data packets correctly and the rest 2 packets of the TG have been corrupted, then this receiver instead of sending request for retransmissions through the satellite link, which are costly in terms of latency, can check the Air Cache to examine if the 2 corrupted packets exist there. If there exist then the receiver does not have to issue requests for retransmission. Evaluating this general idea, is obvious that if we choose in a clever way the contents of the Air Cache then the improvement in performance will be significant compared to just issuing request for retransmissions and going again through the high latency satellite link.

More details of the functionality of Air Cache will be given in the next chapter where we get the general ideas of Air Cache operation combined with ARQ and FEC and we build reliable multicasting protocols that present very promising performance characteristics. Those promising performance results are not located only in the end-to-end latency of those protocols but also in the scalability, robustness and efficiency characteristics of the protocols that combine Air Caching with existing error control techniques like ARQ and FEC. In the next chapter we give a brief overview of the functionality of the proposed reliable multicasting protocols, along with the metrics that we chose to use for the evaluation of them, and also we present in some extend the simulator that we built and used for simulating the functionality of the proposed protocols in order to evaluate the various aspects of their performance behavior.

Chapter 3: Overview of the Air Cache RM Protocols and Metrics

3.1 Introduction

In this chapter we give an overview of the classes of the proposed protocols, their basic characteristics, the techniques that are based on and a brief description of their functionality. Apart from this, we will demonstrate the characteristics of the simulator that we designed and used for the evaluation of the proposed reliable multicasting protocols.

This chapter is useful for the classification of the protocols that we designed, based on characteristics like the combination of the techniques that were used for the designed of these protocols, the requirements of the receiving entities in terms of hardware and processing power and the performance that we expect to achieve using these protocols.

3.2 Protocols

The basic techniques that were used for the design of the reliable multicasting protocols are three as we described then in the previous chapter. Those are:

- Automatic Repeat Request (ARQ)
- Forward Error Correction (FEC)
- Air Caching

We gave the details of the above techniques in chapter 2 of this thesis. As we mentioned there, the innovation of the proposed protocols is based on the air caching technique,

which in combination with the other two techniques, that have already been used in the designing of successful reliable multicasting protocols in the past, boosts significantly the performance of reliable multicasting protocols as we are going to demonstrate in the next two chapters.

In the following two paragraphs we will sketch out the principals of the protocols' operation, before we present them in detail along with their performance evaluation in the following chapters. Below, we focus on the basic characteristics of the protocols like on what combination of techniques their functionality is based, what are the performance goals that we set for each of the protocols, and in what environments are we intended to use them based on characteristics, like hardware and processing requirements of the receivers, end-to-end latency and others.

Each one of the following two paragraphs presents a class of the proposed reliable multicasting protocols. So, we defined two classes of protocols based on the adaptiveness of the Air Cache. The Air Cache as we presented earlier in this thesis, is a dedicated bandwidth for the continuous retransmission of packets that are needed for recovery of corrupted or erroneous packets that belong to the data packets that we want to deliver reliably to the multicast group. The two parameters that can be adapted in terms of the Air Cache is the size of the Air Cache and the content of the Air Cache in some cases. The later comment refers to the fact that if the Air Cache accommodates parity packets then each one of these has the same effect to the receivers, since each one can correct any of the erroneous or corrupted data packets in combination with the correctly received data packets. So, we cannot adapt the content of the Air Cache in this case and we are interested only in the number of discrete erroneous or corrupted data packets from the TG

(transmission group). Adaptation in content can happen only when the Air Cache accommodates data packets, so intuitively the most requested packets exist in the Air Cache so we adapt the content based on the feedback from the receivers. We will visit different combinations of the size and content adaptation when we present individually the proposed protocols. So, following the above discussion the two classes of protocols that we propose in this thesis are the Non-adaptive and Adaptive classes. In the former belong the protocols that do not present any adaptation neither in size nor in content, and in the latter belong the protocols where the Air Cache is adapted either in size or in content or even in both.

3.2.1 Non-Adaptive Protocols

Initially, we will give a brief overview of the non-adaptive reliable multicasting protocols. In those protocols there is no adaptation of the Air Cache in size or content, so there is need for feedback information from the receivers that belong to the multicast group. The latter statement arises from the fact that in order to adapt the Air Cache we need feedback from the receivers about the status of the transmission. In the case that we do not adapt any of the parameters of the Air Cache then we do not need to get any feedback since there is of no use to the class of the protocols that do not belong to the non-adaptive Air Cache protocols. What we expect from this kind of protocols in terms of performance and hardware and processing requirements, is given at the end of this section, following the presentation of the non-adaptive protocols.

In the non-adaptive class of reliable multicasting protocols belong the three following protocols:

- UDPAC (Unchanged Data Packets in the Air Cache)
- RDPAC (Replace Data Packets in Air Cache)
- PPAC (Parity Packets in the Air Cache)

Each of these three protocols is named after its Air Cache characteristics, like the type of packets contained in Air Cache (e.g., data packets or parity packets) and the refreshing rate of the Air Cache (e.g., per TGTR[1] and ACTR[2]). A visual presentation of those parameters (e.g., TGTR and ACTR) is given in the following figure:

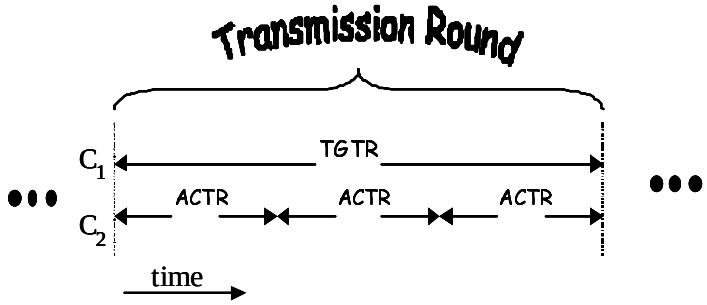


Figure 3.1: Visual description of the TGTR and ACTR parameters

After the presentation of the naming resolution of the protocols we can get an idea about the characteristics of the Air Cache for each one of the non-adaptive protocols. In the following paragraphs we will go through those characteristics along with others, in order to get an idea of the functionality of the proposed reliable multicasting protocols.

[1] TGTR: Transmission Group Transmission Round – is the required time units for the transmission of packets that constitute the TG (e.g., the data packets that we want to deliver reliably to the multicast group).

[2] ACTR: Air Cache Transmission Round – is the required time units for the transmission of the packets that constitute the Air Cache. TGTR is a multiple of ACTR, so a single TGTR could equal more than one ACTR, so during the completition of TGTR the same packets in the air cache can be retransmitted multiple times.

UDPAC (Unchanged Data Packets in Air Cache)

The functionality of the protocol is based on the fact that we have two transmission channels C_1 and C_2. The basic transmission takes place in C_1 where the data packets[3] that we want to deliver reliably to the multicast group are transmitted there. The novelty of the proposed protocols comes with the introduction of C_2, which serves as the Air Cache of the transmission. In channel C_2 are transmitted the packets that constitute the Air Cache. Until the reliable delivery of the data packets that constitute the TG, both transmissions (e.g., in C_1 and in C_2) repeat, in C_1 we have a continuous transmission of the TG and in C_2 we have a continuous transmission of the Air Cache which constitutes of packets that will correct corrupted or erroneous packets at the receiving end As we mentioned above, by decoding the name of this protocols we can exclude some of its characteristics. So, in this case for example the name of the protocol (e.g., UDPAC) reveals two characteristics of the Air Cache. The first one is about the type of packets that constitute the Air Cache, which are data packets (e.g., DPAC – Data Packets in Air Cache). Those data packets are a subset of the data packets of the transmission group that are transmitted in C_1. Each of these packets in the Air Cache can correct only the corresponding data packet, which has been corrupted during the transmission in channel C_1. The second characteristic, which is also revealed from the protocol's name, is about the refreshing rate of the Air Cache's contents (e.g., U - Unchanged). By unchanged we mean that the contents of the Air Cache do not change throughout the TGTR, but in each TGTR the Air Cache is refreshed with a new set of data packets, which may or may not contain data packets that were existed in the previous set of Air Cache's data packets.

[3] This group of data packets is called Transmission Group (TG)

The question that arises is how the Air Cache is filled in each TGTR, and the answer to this is that the Air Cache is filled with randomly chosen data packets from the TG.

As we can observe from the above brief description of the UDPAC protocol, there is no mention of any adaptation in the size of the Air Cache or any dependence of the Air Cache's update, on the feedback from the receiving entities.Those two observations characterize the protocols as a non-adaptive one, since the size of the Air Cache remains constant throughout the transmission and the Air Cache is filled with randomly chosen data packets. Further details on the functionality of this protocol and a more detailed description of its algorithm is given in the next chapter of this thesis where we present the reliable multicasting protocols that belong to the non-adaptive class of protocols.

To wrap up this brief description of the functionality of UDPAC protocol we give the following schematic that demonstrates the basic characteristics of UDPAC in a graphical way:

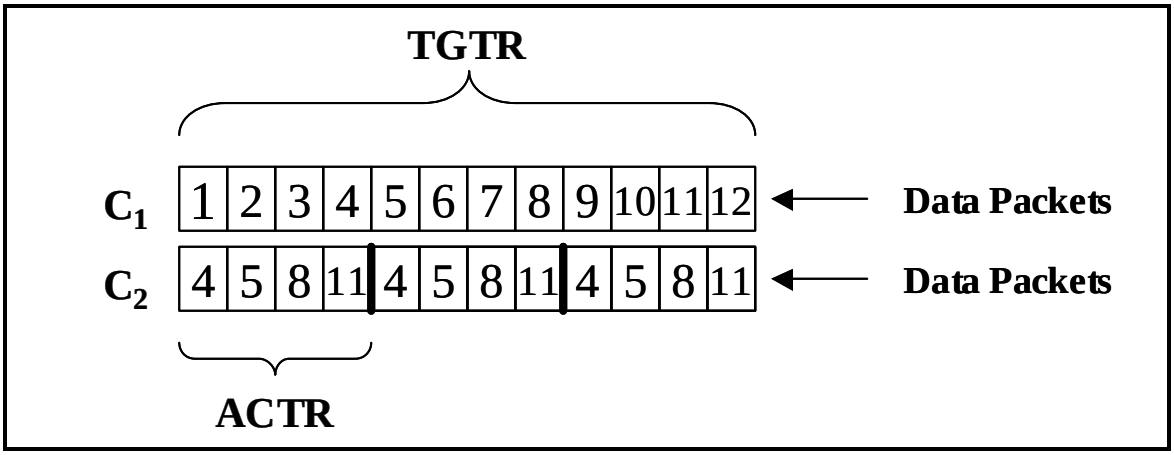


Figure 3.2: Example of UDPAC's Operation

Obviously, the Air Cache remains unchanged throughout the TGTR period. After that period the Air Cache is refreshed with randomly chosen packets from the TG, unless the transmission has been completed reliably for all the members of the multicast group.

RDPAC (Replace Data Packets in Air Cache)

Following the above discussion about UDPAC, we will give a brief overview of RDPAC protocol, whose functionality does not differ much from UDPAC. Again, the basic ingredient of this protocol's operation is the transmission of packets into two dedicated channels, the so-called C_1 and C_2 channels. The transmission on the former channel involves the transmission of the TG (e.g., the group of data packets that we want to deliver reliable to the members of the multicast group) and the transmission on the second channel, which is referred also as Air Cache involves the transmission of packets that will help on the recovery of corrupted or erroneous packets at the receiving ends.

The operation of this protocol has the same characteristics as the one described above with a slight difference on the refreshing rate of the Air Cache, that is why the current protocol is called RDPAC (Replace Data Packets in Air Cache). The Air Cache contains data packets that are randomly chosen from the data packets of the TG and is refreshed per ACTR (e.g., Air Cache Transmission Round). This refreshing rate is equal or larger than the refreshing rate that is employed on the Air Cache of the UDPAC protocol. Even though the UDPAC and RDPAC protocols seem to be the same, this is not the case, because the difference that they have in the refreshing rate of the Air Cache reflects on the performance of the protocols as will explain in the following paragraphs and we will present in the next chapter.

A schematic example of how the RDPAC operates per TGTR follows:

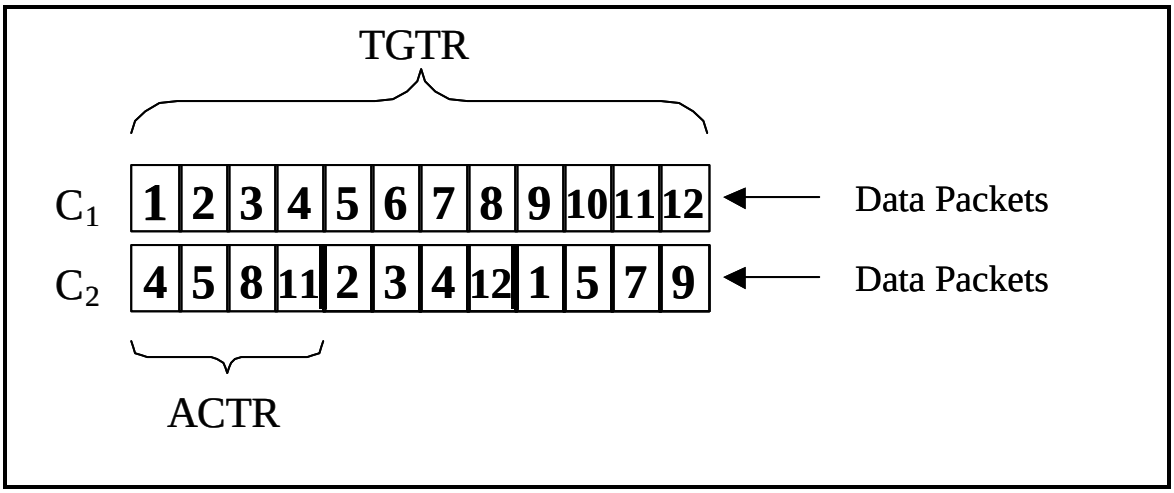


Figure 3.3: Example of RDPAC's Operation

In comparison with previous corresponding figure 3.2, that represents the operation of UDPAC in a single TGTR, we can observe that the Air Cache in this case is refreshed per ACTR with new data packets, and not per TGTR as it happens in the case of UDPAC. What are the performance expectations we have from this protocol and what are the hardware and processing requirements of it, is presented after the presentation of the last reliable multicasting protocol that belongs to the class of the non-adaptive ones.

PPAC (Parity Packets in Air Cache)

In this paragraph we give a brief description of the characteristics that define the PPAC protocol, which is the third one that belongs to the class of non-adaptive reliable multicasting protocols. This protocol differs from the two previously presented because the principals of its operation are quite different from UDPAC and RDPAC, and we can get an idea by taking into consideration the naming conventions. As the name of PPAC (Parity Packets in Air Cache) reveals, the Air Cache contains Parity Packets in Air Cache.

The major difference is that PPAC protocol is not based on the ARQ error recovery technique as the previous two protocols do, but instead it is based on the FEC technique,

which has been proven quite successful in the design of performance efficient reliable multicasting protocols. The FEC technique is involved in the design of the reliable multicasting protocols by using parity packets in the Air Cache. In the case of PPAC each one of the parity packets can correct any corrupted or erroneous data packet from the TG, by combining it with the error-free received data packets in any of the receivers of the multicast group, so there is no question of how we chose the packets that will constitute the Air Cache in each TGTR, because all the parity packets are equivalent in terms of their error recovery potential, in any data packet error.

So, wrapping up the basic characteristics of the protocol, PPAC uses two dedicated channels as UDPAC and RDPAC do. In the first channel the transmission of the TG happens in a continuous and periodic form, until the completition of the reliable transmission. The second channel forms again as the Air Cache of the PPAC, which contains parity packets that are continuously transmitted for the recovery of erroneous or corrupted data packets in each one of the receivers. The contents of C_2 differentiate the PPAC protocol from the other two non-adaptive reliable multicasting protocols. All of the three non-adaptive protocols, included PPAC do not need receiver feedback for their correct functionality, since they do not use it for adapting the Air Cache (e.g., we can adapt the size and/or content of the Air Cache) or for choosing the data packets to be included in the Air Cache (e.g., in the case of RDPAC and UDPAC protocols).

The algorithmic details of the PPAC protocol are given in the following chapter of this thesis. A schematic representation of the PPAC's operational characteristics in a single TGTR follows:

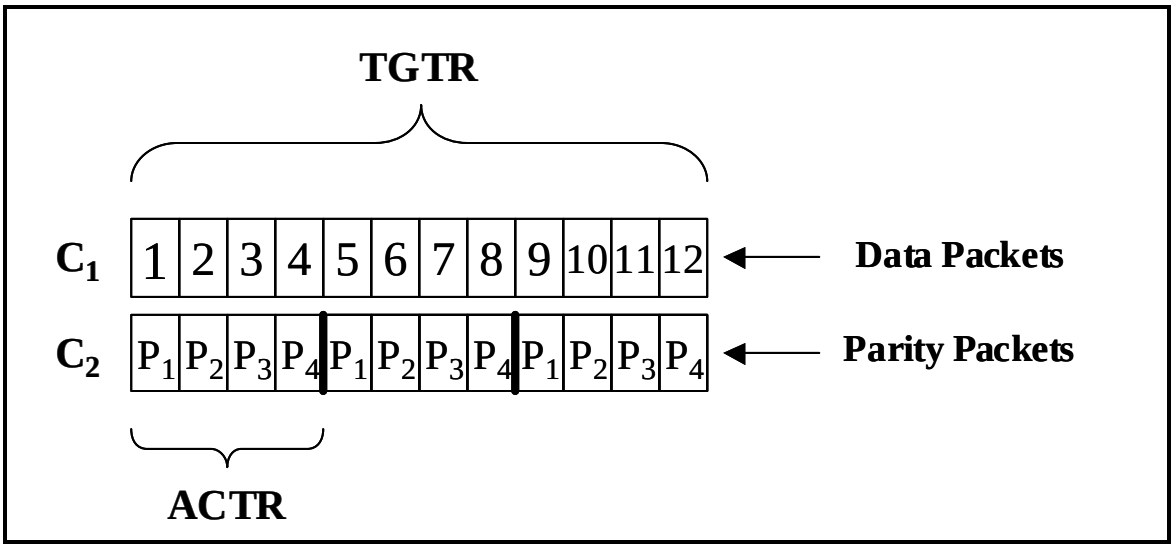


Figure 3.4: Example of PPAC's Operation

The performance expectations that we have from this protocol along with the hardware and processing requirements that this protocol has, are given in the following paragraphs, where we make an effort to guess what the pros and cons of the non-adaptive protocols are, based on their operational characteristics that we demonstrated above.

3.2.1.1 Performance Expectations

In this paragraph we make an effort to present what are the expectations that we have from the non-adaptive reliable multicasting protocols. These expectations originate from the recovery methods that each one of the above protocols is based upon and the operational characteristics that these protocols have.

We will visit each one of the proposed protocols individually by making some comments about what we expect from the performance behavior of each one of these (e.g., UDPAC, RDPAC, PPAC).

UDPAC (Unchanged Data Packets in Air Cache)

The UDPAC protocol uses the ARQ error recovery method for the correction of the corrupted or erroneous data packets at the receiving ends and it does not use FEC, which has been proven a very successful technique for the design of the reliable multicasting protocols. There is a reason for the latter choice that we will discuss in the next paragraph about the hardware and processing requirements that UDPAC protocol has for its correct functionality.

It is expected that due to the fact that UDPAC does not uses FEC and is relying only on ARQ and Air Cache for error recovery, the protocol's performance will not be efficient compared to the performance of the existing protocols that use FEC as base for error recovery. On the other hand, we expect that this protocol will present better performance than the protocols that are fully based on ARQ for error recovery, due to the utilization of Air Cache as a proactive mechanism for error recovery for the reliable transmission of data. Of course, the real performance gain compared to the ARQ based protocols is going to be revealed if we take into consideration the fact that UDPAC uses more bandwidth (e.g., due to the usage of two transmission channels or in other words due to the introduction of Air Cache in the protocol's operational characteristics). We will find out about it in the next chapter where all the above facts are proven through simulation results and we obtain also an answer about the real performance gain compared to a non Air Cache ARQ-based protocol by using a quantity that we call Normalized Gain, which we will introduce in the following chapter.

RDPAC (Replace Data Packets in Air Cache)

The RDPAC protocol operates almost in the same way as UDPAC. RDPAC as UDPAC is not FEC based protocol and just relies on Air Cache and ARQ for error recovery on the reliable transmission of the TG. So, the same comments that we did for UDPAC hold also for RDPAC. So, we expect that the performance of RDPAC is not going to be better than the FEC based reliable multicasting protocols, but we expect though to present a performance gain compared to the reliable multicasting protocols that are based solely on ARQ and do not use Air Caching. Again, by mentioning performance gain, we mean the real performance gain by taking also, into account the extra bandwidth that RDPAC utilizes for Air Caching. We will explore the latter by using the Normalized Gain metric, which we will introduce in the next chapter.

Furthermore, and beyond the expected performance compared to ARQ based protocols, we expect that the RDPAC will present better performance characteristics than UDPAC. The latter can be exploited by referring to the difference that those two protocols have. The difference is the refreshing rate of the Air Cache where in UDPAC happens per TGTR but in RDPAC happens per ACTR and since those two quantities are related with the following relation:

$$TGTR \geq ACTR$$

the RDPAC is expected to outperform UDPAC. The latter statement can be intuitively proven by the fact that the Air Cache of RDPAC can accommodate more data packets than UDPAC since the Air Cache is refreshed with new data packets per ACTR and not per TGTR as it happens in UDPAC (e.g., revisit the above schematics that represent the operational characteristics of UDPAC and RDPAC), so more data packets from the TG

can be recovered in a TGTR period. In the case of equality in the above relation, it is expected that both protocols will perform the same since the refreshing rate will be the same.

All of the above statements will be revisited when we analyze the performance of the protocols based on the simulation results that we collected and we will find out if our intuition is correct. In the following paragraphs we will find out why we put effort on analyzing and designing protocols (e.g., UDPAC, RDPAC) that do not present very promising performance characteristics compared to the existing FEC based reliable multicasting protocols.

PPAC (Parity Packets in Air Cache)

The UDPAC and RDPAC protocols are based solely on ARQ for error recovery, and this is the reason why we do not have great expectations in terms of their performance behavior, compared to FEC-based reliable multicasting protocols. The reason of their existence is going to be explained later. On the other hand, PPAC is a protocol that is based mainly on FEC for error recovery at the receiving ends. FEC has been proven to be a very successful in the design of very efficient and successful reliable multicasting protocols, in terms of the performance characteristics that they present. Based on the latter fact we expect that PPAC will also present very promising performance characteristics.

The question that arises at this point is how promising are those results going to be? Before, answering the above question using facts and results, we will use initially our intuition about the performance we expect that PPAC will deliver. The combination of

the techniques that PPAC utilizes and the performance expectations that we from the PPAC protocol is the source of our inspiration for the design of this protocols. As we mentioned a while ago, we expect that PPAC will perform equally well as other protocols that use FEC as their main technique for error recovery. Furthermore, and due to the introduction of air caching, which acts as a proactive and dedicated mechanism for error recovery, we expect that PPAC will present exceptional performance characteristics compared to the already existing reliable multicasting protocols. The latter intuitive statement, arises from the fact that each parity packet can compensate for any corrupted or erroneous data packet (e.g., this is the basic reason for the success of FEC technique in the class of reliable multicasting protocols) but also these parity packets can be transmitted continuously in a dedicated channel (e.g., Air Cache) and can be immediately available to any of the members of the multicast group. On the other channel the transmission of the data packets will be going uninterrupted, and will also contribute to the correction of corrupted or erroneous data packets at the receiving ends. Intuitively, from the above discussion we expect that PPAC will present nice performance characteristics.

Beyond the very promising performance characteristics that we expect from the PPAC protocol, we have to take into consideration the extra bandwidth that this protocol uses for air caching. So, we have to compare it appropriately with the existing reliable multicasting protocols. As we mentioned in the case of UDPAC and RDPAC this comparison will be done through the Normalized Gain metric, which we will introduce later in the text. As we can conclude from the discussion above, the performance of PPAC is also expected, to be much better than the other two non-adaptive Air Cache

reliable multicasting protocols (e.g., UDPAC and RDPAC). On the other hand and compared to UDPAC and RDPAC, we have to take into account the extra complexity of PPAC and the hardware and processing requirements the PPAC has compared to the former protocols. The latter will be exploited in the following paragraph, where we briefly give an overview of what are the requirements of the proposed, non-adaptive protocols based on their already mentioned operational characteristics.

3.2.1.1 Hardware and Processing Requirements

In this paragraph we will briefly describe what are the hardware and processing requirements of each one of the above proposed reliable multicasting protocols (e.g., UDPAC, RDPAC and PPAC), based on their operational characteristics and more specifically referring on the error recovery techniques on which each one of these protocols is based on.

Before we proceed, the question that arises is what we define as hardware and processing requirements. As hardware requirements we define the buffering space that is needed for the correct operation of the protocol, both in the sender and the receivers. At the receivers' side, the buffering space is required for the storage of the correctly received packets, until the reliable reception of each one of the data packets belonging to the TG. Equivalently, the memory space at the sender is used for the storage of the required packets that will be transmitted and will be used for the error recovery of corrupted or erroneous data packets at the receivers, in each transmission round that will follow the initial transmission of the TG, until the reliable reception of each one of the data packets from all the members of the multicast group.

Finally, the definition of the processing requirements that the protocols have for their correct functionality is the processing power that the receiving and transmitting entities should have in order to support the efficient and correct operation of the corresponding reliable multicasting protocol. For example the processing power is used from the sender for the processing of the packets that will be transmitted in the subsequent rounds, for the processing of the responses that collects from the receiving entities and in the case of FEC for the creation (e.g., application of encoding schemes like Reed-Solomon encoding) of the parity packets that will be transmitted for the recovery of corrupted or erroneous data packets. In the same fashion the processing power is used from the receiving entities for the processing of the receiving packets (i.e., Cyclic Redundancy Check (CRC)), for establishing the feedback messages and mainly in the case of FEC, for the decoding of parity packets for the recovery of the data packets that have been corrupted.

Since we have defined what we mean by hardware and processing requirements, we can make our comments about these requirements that each one of the proposed protocols we expect that it will have for its correct functionality and for achieving its potential performance behavior, based on its operational characteristics and the synthesis of the error recovery techniques that it combines.

UDPAC (Unchanged Packets in Air Cache)

As we already know, UDPAC is based solely on the simplest error recovery technique, the ARQ technique that is one of the main reasons why UDPAC does not present any algorithmic complexity. The latter observation provides us with the extra

information that the hardware and processing requirements are going to be minimal compared to other reliable multicasting protocols, which use FEC for error recovery, even though the expected performance behavior is not expected to be very promising compared to the latter class of protocols. Why the requirements are much less compared to the FEC-based reliable multicasting protocols, we will explain it further when we discuss about the requirements of PPAC protocol, which is based on FEC error recovery method. Actually the UDPAC protocol requires buffer space equal to the size of the number of packets that constitute the TG, because it stores only the correctly received packets and expects to receive the erroneous or corrupted from subsequent transmissions of the TG or from transmissions that involve the contents of the Air Cache. The already correctly received data packets are discarded, since they are already stored to the buffer space. Obviously, the memory requirements of this protocol do not seem to be very demanding, since there is a known bound for the buffer space required, which does not exceeds the TG size.

Furthermore and exploiting the processing requirements of the UDPAC protocol, we refer to the simplicity of the protocol and as we mentioned above, this is due to the fact that is based on a very simple technique (e.g., ARQ). So, due to the same reasons that we use to conclude that the hardware requirements of UDPAC are low, we use to conclude that the processing requirements are also low. Actually, the receiving devices can be very simple and is not required to have powerful processors. The latter statement has an impact on the cost of the devices, which is also can be low due to the low memory and processing requirements of the protocol. Equivalently, the simplicity and the low requirements of UDPAC have an impact to the size and the power consumption of the

hardware devices that handle the traffic. Those devices can be small and lightweight without putting any extra complexity in the manufacturing process. The power consumption is going to be low and the portable devices can be alive for longer amount of time. The low power consumption arises from the fact that the processing requirements are low and there is one to one relationship between those two factors (e.g., power consumption, processing power). To wrap up, UDPAC can be supported by devices, which are characterized by low memory and low processing power.

RDPAC (Replace Data Packets in Air Cache)

As we mentioned at the paragraph above, where we give a brief description of RDPAC protocol, this protocol operates almost the same way as UDPAC, with a slight difference on the refreshing rate of the Air Cache. Both protocols are based on ARQ error recovery technique, so we expect that RDPAC will have the same hardware and processing requirements as UDPAC. So, the conclusions that we stated above for UDPAC hold also for RDPAC. The hardware and processing requirements are also low in RDPAC, which can function correctly on devices that do not have large memory and processing power. Because of the algorithmic characteristics of RDPAC, the power consumption on those devices is expected to be low, and those devices can be manufactured with low cost. All those conclusions are very important in the area of sensors, which we want to be of low cost, very simple and to communicate with power efficient protocols.

So, as we mentioned in the paragraph for UDPAC, even though those two reliable multicasting protocols are expected not to be very efficient in terms of performance

compared to the protocols based on FEC, can be applied to cases where the participating devices are not so powerful. In those cases we just need a lightweight protocol to be used for the reliable multicast communications, which is expected to perform better than a protocol that is solely based on ARQ error recovery technique (e.g., non Air Cache ARQ based reliable multicasting protocol). The introduction of air caching boosts the performance of the ARQ error recovery method, so the protocols that combine those two methods perform much better than a protocol that is based only on ARQ, as we will find out in the next chapter.

PPAC (Parity Packets in Air Cache)

After exploiting the hardware and processing requirements of UDPAC and RDPAC, we will do the same for PPAC protocol. Before proceeding into this discussion, we expect that the requirements of PPAC are going to be different from the previous two protocols. The latter is based on the fact that PPAC uses mainly FEC for error recovery, which involves parity and data packets and also, encoding and decoding of packets at the sender and the receivers, respectively. Based on these basic differences that PPAC presents compared to UDPAC and RDPAC, we can reach to some very interesting and important conclusions about the memory and processing requirements of this protocol.

Obviously, PPAC seems more demanding in terms of required memory for its correct functionality compared to UDPAC and RDPAC. As we mentioned above, these increased memory requirements are due to the fact that the nodes that use PPAC need to store not only the correctly received data packets but also the correctly received parity packets, which will be used for the recovery of corrupted and erroneous data packets. Apart from

this general conclusion, we should answer the question of how much demanding PPAC seems to be compared to the memory requirements that UDPAC and RDPAC pose. Logically thinking, the memory requirements that PPAC presents to have do not differ much from the memory requirements of the other two protocols. The latter can be explained by referring to the basic property of FEC and more specifically of parity packets. As we mentioned in chapter 2 when we described the FEC technique, a parity packet can compensate for each one of the erroneous or corrupted data packets in combination with the correctly received data packets. Based on this property of parity packets, it is obvious that the receivers do not have to wait for the correct reception of all the data packets but just for a group of them, and the rest can be reconstructed from the correctly received data packets. So, from this point of view the memory requirements at the receivers do not appear to present much difference, even though it seems to be increased in the case of PPAC. What about the sender's memory requirements? In the case of PPAC and due to the fact that is based on FEC, the sender needs to store, apart from the entire TG, the parity packets that will be transmitted as part of the Air Cache, until the completition of the reliable transmission. So, the latter proves the increased memory requirements that PPAC poses to the sender, too. There is a way to override this extra burden of the sender by producing the parity packets on the fly. In other words, the sender can produce the parity packets online, so the parity packets will be produced only the time that are going to be transmitted and in this scenario there is no need for buffering those. Of course, using the latter as a solution to decrease the memory demands at the sender, creates a trade off between memory requirements and processing power, since the online production of parity packets requires powerful processors at the sender. More

details about that are given to the following paragraph, where the processing requirements of PPAC are being exploited.

Furthermore, a very important aspect is the processing requirements of the PPAC protocol for its correct operation. We will present the processing requirements both from the point of view of the receivers and from the point of view of the sender. We already had an initial discussion for the processing requirements at the sender, but we are going to elaborate more on this topic. In general, the use of Forward Error Correction (FEC) requires encoding and decoding of packets. The encoding is performed at the sender and the decoding at the receivers. So, both parties are required to have powerful processors in order not to affect the performance of the protocol. In the case of the sender, we can avoid this burden of processing power, which affects also a critical factor for the portable devices like the power consumption is. This is done by producing the parity packets offline and storing them until the time that will be needed. Of course, this would affect the memory requirements, since more memory and storage space will be required. The latter is an issue that we have already mentioned in the previous paragraph, when we were presenting the memory requirements of the sender. So, this is a trade off that could be resolved when we deal with the corresponding network scenario. For example when the sender has enough processing power and there is no problem with the power consumption, we could produce the parity packets online, so we can save buffering space, otherwise we could produce the parity packets offline and store them, until it is required to be transmitted. The latter solution will save us processing power and consumed power, which is a very important issue when the communication involves portable devices. In the case of receivers there are no such choices, as there are for the sender. That is,

because the receiver has to decode the packets and reconstruct the erroneous or corrupted packets online. The latter potential behavior of the protocol requires processors with sufficient power, which depends on the speed that we want to decode the parity packets and reconstruct the erroneous or corrupted data packets. On the other hand, the reliability is needed in cases that we do not require live stream, so the delay requirements are not so demanding, so we can afford a processor, which is not powerful but which is powerful enough to decode the parity packets in an acceptable amount of time. One other thing is that the processor's power affects the memory requirements and that is because as slow is the processor, more memory is needed for the buffering of the received packets.

Obviously, there are trade-offs that we need to go through when we design or choose the receiving devices or the technical specifications of the sender, which in our scenario is a satellite, but in the general case could be any device with broadcast capabilities.

In the next paragraph we go through the brief description of the protocols that constitute the adaptive class of reliable multicasting protocols, along with their processing and memory requirements in the same fashion that we did for the protocols of the non-adaptive class.

3.2.2 Adaptive Protocols

In this paragraph we will give a brief overview of the proposed protocols that constitute the adaptive class of reliable multicasting protocols. The parameters that can be adapted in those protocols have to do with the characteristics of the Air Cache, and namely those parameters are the size and the content. When we adapt the size of the Air Cache, we specify the number of packets that the Air Cache can accommodate per TGTR

and when we adapt the content of the Air Cache we define the type of packets that will be accommodated and also in the case of data Air Cache (e.g., the Air Cache contains data packets from the TG), we specify which packets will constitute the Air Cache.

Based on the above description, the adaptation of the Air Cache can happen into two dimensions (e.g., size and content), the different combinations of adapting or not the Air Cache towards those dimensions, specify the characteristics of the different adaptive reliable multicasting protocols. Namely, these protocols are:

- ACDAC (Adaptive Content in Data Air Cache)
- ASPAC (Adaptive Size of Parity Air Cache)
- HADAC (Hybrid Adaptation of Data Air Cache)

Obviously, the names of the proposed protocols are based on the adaptation and Air Cache type. The names of the proposed protocols reveal their basic characteristics, as it was the case for the non-adaptive reliable multicasting protocols. In the following paragraphs we will briefly, go through the principal characteristics of the proposed protocols, the error correction techniques that are being used and the performance expectations that we have from those protocols, intuitively and prior their performance analysis. Finally, and based on the characteristics of the adaptive reliable multicasting protocols, we will estimate the corresponding memory and processing requirements for the sufficient hardware support of each one of those.

Before proceeding in the brief description of the characteristics of the adaptive reliable multicasting protocols, we will make some general comments about the functionality of those protocols. Those protocols operate in the same fashion as the non-adaptive protocols. In other words, all three adaptive protocols that we are going to

propose use two dedicated transmission channels for their operation. The first channel C_1 is used for the normal transmission of the data packets that belong in the TG, and the second channel C_2, which is called Air Cache is used for the transmission of packets (e.g., data or parity packets) that are indented to recover the erroneous or corrupted data packets at the receivers. The difference as we mentioned above is the adaptation of some of the characteristics of the Air Cache, like the size and the content, based on the feedback from the members of the multicast group. Brief details of this adaptation process are revealed in the following paragraphs and complete description is given in chapter 5, where we demonstrate and analyze the proposed adaptive protocols extensively.

ACDAC (Adaptive Content in Data Air Cache)

Following the same procedure as in the case of non-adaptive reliable multicasting protocols, the naming of the adaptive protocols reflect and reveal some of the characteristics that those protocols present. The Adaptive Content Data Air Cache protocol (ACDAC), operates on two dedicated transmission channels as we mentioned above, the TG data transmission channel and the Air Cache channel, which can contain data or parity packets and can be adapted in a certain way. Using the naming resolution, we could immediately observe that the Air Cache contains data packets, and the adaptation is done in terms of the content of the Air Cache per transmission round .

Taking into consideration the already presented non-adaptive reliable multicasting protocols, we could say that the ACDAC protocol is almost the same with the UDPAC and RDPAC protocols with the added capability of adapting the content of the Air Cache.

So, everything that we mentioned about the functionality of those protocols holds too, for the ACDAC. The techniques that the ACDAC protocol combines are the ARQ error recovery technique and the air caching for boosting the error recovery process. Furthermore, we need to elaborate more on the description of adaptation process. In this paragraph, we will make some remarks but the detailed description will be given in chapter 5.

The adaptation process and in general the functionality of the ACDAC protocol is shown in the following figure:

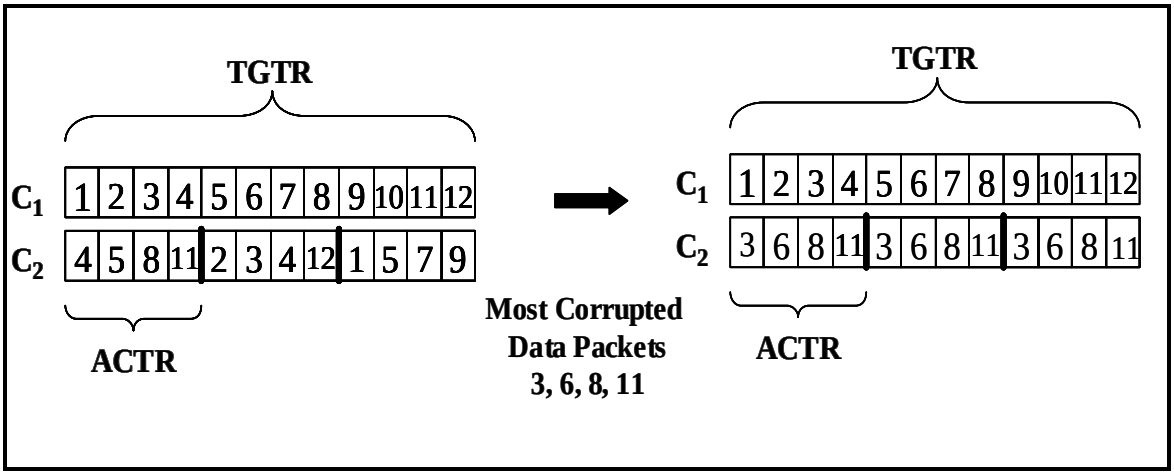


Figure 3.5: Example of ACDAC's Operation

Obviously, the general operation is similar to UDPAC and RDPAC. The adaptation process is based on the feedback from the members of the multicast group. The question that arises is how this feedback is used from the sender in order to reconstruct the content of the Air Cache. The answer to this question describes also the adaptive operation of the Air Cache. Initially, the Air Cache is filled randomly with data packets from the TG (e.g., the same way it is happening in UDPAC and RDPAC), after this initial round the sender in the subsequent rounds collects feedback, which analyzes. The feedback from each member of the multicast group contains the sequence number of packets that have been

corrupted. So, the sender can conclude, which data packets of the transmission group has been corrupted most, and those get the highest priority to be included in the Air Cache of the next round. Obviously, the adaptation procedure is simple to understand but is based on the feedback from the receivers, something that may affect the performance of the protocol, but we will revisit this in chapter 5. Another parameter of the Air Cache is the size, which in ACDAC protocol remains constant throughout the reliable transmission of the TG. After this brief description of ACDAC, the question that arises is what is the performance that we can expect from a protocol such as ACDAC, and what are the expected gain if there is any, in the performance behavior of the protocol due to the corresponding content adaptation of the Air Cache compared to its non-adaptive counterparts (e.g., UDPAC and RDPAC).

ASPAC (Adaptive Size of Parity Air Cache)

Using the same naming resolution, we named the Adaptive Size Parity Air Cache (ASPAC) protocol based on the characteristics, which distinguish it from the rest of the proposed reliable multicasting protocols. Taking into consideration the operation characteristics and functionality of the non-adaptive protocols, we can consider ASPAC very similar to the Parity Packets in Air Cache (PPAC) protocol. The difference between the two protocols is the dynamic characteristics of the ASPAC protocol, in terms of the Air Cache behavior throughout the reliable multicast transmission. We are going to refer on these dynamic characteristics of the Air Cache after we exploit the basic and similar to PPAC, characteristics of ASPAC.

The ASPAC protocol operates on two dedicated transmission channels, in the similar way compared to the rest of the proposed reliable multicasting protocols. The first channel C_1 is dedicated to the transmission of the TG data packets, and the channel C_2, which is called Air Cache is dedicated to the transmission of packets that will contribute to the recovery of corrupted or erroneous packets at the receiving ends. AS we already have mentioned, the Air Cache contain parity or data packets. In the case of ASPAC and as its name reveals, the Air Cache consists of parity packets. So, ASPAC belongs to the category of reliable multicasting protocols that combine FEC and ARQ for error recovery, along with air caching, which boosts those error recovery methods. Obviously, those general characteristics of ASPAC are similar to PPAC, which we expect to have very promising performance characteristics. The performance expectations of ASPAC are given to the following paragraphs of this chapter.

Prior to the discussion of the performance expectations that we have from ASPAC, we have to give some insight about the adaptive characteristics of the Air Cache, a more detailed description will be given in chapter 5, where we present the adaptive protocol in a very detailed way, along with their analysis and performance evaluation. Initially, we can get some insight on the operation of ASPAC by looking at the following figure that presents the functionality of the ASPAC, in very general terms:

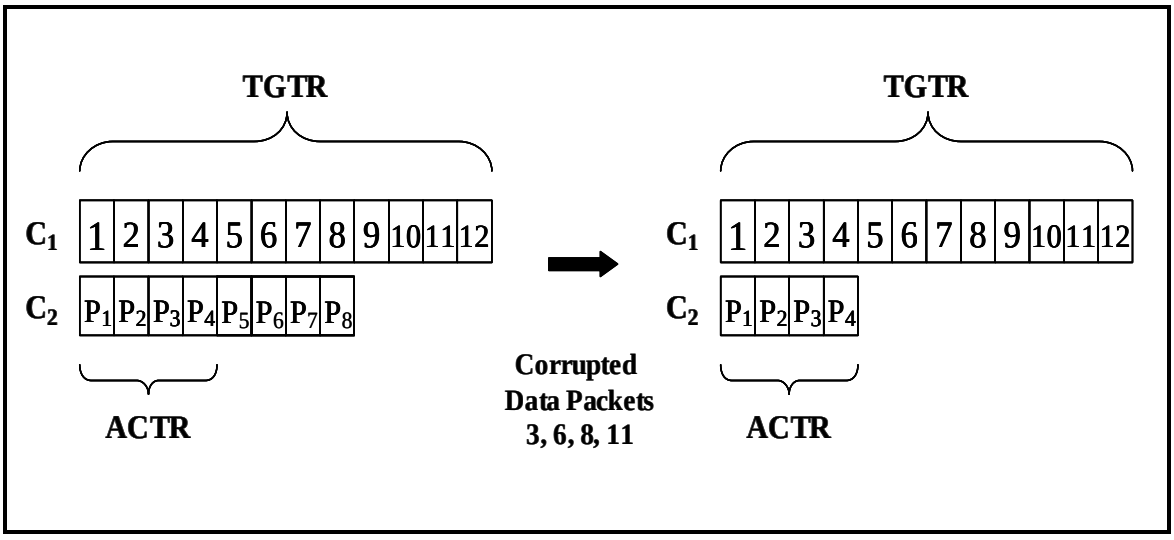


Figure 3.6: Example of ASPAC's Operation

Furthermore, as the protocol's name reveals, the Air Cache is adapted in one dimension, only in size and not in content. We cannot have adaptation in content since the Air Cache consists of parity packets, which have equivalent healing properties for any corrupted or erroneous data packet at the receiving ends. The size is adapted based on the feedback from the receivers. Initially, we dedicated a specific amount of bandwidth for the Air Cache. This amount of bandwidth is also the maximum that could be used throughout the reliable multicast transmission. The question that arises is how the size is adapted based on the feedback. The answer to this question reveals the way that the Air Cache size is adapted. Actually, when the sender receives the requests for retransmission, can estimate how many of the data packets of the TG have been corrupted the most, so based on this number the size of the Air Cache is adapted correspondingly, since we include in the Air Cache so many parity packets as the number of the most corrupted individual data packets of the TG, but never this number can exceed the maximum size of the Air Cache. Details of this adaptation process are given when we present the protocol in a more formal way, in chapter 5.

How this adaptation process affects the performance of ASPAC, and what are our performance expectations from this protocol? We already mentioned that the functionality of ASPAC is similar to PPAC, which is a protocol that is expected to have promising performance characteristics. We need to make some comments on how the adaptation in size is expected to affect further the performance behavior of the protocol compared to PPAC. We are going to discuss all of the above in the following paragraphs. Also, we will give some insight about the memory and processing requirements of the protocol.

HADAC (Hybrid Adaptation of Data Air Cache)

Finally, the third proposed reliable multicasting protocol that belongs to the adaptive class is the Hybrid Adaptive Data Air Cache (HADAC) protocol. Using the naming rules, that we have followed for each proposed protocol, we named HADAC based on the two important characteristics that it has, which distinguish it from the rest of the proposed reliable multicasting protocols. Before going through the procedure of presenting those characteristics, we will give some general insight about the functionality of Hybrid Adaptive Data Air Cache (HADAC) protocol.

Similarly, to the previously described protocols, HADAC operates on two dedicated transmission channels. The first channel C_1 is dedicated to the transmission of the data packets of the TG. The second channel C_2, which is called Air Cache, is dedicated to the transmission of packets that will be used for the recovery of corrupted or erroneous data packets at the receiving ends. In order to get an idea about the general characteristics of the operation of HADAC, based on the already existing presentations of

other reliable multicasting protocols, we can compare HADAC to the UDPAC and RDPAC protocols. HADAC presents similar operation characteristics with those two protocols from the non-adaptive class of reliable multicasting protocols. The difference between those protocols is the adaptation of the Air Cache that characterizes the operation of HADAC. Beyond the adaptation characteristics of the Air Cache, another basic characteristic is the type of packets that it carries. Based on the naming resolution, we can easily figure out that HADAC contains data packets in the Air Cache. Those data packets are chosen among the data packets that constitute the TG. How we choose the data packets that belong to the Air Cache is part of the adaptation process. Actually, the adaptation of the Air Cache in HADAC happens into two dimensions, that is why we named the protocol as Hybrid Adaptation of the Data Air Cache. The term "Hybrid Adaptation" means that we have adaptation both in size and content. Before the brief description of the adaptation process (e.g., a more detailed one exists in chapter 5, where we present the protocol extensively), we could get a general idea of the functionality of HADAC by observing the following figure:

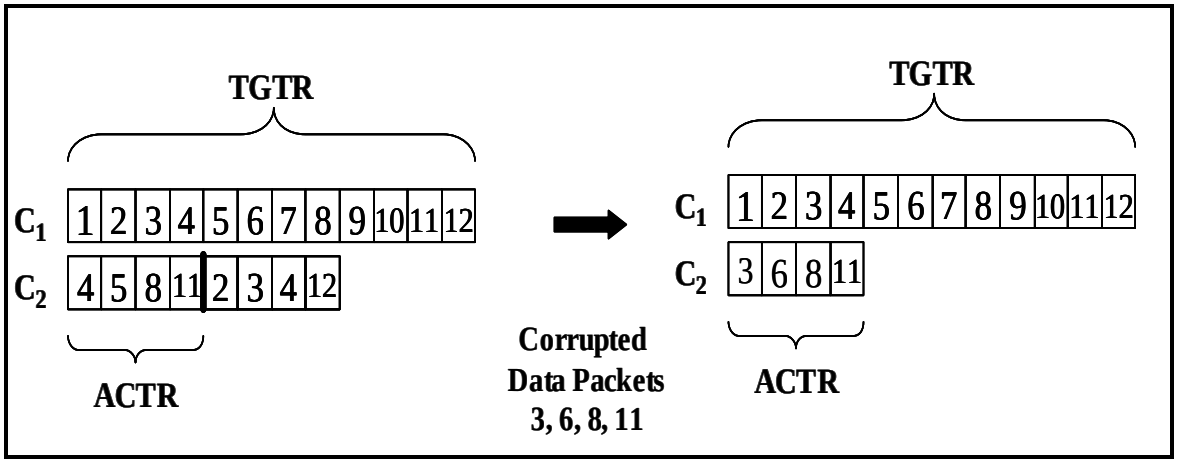


Figure 3.7: Example of HADAC's Operation

The adaptation process that is used for the Air Cache in HADAC protocol combines the adaptation processes of the previous two adaptive reliable multicasting protocols (e.g., ACDAC and ASPAC). In ACDAC we adapt the content of the Air Cache and in ASPAC we adapt the size of the Air Cache. In HADAC we adapt both the content and size of the Air Cache in the same way it is happening in ACDAC and ASPAC, respectively. Both the size and content adaptation are happening based on the feedback from the receiving ends. The content is adapted based on the number of the most requested packets (e.g., most corrupted packets from the TG). So, when the sender collects the requests for retransmission from the multicast group, estimates which of the packets of the transmission group have been corrupted the most by calculating the number of requests for each data packet. Based on this information the sender includes the most corrupted packets in the contents of the Air Cache that will be transmitted in the next transmission round. How many of those packets the Air Cache will contain, it depends on the size that will be available for air caching in the corresponding round. The available size dedicated to the Air Cache in each transmission round depends also on the feedback from the multicast group, since in HADAC the size is adapted as well as the content does. The size is adapted based on the number of packets that are required to be retransmitted in each transmission round. This number cannot exceed a maximum size that we define initially for the protocol. So, if the number of required for retransmission packets exceeds the maximum size, then we have to choose which of those packets we will include in the Air Cache, so the content adaptation process decides which of the data packets of the TG will be included. Otherwise, if the number of requested packets is less than the maximum size dedicated to Air Cache, then the size of the Air Cache for the

corresponding round is adapted accordingly. Further details for the Hybrid Adaptation process, that combines content and size adaptation are given in chapter 5, where we describe the HADAC protocol extensively.

In the following paragraphs we present briefly what we expect from the adaptive reliable multicasting protocols in terms of their performance characteristics, based on the characteristics of each one of them as they were presented above. Also, we discuss about the memory and processing characteristics that the receivers and the sender need to have in order the corresponding protocols to be sufficiently supported by the hardware of the participating devices and perform in accordance to their performance potential.

3.2.2.1 Performance Expectations

In this paragraph, following the procedure that we used for the brief presentation of the non-adaptive reliable multicasting protocols, we are going to discuss the performance expectations that we have from the adaptive protocols. We will base our statements on the operation characteristics of those protocols, which we have already presented in the previous paragraph. As we mentioned above when we were presenting the functionality of the adaptive protocols, those protocols present enough similarities with protocols that belong to the non-adaptive class of reliable multicasting protocols. The latter is going to be helpful in order to make comments on the expected performance of the adaptive protocols since we will have as a reference point the comments that we made for the expected performance of their non-adaptive counterparts. The conclusions that we will present in the following paragraphs are based solely on our intuition and the operation characteristics that those protocol have. A question that arises is what is the meaning of

presenting our intuition on the performance behavior of the adaptive Air Cache reliable multicasting protocols. The answer to this question is that we try to present what was the logic and intuition that we used for the design and proposition of each one of those protocols. The intuitive statements that we do here are going to be checked later when we evaluate and analyze the adaptive protocols, in chapter 5.

ACDAC (Adaptive Content Data Air Cache)

During the presentation of ACDAC we pointed out the similarities of this protocol with its non-adaptive counterparts (UDPAC and RDPAC). Based on this observation someone can get a general idea of the performance expectations that we have from ACDAC, since we have already made the same discussion about UDPAC and RDPAC. The way of thinking is the same as it was for both the non-adaptive protocols. The Adaptive Content Data Air Cache protocol (ACDAC) is based on ARQ for the recovery of erroneous or corrupted data packets at the receiving ends and on Air Caching for fast and proactive recovery of those packets. The protocol does not use FEC as an error recovery method, so there are no parity packets involved. This choice in the design of the protocol carries both advantages and disadvantages. The main disadvantage compared to the protocols that are based on FEC is the performance behavior of the protocol. On the other hand, we expect that the performance behavior of ACDAC is going to be better than a reliable multicasting protocol that uses ARQ but not Air Caching. The bottom line is that this protocol is expected to present promising results among the ARQ-based reliable multicasting protocols, but not among its FEC-based counterparts.

One possible question that could follow the statements of the above paragraph is why we have to study a protocol that is not going to give us nice performance results. The answer falls in the area of hardware and processing requirements that each protocol has in order to perform accordingly to its performance potential. The encoding and decoding that is required in the case of FEC could be prohibitive for machines that do not have the processing and battery power and the buffering space to perform it frequently. So, a light-weight reliable multicasting protocol will be required in this case, even though we will have to trade the performance for the low processing and battery power and the limited buffering space of the participating machines. Even if we have to go with this trade off the performance we will get, it will be better than the performance of a protocol that uses just ARQ and not Air Caching.

Furthermore, we have to exploit what are the advantages and disadvantages of the ACDAC, because of the adaptation process in the Air Cache. We will reveal our expectations compared to UDPAC and RDPAC, which belong to the non-adaptive class of reliable multicasting protocols. In the latter two protocols, the content of the Air Cache in each transmission round is chosen randomly among the data packets of the TG, so we do not have any adaptation of the Air Cache content based on feedback or other information. In UDPAC and RDPAC we solely depend on luck for getting better performance characteristics compared to reliable multicasting protocols, which rely only on ARQ for error recovery. On the other hand in the case of ACDAC, the content of the Air Cache is adapted based on the feedback from the receivers in each transmission round. The way that the Air Cache is adapted, as we briefly presented above, focuses on the most corrupted packets at the receiving ends. Obviously, we expect that the

performance of the ACDAC is going to be much better than the performance of the UDPAC and RDPAC protocols, since the adaptive Air Cache aims on the correction of the packets that have been requested the most for retransmission. The question that arises is how much better is the performance of ACDAC compared to the UDPAC and RDPAC protocols. The answer to this question is going to be given in chapter 5 were we present the performance evaluation of the protocol, where we can compare the performance difference between the protocols. The difference in performance between those protocols is going to specify the effectiveness of the ACDAC protocol, since one very important difference between the ACDAC and the RDPAC and UDPAC protocols, beyond the adaptation of the Air Cache, is that ACDAC requires collection of feedback from the receivers in each transmission round in order to use it for the adaptation process. The feedback will be delayed due to the propagation delay of the satellite link, something that we can avoid in the case of non-adaptive reliable multicasting protocols. We will discuss about it in chapter 5, where we present the simulation results.

ASPAC (Adaptive Size Parity Air Cache)

AS we have already mentioned this protocol is the adaptive counterpart of PPAC, so in terms of performance behavior we expect that ASPAC is going to have very promising characteristics. The most important characteristic of ASPAC, which is expected to boost its performance is the use of FEC as error recovery method. So, in combination with Air Caching and ARQ, the protocol is expected to have the performance characteristics that will make it a successful reliable multicasting protocol, as PPAC is expected to be.

The performance expectations that we have from the protocol seem that do not differ much from the corresponding expectations that we have for its non-adaptive counterpart. Our next step is to include in the current discussion and take under consideration the size adaptation characteristics that the protocol presents. ASPAC, as all the reliable multicasting protocols that we present here is based on Air Caching, a technique that seems promising but requires extra bandwidth (e.g., bandwidth that is not used for the transmission of the data packets, which belong to the TG) to operate. So, even though we may get a performance gain compared to reliable multicasting protocols that do not use Air Caching, due to the extra bandwidth usage, the real performance gain is expected to be less than the one calculated without taking into consideration the extra bandwidth. The real performance gain can be computed if we include in our comparison calculations the amount of bandwidth that each protocol uses. So, we expect that by adapting the size of the Air Cache in each transmission round the average Air Cache size is going to be less and the performance will hopefully, remain on the same levels, resulting in a better real performance gain compared to the non-Air Cache and the non adaptive Air Cache protocols. For ASPAC we expect that the latter will happen and by not losing in performance we can decrease the bandwidth that is dedicated to the Air Cache.

The question that arises at this point is that if the above described expected performance behavior of the protocol comes for free. The answer is no, because in the case of ASPAC the sender needs to collect feedback from the receiving points in order to collect the appropriate information for the adaptation of the Air Cache size. The requests for retransmission from the receivers to the sender will be delayed due to the high propagation delay of the satellite link, a problem that we can avoid in the case of PPAC,

which does not require the use of feedback for its correct operation since the size of the Air Cache remains constant throughout the reliable transmission of the TG. We will discuss about this implication in chapter 5, where we present the adaptive class of reliable multicasting protocols. In general we expect that ASPAC will give the better performance characteristics compared to the other proposed adaptive protocols and this is due to the combination of FEC and Air Caching.

HADAC (Hybrid Adaptation of Data Air Cache)

The last reliable multicasting protocol of the adaptive class of protocols that we will refer to, in order to discuss what are the performance expectations that we have from it based on its operation characteristics, is the Hybrid Adaptive Data Air Cache (HADAC) protocol. As we mentioned when we were briefly presenting the functionality of HADAC; RDPAC and UDPAC are its non-adaptive counterparts. There are basic similarities in the functionality of these three protocols, a fact that we will guide us in estimating intuitively the performance characteristics that HADAC will present. The difference of HADAC with the RDPAC and UDPAC protocols exists because of the adaptation characteristics of the Air Cache. The question that arises is what we gain in performance from the adaptation process in HADAC compared to RDPAC and UDPAC. Initially we will base our discussion on the similarities that HADAC has with its non-adaptive counterparts and then we will include in the discussion the adaptation characteristics that the protocol presents. Once more, we are doing this discussion because we want to present how we made the design decisions in order to evolve from the non-adaptive class of protocols and design protocols that are more efficient in terms

of performance and resource utilization, by inserting dynamic characteristics to the Air Cache.

Based on the fact that HADAC is based solely on ARQ and Air Caching, we do not expect that this protocol will present performance behavior more promising than the protocols that use FEC as error recovery method. As we have already mentioned, ARQ is not the best solution for designing reliable multicasting protocol. On the other hand if we compare HADAC protocol with protocols that use only ARQ as an error recovery method and not air caching, we expect that HADAC will present better performance behavior due to the fact that it uses air caching for overcoming as best as possible the propagation delay of the satellite link, since it transfers the packets that are required for error recovery close to the receiving ends. The latter statements have been already mentioned when we were discussing about UDPAC and RDPAC, but what is the new element that makes HADAC different from the former two protocols, and what are the performance gains that we expect due to this element? We will answer this question in the following paragraph.

The new element is the adaptation of the Air Cache in two dimensions, in size and in content. We have already discussed about ACDAC where we have only content adaptation. For ACDAC we expect that its performance will be better than its non-adaptive counterparts (e.g., UDPAC and RDPAC) due to the content adaptation. The same we expect for the HADAC due to this characteristic, but what if we compare it with ACDAC, what we get by adapting also the size of the Air Cache. The answer follows the same pattern as in the case of ASPAC and PPAC, where we want to retain the performance behavior that the PPAC is expected to have but with less bandwidth in

average dedicated to Air Cache. So, in HADAC by adapting both the content and size we expect that the performance behavior of the protocol will be as good as the performance of ACDAC with the extra advantage of using less extra bandwidth for the Air Cache in average. Still the performance is not expected to be better than the performance of a reliable multicasting protocol, which is based on FEC, but is expected to have nice performance characteristics among the protocols that just use ARQ as an error recovery method. The evaluation of these statements will happen in chapter 5, where we demonstrate and analyze the performance of the adaptive reliable multicasting protocols.

One very interesting question is why to make the effort of presenting protocols that are not expecting to have exceptional performance and are not based on techniques as effective as FEC. The answer will be given in the following paragraph where we present the memory and processing characteristics that the adaptive protocols require for their correct functionality and in order to have the ability to reach their performance potential. A preview of the answer to the latter question is that protocols that are not based on FEC are lightweight compared to protocols that are based on techniques that require encoding, decoding and excessive amount of buffering.

3.2.2.2 Hardware and Processing Requirements

In this paragraph we will follow the procedure that we followed when we were discussing the corresponding requirements of the non-adaptive reliable multicasting protocols. Because our conclusions are based solely on the techniques that each protocol utilizes (e.g., ARQ, FEC and Air Caching) and less on the adaptation procedure of the Air Cache, the discussion that follows does not differ much from the corresponding

discussion for the non-adaptive protocols. So, we will be brief by referring to the corresponding non-adaptive cases.

ACDAC (Adaptive Content Data Air Cache)

The Adaptive Content Data Air Cache (ACDAC) protocol, like the UDPAC and RDPAC protocols, is based on ARQ and Air Caching. The memory and processing requirements that ACDAC has are similar with those two non-adaptive reliable multicasting protocols.

Specifically, about the processing requirements, ACDAC does not require much processing power since does not involve any online encoding or decoding, because is not based on FEC. The only difference on the processing requirements may come from the fact that ACDAC has to process the feedback from the receivers in each transmission round, but this cannot be considered as a demanding process. The protocol is considered lightweight in terms of the required processing power at the sender and at the receiving entities. The latter conclusion has an effect on the battery power consumption, of those entities because the less power the protocol uses the slowest the power source will be drained. Also the processors cost less if they are not very powerful and the space required to accommodate slow processors is small, since the cooling of the processor is not so crucial and can be accomplished easily even in limited amount of available space. In terms of the memory requirements, the protocol requires buffering space no more than the size of the TG. So, the memory dedicated for buffering depends on how large we will build the TG.

In general, ACDAC is a lightweight protocol and can be accommodated from small and not so powerful devices, which is an important factor when we deal with scenarios

that have to do with military communications. A soldier is better to carry small devices, whose battery drains slow and they have also the privilege of reliable communications even though they may receive the information little bit delayed in the order of *ms*. The latter observation is the crucial one that gave us the reason to study protocols like the ACDAC, even though we do not expect exceptional performance results from this protocol.

ASPAC (Adaptive Size Parity Air Cache)

As we have already mentioned, ASPAC is similar to PPAC, which belongs to the non-adaptive class of reliable multicasting protocols. So, the discussion that will follow is similar to the corresponding discussion that we did about PPAC above. The effect that the adaptation process has on the memory and processing requirements is negligible, since ASPAC requires only bookkeeping of the number of packets being retransmitted.

Briefly, in terms of the processing requirements, ASPAC is more demanding than the protocols that do not depend on FEC. ASPAC is based on FEC, which requires online decoding at the receivers and probably online encoding at the sender. At the sender the procedure could take place offline but this would require more buffering space for the encoded parity packets. Obviously, this is a design trade-off, which is mainly based on the hardware specifications of the sender. For example if a sender has enough memory but does not have a powerful processing unit then the encoding is better to happen offline, otherwise if there is enough processing power available then it is better the encoding to be online, since we will encode only the appropriate number of parity packets, a number which is unknown when we encode offline. In general the ASPAC is

more demanding in processing power than the other two adaptive reliable multicasting protocols.

In terms of the memory requirements, the ASPAC protocol is more demanding again compared to the protocols that just use ARQ. That is because the transmission in this case does not involves only the data packets but also involves the parity packets as well, which are transmitted concurrently with the data packets, so at the receiving ends more buffering space is required compared to ACDAC and HADAC. From the point of view of the sender the same conclusion holds, but also the buffering space that is required may be even more, and that is because in the case where the parity packets have to be encoded offline, they have to be buffered until the end of the reliable transmission of the TG.

The general conclusion is that ASPAC is demanding in terms of processing power since it involves encoding and decoding. In terms of buffering, even though ASPAC is more demanding compared to ACDAC and HADAC, the difference is not so crucial from the point of view of the receivers. From the point of view of the sender, the difference in requirements may be significant in the case where the encoding happens offline otherwise the difference in memory requirements is small.

HADAC (Hybrid Adaptation of Data Air Cache)

Finally, as we have mentioned HADAC is based solely on ARQ and Air Caching and does not differ much from ACDAC protocol, whose requirements we presented few paragraphs above. Because we do not want to repeat the discussion, we will briefly mention, that the processing requirements of the protocol are not demanding, a fact that affects in a positive manner the power consumption and the cost of the communication

devices. In terms of the memory requirements, HADAC, as ACDAC, does not require buffering space more than the number of the data packets that constitute the TG. So, our choice of the TG size defines the buffering specifications that the protocol requires from the participating devices to meet. The fact that HADAC is not a demanding protocol in terms of the memory and processing characteristics that the participating entities need to have, makes this protocol worth investigating and evaluating.

3.3 Simulator

In the previous paragraphs we gave an overview of the non-adaptive and adaptive reliable multicasting protocols that we designed. In this paragraph we will present the characteristics of the customized simulator that we used to simulate the protocols and also we will refer to the metrics that we used to evaluate them. The results of the simulation and evaluation process are given in chapters 4 and 5 along with the detailed description of the non-adaptive and adaptive reliable multicasting protocols, respectively. In the following paragraphs we will present briefly, the design characteristics of the simulator and its functionality. Also, we will get a first look on the performance metrics that we used to evaluate the protocol.

3.3.1 Design and Functionality of the Simulator

For the simulation of the proposed protocols we did not use any existing simulator but instead we build one that has been customized on the protocols that we want to simulate and evaluate. The simulator that we built is written in C and supports networks

that are characterized from flat hierarchy. Actually, the general scenario that is simulated is that of a single sender and multiple receivers, which can be reached from the sender with a single transmission, or in other words they are one hop away from the sender. We designed our simulator like that because we want to evaluate our protocols in an environment where there is no network hierarchy, since the protocols that we propose, we claim that have been designed to perform well on such networks. Another design characteristic of the simulator is that permits the concurrent transmission of packets in two separate channels, as it happens in all the reliable multicasting protocols that we propose. As we mentioned, the first channel accommodates the normal transmission of data packets and the second channel serves as Air Cache. The same simulator can be used for the simulation of reliable multicasting protocols that use just one dedicate channel for transmissions as it happens in a large number of reliable multicasting protocols. We designed the simulator in such a way in order to be able to simulate protocols that do not belong to the set of the reliable multicasting protocols that we propose in this thesis. In such a way we can compare the different protocols based on the different metrics that we specify in the following section. In general, the simulator is customized to simulate reliable multicasting protocols in flat hierarchy networks and specifically aims to the simulation of the proposed protocols, which is the case here.

Furthermore, and beyond the scenarios that the simulator can accommodate, we need to customize the simulator for each simulated scenario. The customization process for each simulated scenario or protocol is done by assigning the appropriate values to the input parameters of the simulator. The values can be assigned from the command line and

by the time we call the simulator for execution. The input parameters that actually control the simulation scenario are given below:

- Transmission Group (TG)
- Air Cache Size
- Air Cache Rounds
- Group Size
- Packet Error Probability

A sort explanation for each of the above parameters is given in this paragraph. The Transmission Group refers to the number of data packets that we want to transmit reliable at the multicast group. The Air Cache size determines the number of packets that the Air Cache can accommodate per Transmission Group Transmission Round (TGTR). In the case of non-adaptive class of reliable multicasting protocols, this number defines the size of the Air Cache throughout the reliable transmission of the TG. In the case of the adaptive class of reliable multicasting protocols this number defines the maximum size that the Air Cache can be throughout the reliable transmission of the TG, and also this number is the size of the Air Cache at the initial round (e.g., before the adaptation process takes over and modifies appropriately the Air Cache size). The Air Cache Rounds parameter specifies how many times the contents of the Air Cache will be transmitted per Transmission Group Transmission Round (TGTR). From another point of view, Air Cache Rounds can be translated as the number of Air Cache Transmission Rounds (ACTR) (e.g., ACTR is the amount of time that is required for a single transmission of the contents of the Air Cache in C_2) that consist a single Transmission Group Transmission Round (TGTR), which is the amount of time that is required for a single

transmission of the TG data packets in C_1.The Group Size parameter defines the size of the multicast group. So, by modifying this parameter we can collect information about the scalability of each of the proposed protocol, since we can observe the behavior of the performance of each of the protocols when we increase the Group Size parameter by a constant rate, or when we assign a large value to this parameter. This parameter is very important for the evaluation of the protocols as will find out in chapters 4 and 5. Finally, one very important input parameter is ht Packet Error Probability (PEP), which actually defines the probability that a packet will be corrupted. So, by modifying this parameter between different scenarios of the same protocol we can collect information about the robustness (e.g., the ability of the protocol to retain its performance characteristics in high error probability environments) of the reliable multicasting protocol. The error model that we chose to follow in our simulation scenarios is the simplest one. Actually, we assumed that the packet error probabilities follow the uniform distribution, so the packet error probability (PEP) is the same for every transmitted packet. Even though this is a simplistic model, we can compare our proposed protocols under the same error model and also we can simulate other reliable multicasting protocols under the assumption of uniformly distributed PEP, in order to compare them with our protocols. Beyond that, the simulator can accommodate customized error models, if we want to be more accurate in simulating the different reliable multicasting protocols that we want to evaluate and analyze.

3.3.2 Metrics

Since we presented some of the characteristics of the customized simulator that we built for the simulation of the reliable multicasting protocols that we proposed, it is required to predefine the performance metrics that we will use for evaluating and comparing the different protocols. We consider the following metrics as some of the most important for the evaluation of the protocols that we designed, and based on them we will decide the effectiveness of those protocols and also we will have the opportunity to compare the performance behavior that we will get from the simulation with the intuitive statements that we made in the previous paragraphs. Those performance metrics are:

- Transmission Rounds vs. Air Cache Size
- Transmission Rounds vs. Group Size
- Transmission Rounds vs. Packet Error Probability
- Normalized Gain

Before we move on to the actual study of the non-adaptive and adaptive reliable multicasting protocols (e.g., chapters 4 and 5, respectively), we will give an overview of the above listed metrics. Most of the metrics have to do with the Transmission Rounds metric, which is computed in accordance with parameters that have to do with the dynamic characteristics of the simulated protocol (e.g., Air Cache size) and with the characteristics of the simulated network scenario (e.g., Group Size and PEP). The definition of the Transmission Rounds is the number of repetitive transmissions of the TG that are required in order the TG data packets to be delivered reliably to each one of the members of the multicast group. So, with this performance metric we can get an idea of the end-to-end delay for each one of the simulated protocols. We used this

performance metric, because was used before on the evaluation of reliable multicasting protocols from other researchers, so we had to be consistent. Apart from the consistency reasons we will have also, the opportunity to compare our results with the corresponding results that already exist for some of the known reliable multicasting protocols. Furthermore, the Transmission Rounds metric, gives us a very representative picture for the performance of the simulated protocols without having to go into the details of the network when we simulate the different scenarios (e.g., like the propagation delay, the transmission delay, the computation delay, etc.).

The metric that refers to the Transmission Rounds versus the Air Cache size is computed by simulating different values of Air Cache size while we keep unchanged the rest of the input parameters, so we can get an idea how the Air Cache size affects the performance behavior of the protocol. Also, we can make observations such as what is the optimal size of the Air Cache for specific TG sizes. We will discuss about this metric when we give out results that refer on it. The next metric is the Transmission Rounds versus the Group Size. We collect results for this metric by modifying the size of the multicast group, while we keep the rest of the input parameters unchanged. This metric will provide us information about the scalability of the protocol. The definition of scalability as we understand it, is the ability of a protocol to retain its performance characteristics as the multicast group gets larger. One very important metric is the Transmission Rounds versus the Packet Error Probability. We collect results for this metric by modifying the PEP while we keep the rest of the input parameters unchanged. The reason of the existence of this metric is to provide us with information that has to do with the robustness of the simulated protocol, which is a very important issue, especially for protocols that aim on

networks, which are characterized from high error probability links. Based on this metric we will evaluate the robustness of our protocols in the following chapters.

Finally, the Normalized Gain metric is considered extremely important when we have to compare the proposed protocols with other reliable multicasting protocols. We introduced this metric because the proposed protocols include the notion of the Air Cache, or in other words the use of an extra dedicated channel for the transmission of packets crucial to the recovery of corrupted or erroneous packets at the receiving ends. As we mentioned above when we were discussing the performance expectations that we have from the proposed protocols, we estimate that our protocols will have better performance characteristics compared to the protocols that do not use air caching. When we compute the performance gain between the protocols that use air caching and those that do not, we have to take into consideration the bandwidth that is dedicated and used from each of the protocols along with their performance. So, the Normalized Gain combines the performance and bandwidth of the protocols that we compare in order to get the real performance gain. This metric is very important since we will compare our protocols with already existing reliable multicasting protocols and will decide the effectiveness of the former ones mostly based on this. The Normalized Gain is the ratio of the normalized performances of the protocols that we compare. Normalized performance for a protocol is the performance that a protocol has normalized by the total amount of bandwidth that requires in order to achieve this performance. In our case the performance of a protocol is computed through the number of required Transmission Rounds for the reliable delivery of a TG. For example if we want to compare a protocol A with a protocol B then the Normalized Gain will be given from the following ratio:

$$NormalizedGain = \frac{TransmissionRounds_A * BWsize_A}{TransmissionRounds_B * BWsize_B}$$

By comparing the above ratio with one we can conclude, which of the protocols performance is better. For example:

if $NormalizedGain < 1$ then A has better relative performance than B
if $NormalizedGain > 1$ then B has better relative performance than A

We will follow this method, when we compare the protocols that we propose with protocols that already have been proposed by other researchers.

Chapter 4: Non-Adaptive Air Cache RM Protocols

4.1 Introduction

In this chapter we exploit and evaluate the Non-Adaptive Air Cache Reliable Multicasting Protocols. The non-adaptive Air Cache protocols compared to the adaptive ones are expected to have fewer advantages than disadvantages, but one major advantage is that there is no need for feedback from the receivers. As we are going to exploit in the next chapter the adaptation of the Air Cache is based merely on feedback. The three protocols that we discuss here are the UDPAC (Unchanged Data Packets in the Air Cache), the RDPAC (Replace Data Packets in the Air Cache) and PPAC (Parity Packets in the Air Cache – Extending Buffering). The functioning of all three is based on a constant size Air Cache. The dedicated bandwidth to the Air Cache does not change throughout the transmission session. UDPAC and RDPAC do not use parity packets in contrast with PPAC. The advantage of that is the simplicity, the low energy consumption and the low complexity of the protocols along with the increased performance and the robustness, compared not only to the simple ARQ protocols but also in some cases (e.g. small group sizes < 1000) to the FEC protocols. In the following section we describe the above protocols from the algorithmic point of view. Their performance in terms of the required Transmission Rounds for the reliable delivery of the transmission group (TG), and their robustness in error prone environments is given in the third section of this chapter. Also, one more important performance metric that we are going to use is the Normalized Gain, which is defined in the subsequent paragraphs, and has to do with the

relative performance of the protocols by combining the performance and the extra bandwidth usage because of the Air Cache.

4.2 Protocols' Functionality

In the following paragraphs we describe the characteristics and the functionality of the three proposed non-adaptive protocols. The protocols are characterized based on the refreshing rate of the Air Cache, on the content of the Air Cache (i.e. data packets or parity packets) and on how the packets that belong to the Air Cache are chosen. The three protocols are the UDPAC (Unchanged Data Packets in Air Cache), the RDPAC (Replace Data Packets in Air Cache) and the PPAC (Parity Packets in the Air Cache). The following algorithmic presentation of the protocols clarifies their functionality and their internal characteristics.

4.2.1 Protocol UDPAC (Unchanged Data Packets in Air Cache)

A general description of the UDPAC protocol based on its functionality is that it is an enhanced ARQ protocol. The enhancement is due to the use of air caching. Similar to all the protocols we propose, UDPAC's operation is based on the concurrent transmission of the packets in channel C_1, which is dedicated to the transmission of the TG data packets, and in channel C_2, which acts as Air Cache. The parameters that differentiate the operation of this protocol from the rest of the protocols are the Air Cache's refreshing rate, the type and content of the Air Cache and how the packets that consist the Air Cache are chosen. Specifically, speaking about UDPAC, the Air Cache is updated per

Transmission Group Transmission Round (TGTR), contains data packets and these packets are chosen randomly from the transmission group. The algorithm of the protocol follows:

Step I: Transmit in C_1 the TG data packets and in C_2 the packets of Air Cache. The Air Cache is filled by picking at random data packets from the TG.

Step K: The source waits to collect a number of positive responses (e.g. each member of the group informs the satellite only when it receives reliable all the TG data packets) equal to the number of hosts in the multicasting group. If the source has received positive requests equal to the number of participants in the multicast group then the transmission of the TG data packets stops here and the transmission rounds for the completition of the reliable transmission of the TG data packets are *K-1*. Otherwise, the source retransmits the data packets and the Air Cache packets in the same fashion as in Step I, in channels C_1 and C_2 respectively. The difference from Step I is that the contents of the Air Cache have changed by picking again at random data packets from the TG. This step is repeated by substituting *K* with *K+1*.

4.2.1.1 Pros and Cons of UDPAC

As we can easily observe from the algorithmic steps of UDPAC, the protocol is simple and lightweight in terms of the overhead that is created due to the feedback messages from the participants of the group to the satellite, which in our scenario we have assumed as the source of the transmission. The simplicity occurs because the protocol does not involve any processing of feedback at the source. The protocol behaves

like an ARQ protocol with the difference that there is no negative feedback involved. The source just retransmits in a per round fashion.

UDPAC is not only lightweight due to the lack of negative feedback but can be characterized also as lightweight because it does not require from the receiving devices to have the processing power that an FEC protocol demands. This is because there are no parity packets involved, which most of the time, require online encoding and decoding. Those actions demand increased processing power from the sending and receiving devices in order to produce and correct a group of erroneous or corrupted packets, respectively. Because of the protocol hardware requirements we can characterize the protocol as an energy saver protocol. In environments were the devices are portable and their source of energy is finite and not rechargeable the latter characteristic of the protocol can be proved of increased significance, in combination also with UDPAC's performance.

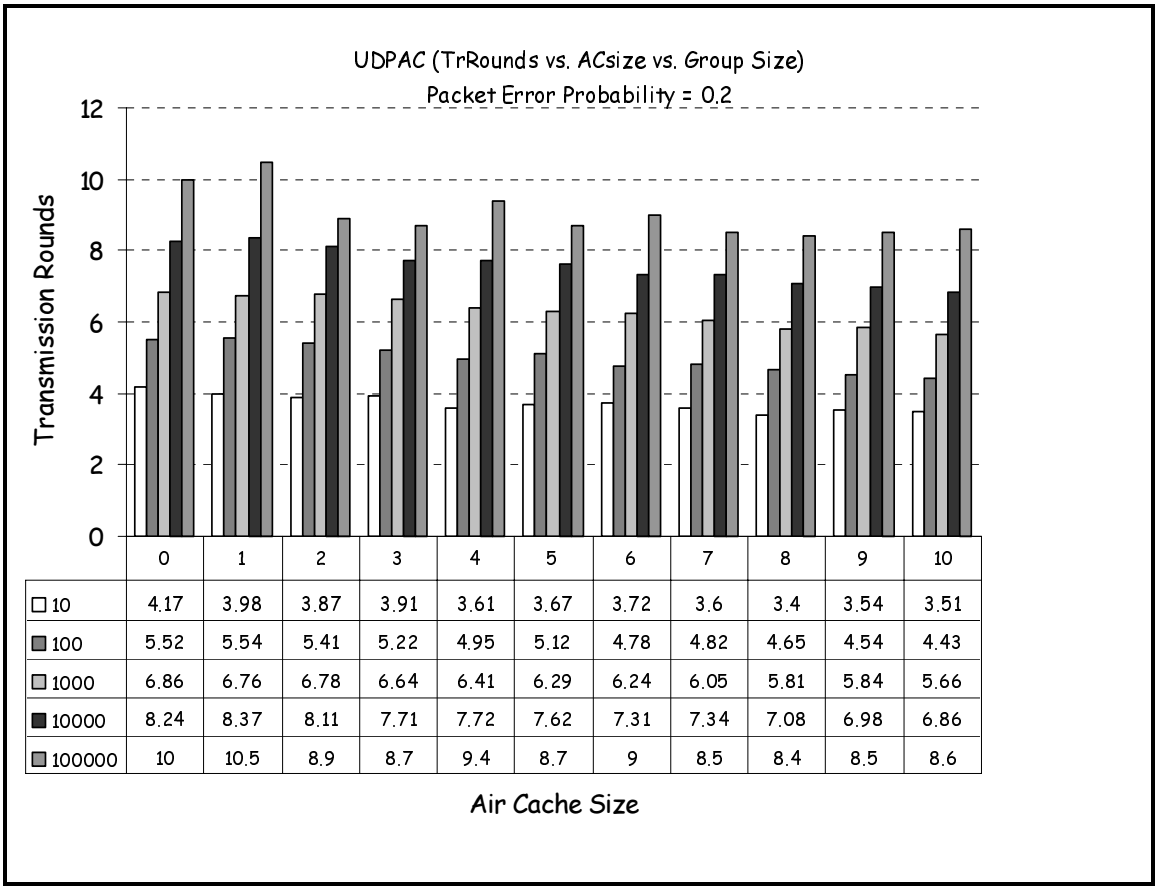
The drawback of a lot of reliable multicasting protocols is that they require increased buffering capabilities at the receivers; usually this is the case for the receiver-oriented category for this kind of protocols. Based on the fact that there are network environments, where the receiver devices due to their specifications or due to small cost cannot accommodate the appropriate amount of hardware for the buffering requirements of a reliable multicasting protocol, protocols that are based on buffering large amounts of packets at the receiving end, are destined to fail in those environments. So, designing a protocol that can perform efficiently in those environments would be a great accomplishment. The performance analysis of UDPAC at the end of the paragraph presents in what degree we succeeded on that.

Beyond, the advantages of UDPAC, which are basically, related to the hardware requirements and the power consumption at the receiving end, there are also drawbacks to the protocol. The most important is its performance in terms of the required transmission rounds. Because the protocol operates on the same manner as a simple ARQ protocol, its performance is slightly better than that because of the air caching. There is no huge improvement in transmission rounds because at the moment we deal with non-adaptive protocols and the content of the Air Cache is selected randomly and is not based on any kind of feedback information. Also, one more drawback of the protocol is that we assume that the source knows the number of participating hosts in order to avoid the negative feedback per round, and just collect a number of positive feedback equal to the number of participants to the multicast group. From the practical point of view this assumption is valid in the case where the group is static, but in the case of dynamic membership changes during the transmission (i.e. nodes fail, nodes migrate, etc.), we have to reconsider it. Without loss of generality we can assume that there always exists a feedback channel that informs the source for the membership changes.

4.2.1.2 Simulation Results for UDPAC protocol

Using the simulator that we described in Chapter 3, we simulated UDPAC in order to get a feeling about its performance in terms of the required transmission rounds for the reliable delivery of TG data packets. Apart from that performance metric, we evaluated UDPAC in terms of its relative performance compared to the simple ARQ protocol. The relative performance is a measure that combines the transmission rounds and the bandwidth usage.

The results that we got concerning the transmission rounds that are required for the reliable delivery of TG data packets are shown in the following graph. We assumed that the PEP is 0.2 and the TG is 20 data packets.



	0	1	2	3	4	5	6	7	8	9	10
10	4.17	3.98	3.87	3.91	3.61	3.67	3.72	3.6	3.4	3.54	3.51
100	5.52	5.54	5.41	5.22	4.95	5.12	4.78	4.82	4.65	4.54	4.43
1000	6.86	6.76	6.78	6.64	6.41	6.29	6.24	6.05	5.81	5.84	5.66
10000	8.24	8.37	8.11	7.71	7.72	7.62	7.31	7.34	7.08	6.98	6.86
100000	10	10.5	8.9	8.7	9.4	8.7	9	8.5	8.4	8.5	8.6

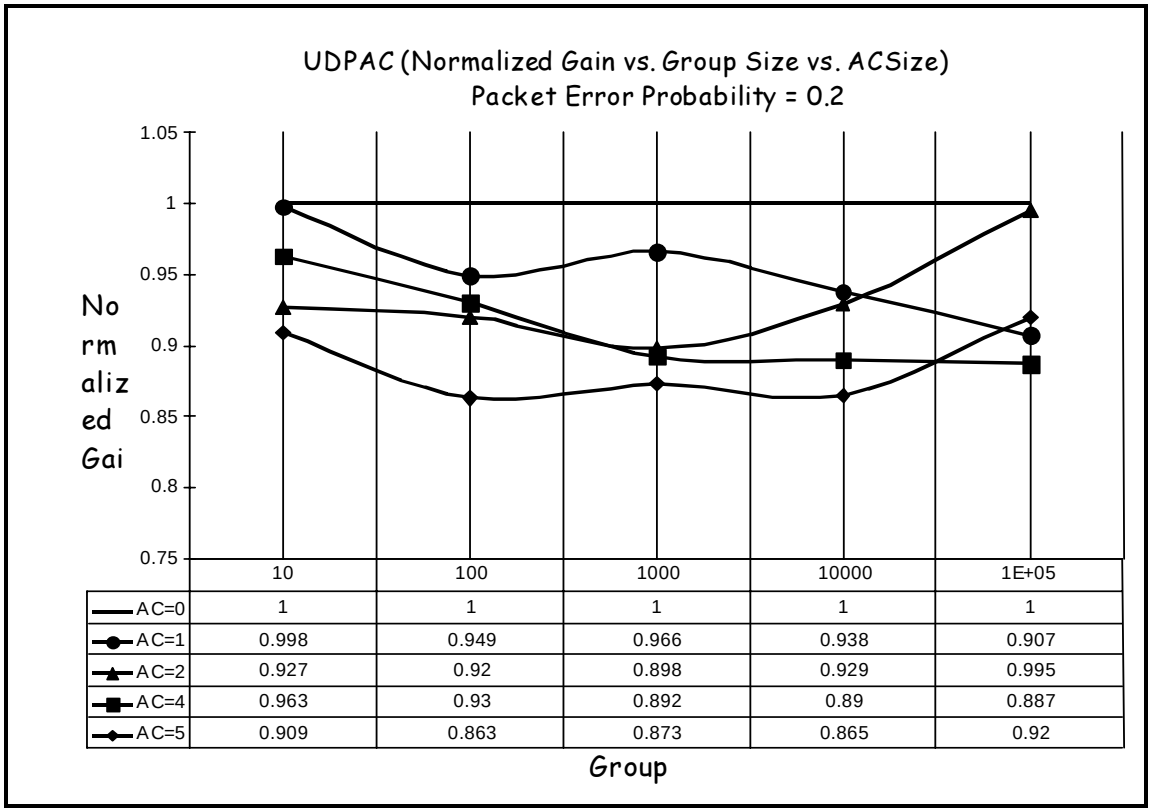
Figure 4.1: (UDPAC) Transmission Rounds vs. AC Size vs. Group Size (PEP=0.2,TG=20)

The normalized gain in the case of UDPAC is given from the following formula:

$$NormalizedGain = \frac{TransmissionRounds_{ARQ} * TGsize}{TransmissionRounds_{UDPAC} * (TGsize + ACsize)}$$

We introduce this metric because we need to have a clear view of the protocol's performance. As we can observe from the above figure the performance of UDPAC is better than the performance of a simple ARQ protocol. By "performance" we mean the number of required Transmission Rounds for the reliable delivery of TG data packets to

each one of the members of the multicast group. Moreover and beyond the latter result we have to take into consideration the extra bandwidth (e.g., reserved for the Air Cache) usage from UDPAC compared to a non Air Cache ARQ-based protocol. With the use of normalized gain we can find out if the improvement in performance is adequate to compensate for the extra bandwidth usage. How the two parameters (e.g. transmission rounds, bandwidth usage) are combined in order to demonstrate the real performance gain using UDPAC is obvious from the above relation, which is called Normalized Gain ratio. The graph, which presents the values of this metric, follows. The values are based on the collection of results that are presented in the previous graph and correspond in the scenario where the TG is 20 data packets and the PEP was assumed 0.2.

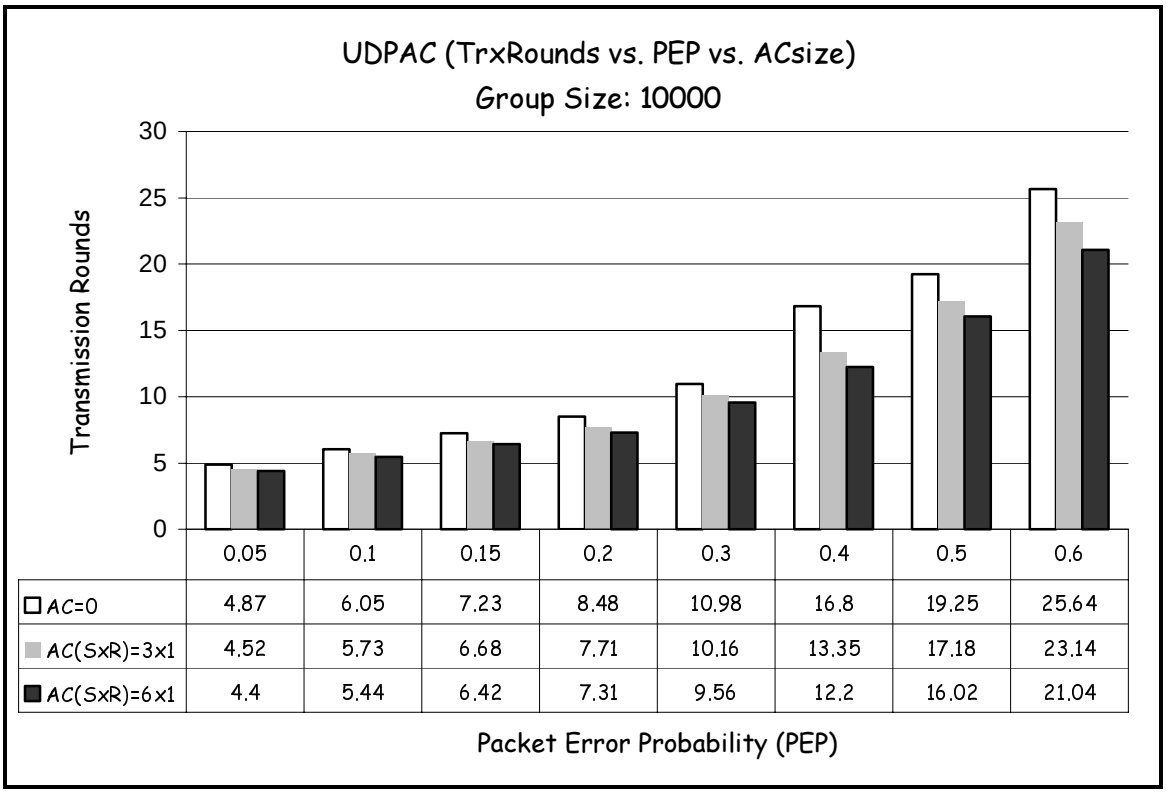


	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	0.998	0.949	0.966	0.938	0.907
AC=2	0.927	0.92	0.898	0.929	0.995
AC=4	0.963	0.93	0.892	0.89	0.887
AC=5	0.909	0.863	0.873	0.865	0.92

Figure 4.2: (UDPAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

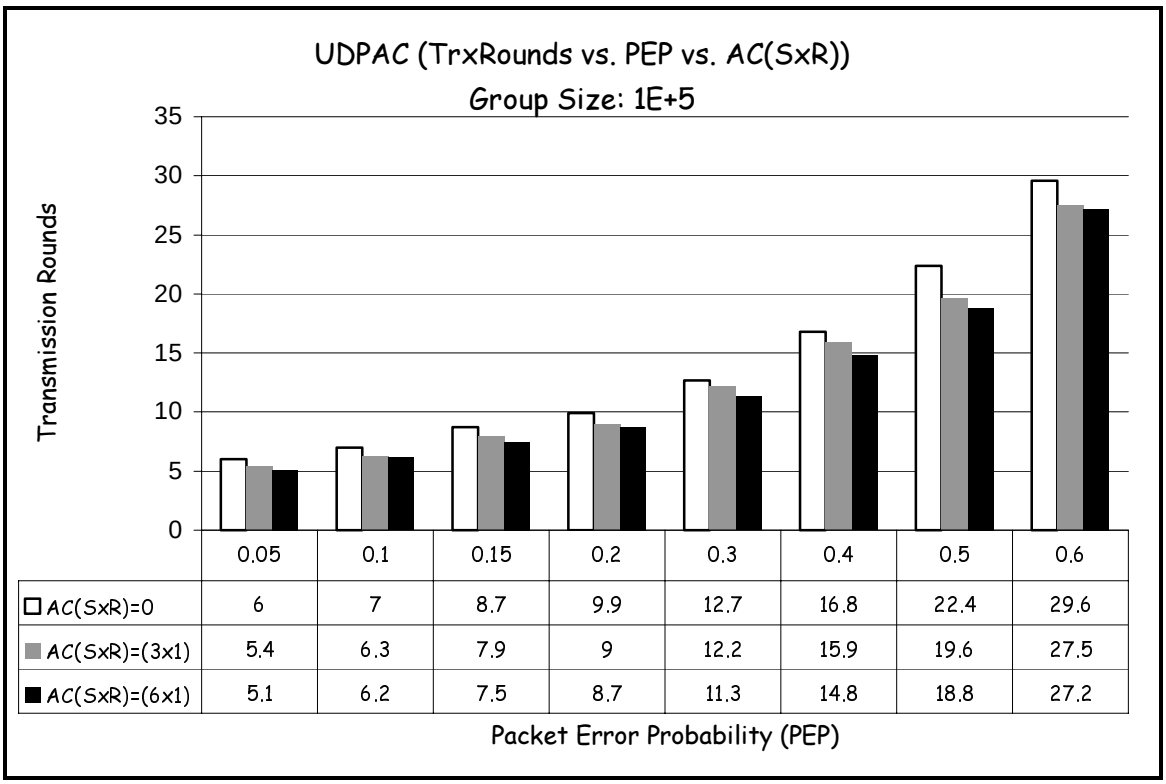
The results that appear for Air Cache Size equal to zero correspond to a simple ARQ protocol because we just apply ARQ and not air caching (e.g., *Air Cache size is* 0) for the reliable transmission of the TG data packets. Obviously, when the UDPAC protocol is applied without Air Caching is the same as a simple ARQ protocol, and this is how we explain the flat line to Normalized Gain equal to one when the Air Cache size in UDPAC equals to 0 packets.

Furthermore, and continuing the evaluation of the UDPAC protocol, we explore the error behavior of the protocol compared to the *Group Size* and the *Air Cache size*. The results that follow demonstrate the performance of the protocol in terms of required transmission rounds. We simulate the protocol for different Packet Error Probabilities (PEPs) for group sizes 10000 and 100000 and for three different Air Cache sizes (e.g., 0 (non Air Cache ARQ based protocol), 3 and 6 packets). Our goal is to exploit the behavior of UDPAC in error prone environments, and how the increase in PEP affects the required Transmission Rounds.



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC=0	4.87	6.05	7.23	8.48	10.98	16.8	19.25	25.64
AC(SxR)=3x1	4.52	5.73	6.68	7.71	10.16	13.35	17.18	23.14
AC(SxR)=6x1	4.4	5.44	6.42	7.31	9.56	12.2	16.02	21.04

Figure 4.3: (UDPAC) Transmission Rounds vs. PEP vs. AC Size (TG=20, GS=10^4)



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=0	6	7	8.7	9.9	12.7	16.8	22.4	29.6
AC(SxR)=(3x1)	5.4	6.3	7.9	9	12.2	15.9	19.6	27.5
AC(SxR)=(6x1)	5.1	6.2	7.5	8.7	11.3	14.8	18.8	27.2

Figure 4.4: (UDPAC) Transmission Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

Commenting on the results presented on the above graphs, the behavior of UDPAC in error prone environments does not differ from the error behavior of a simple ARQ protocol (i.e., corresponds to the results for Air Cache size equal to zero (AC=0)). So, UDPAC as ARQ is not robust to error prone environments because its performance is degraded exponentially as the Packet Error Probability increases. Using the results from the above graphs, for group size 100000 and for PEP=0.5, the *required Transmission Rounds* have increased by 100% compared to the required transmission rounds for the case where the PEP=0.2.

4.2.1.3 Performance Analysis of UDPAC protocol

Based on the results that we collected from the simulation of UDPAC protocol, we can comment the overall performance of the protocol. As we were expecting intuitively, the performance of the protocol in terms of transmission rounds is better that an ARQ protocol and that is due to the use of air caching. Even though the improvement in performance is not significant and that can be explained from the fact that the Air Cache is filled randomly, without the use of feedback (e.g. there are no requests for retransmission from the participating members of the receiving multicasting group). So, in order the UDPAC protocol to be feedback free we sacrifice its delay performance.

Also, analyzing the results concerning the Normalized Gain metric we can conclude that the increase in delay performance is not adequate to compensate for the extra bandwidth usage. We have drawn that conclusion because in each instance of UDPAC protocol, for different *Air Cache sizes*, the *Normalized Gain* is constantly less than one. For a protocol that uses Air Cache to compensate for the extra bandwidth usage with its

delay performance, the following should be true, as it appears from the formula that characterizes the corresponding performance metric:

$$Normalized\ Gain \geq 1$$

Obviously, by analyzing the results taken from simulating the UDPAC protocol, we draw the conclusion that the UDPAC protocol is not a protocol that we would use for improvement in delay performance of a reliable multicast transmission session. We favor and study this protocol because it can operate without the use of feedback (i.e. no negative feedback is required for its operation) and because of the lightweight memory and processing requirements that poses to the receiving devices. So, we could use UDPAC in environments were the receiving devices are not so powerful and the power consumption is extremely important requirement for the survivability and connectivity of the wireless network. One more requirement in order UDPAC to be effective is that the multicasting group must not change often (e.g. as we mentioned the transmission stops by the time the source has collected number of positive acknowledgments equal to the number of participating nodes).

In terms of the error behavior of UDPAC and referring to the graphs above, where this error behavior is demonstrated, we draw the conclusion that UDPAC does not behave better than a simple ARQ protocol, regardless of the extra bandwidth usage. So, even though we use air caching, the protocol is not more robust than an ARQ protocol in error prone environments.

We compared UDPAC with a simple ARQ protocol because does not involve Forward Error Correction (FEC). Even though FEC has been proven very efficient solution for reliable multicasting, demands increased processing power and more

buffering space, not only at the receiving nodes but at the source as well. So, among the RM protocols that are not using parity packets because of the limitations we mentioned above, UDPAC seems a good solution. That is because in some cases we do not care about the delay but we require reliable delivery of data to devices that are not very powerful, so a lightweight protocol, like UDPAC, it would be a nice solution.

4.2.2 Protocol RDPAC (Replace Data Packets in Air Cache)

The RDPAC protocol operates in the same fashion as the UDPAC protocol. RDPAC is not based on FEC but uses air caching of data packets and also is an ARQ based protocol. Similarly with UDPAC, RDPAC is a feedback free protocol and the Air Cache is filled randomly with data packets from the transmission group. The algorithmic difference between the two protocols is located on the refreshing rate of the Air Cache. In UDPAC the refreshing rate is per TGTR (Transmission Group Transmission Round), but in the case or RDPAC is per ACTR (Air Cache Transmission Round). The meaning of the latter is that the Air Cache in the case of RDPAC is filled with a new set of packets more frequently than UDPAC because $TGTR \geq ACTR$ (i.e., see figure 3.1). Based on the last observation someone expects the RDPAC protocol will have better performance characteristics than UDPAC as we already mentioned in chapter 3. We will evaluate our intuition after the presentation of the performance results. Before getting into the details of RDPAC's performance evaluation, it's better to describe the basic algorithm of the protocol.

Step I: Transmit in C_1 the TG data packets and in C_2 the packets of Air Cache. The Air Cache is refreshed every ACTR by picking at random ACRS (Air Cache Round Size) data packets

Step K: The source waits to collect a number of positive responses (e.g. each member of the group informs the satellite only when it receives all the TG data packets reliably) equal with the number of hosts in the multicast group. If the source has received positive requests from each one of the members of the multicast group then the transmission of the TG data packets stops here and the Transmission Rounds for the completition of the reliable transmission of the TG data packets are *K-1*. Otherwise, the source retransmits the data packets and the Air Cache packets in the same fashion as in Step I, in channels C_1 and C_2 respectively. During the transmission in C_1, the Air Cache is concurrently transmitted in C_2 and simultaneously is refreshed with new data packets every ACTR. We repeat this step by substituting *K* with *K+1*.

As we can observe from the steps of the algorithm above, RDPAC slightly differs from UDPAC, but how about the effectiveness of the protocol? The answer to this question is given to the following paragraph where we present the simulation results of RDPAC. Similarly, with the performance evaluation of UDPAC, the performance metrics we use are the required Transmission Rounds for the reliable delivery of a certain number (TG) of data packets to each one of the members of the multicasting group, and the Normalized Gain. We use the latter metric to find out if the improvement in performance compared to a simple ARQ protocol is sufficient to compensate for the extra bandwidth usage.

4.2.2.1 Pros and Cons of RDPAC

Due to the similarity of RDPAC protocol with the UDPAC, the advantages and disadvantages of RDPAC are quite similar to UDPAC's advantages and disadvantages. RDPAC is a lightweight protocol because it is feedback free and does not demand powerful receiving devices in terms of buffering space (e.g., memory) and processing power.

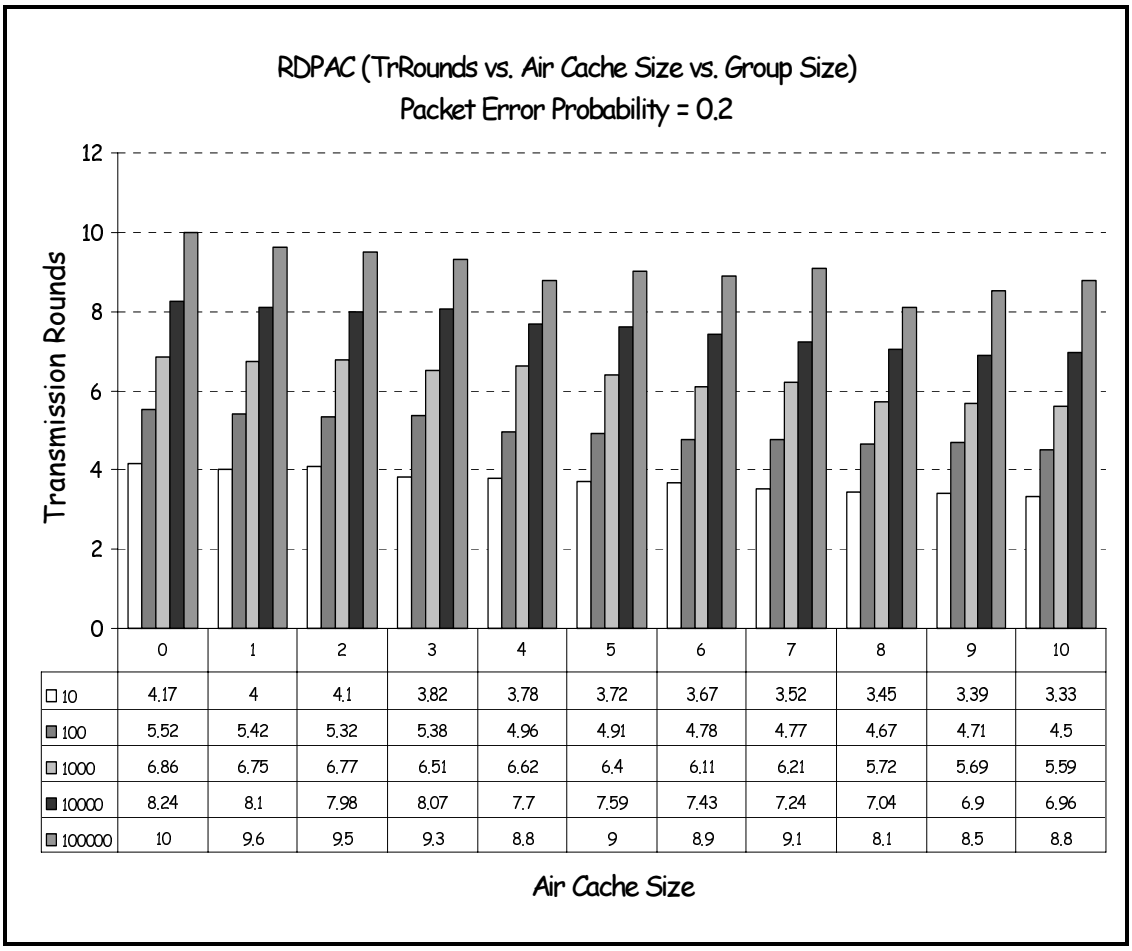
Similar to UDPAC the disadvantages of RDPAC are located basically on the delay performance of the protocol (i.e. transmission rounds for the reliable delivery of the data packets). RDPAC, like UDPAC, is not based on Forward Error Correction (FEC), and this results in a performance that is slightly better that a simple ARQ protocol. As we mentioned above the sacrifice in performance is because we wanted RDPAC to be a lightweight protocol for its use in environments where the receiving devices have limited power and memory characteristics.

Furthermore and compared to UDPAC, RDPAC has advantages and disadvantages that result from the algorithmic differences that this protocol has compared to UDPAC. The difference is in terms of the refreshing rate of the Air Cache, which in RDPAC happens per ACTR and in UDPAC per TGTR. Intuitively, in the case that we have multiple Air Cache Transmission Rounds per Transmission Group Transmission Round, it seems that RDPAC sends through the Air Cache larger set of different data packets to the multicast group than the UDPAC does. Beyond, this intuitive conclusion, UDPAC seems to be more robust in error prone environments where the packet error probability (PEP) is high. We base the latter intuitive comment on the fact that UDPAC sends the same group of data packets per ACTR, and because during the TGTR we have multiple

ACTR, even though some of the Air Cache packets may be corrupted, eventually can be received correctly in any of the multiple ACTR. We will have the time to evaluate the correctness of our intuitive comments on RDPAC's performance analysis paragraph that follows (§ 4.2.2.3). In the next paragraph we present the simulation results for RDPAC protocol.

4.2.2.2 Simulation Results for RDPAC protocol

The results that we collected during the simulation of the protocol concern the required Transmission Rounds and the Normalized Gain metrics. The following graph presents the behavior of RDPAC protocol in terms of the *required Transmission Rounds* for the complete reliable delivery of the TG data packets to a multicast group of *Group Size*, which varies from 10 to 10^5 members. We collected the following results by assuming that the TG is 20 data packets and the Packet Error Probability (PEP) is 0.2.



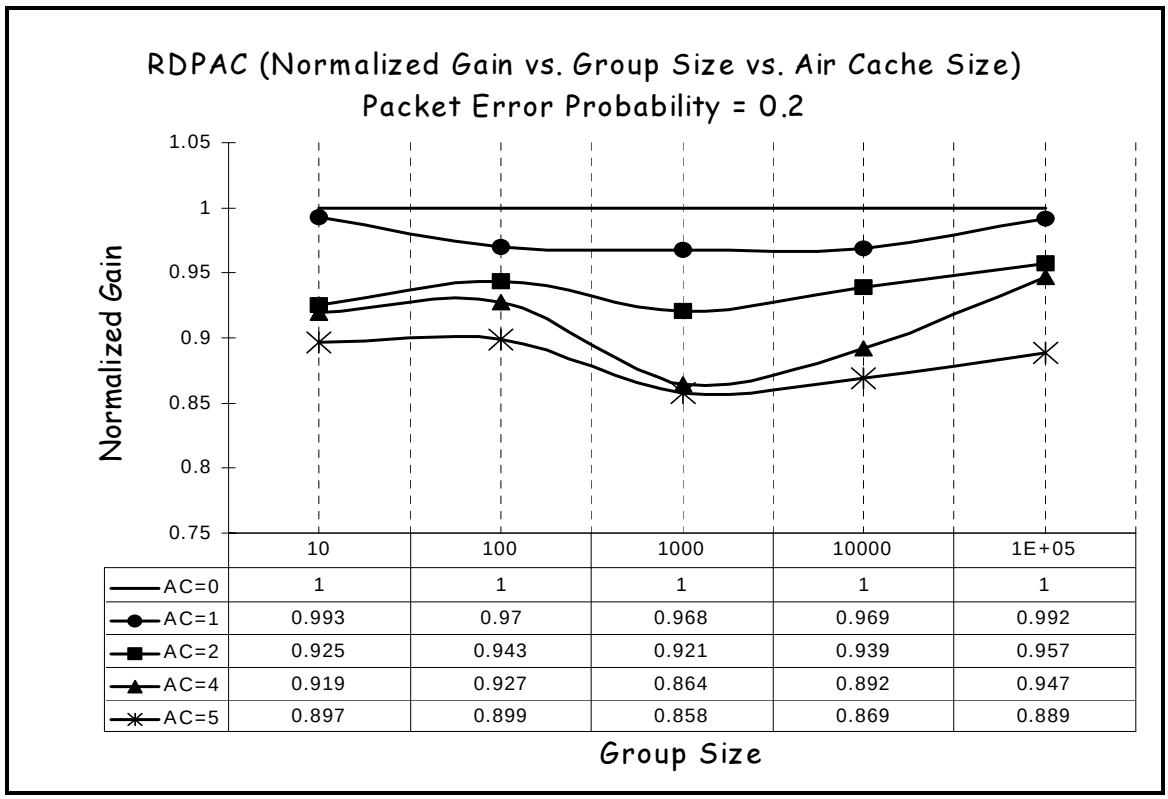
	0	1	2	3	4	5	6	7	8	9	10
10	4.17	4	4.1	3.82	3.78	3.72	3.67	3.52	3.45	3.39	3.33
100	5.52	5.42	5.32	5.38	4.96	4.91	4.78	4.77	4.67	4.71	4.5
1000	6.86	6.75	6.77	6.51	6.62	6.4	6.11	6.21	5.72	5.69	5.59
10000	8.24	8.1	7.98	8.07	7.7	7.59	7.43	7.24	7.04	6.9	6.96
100000	10	9.6	9.5	9.3	8.8	9	8.9	9.1	8.1	8.5	8.8

Figure 4.5: (RDPAC) Transmission Rounds vs. AC Size vs. Group Size (PEP=0.2, TG=20)

The performance of RDPAC does not differ form the performance of UDPAC. Despite the use of Air Cache, RDPAC does not appear to contribute to any dramatic improvement in the performance, compared to a simple ARQ protocol. This result was expected due to the similarity of RDPAC with UDPAC. In the same manner, someone is expecting RDPAC to have the same behavior in terms of the Normalized Gain. The latter statement is proven true by observing the following graph, which represents the behavior of the Normalized Gain metric for various group sizes and for various Air Cache sizes. We calculated the Normalized Gain values by substituting the corresponding required Transmission Rounds performance results in the following formula:

$$NormalizedGain = \frac{TransmissionRounds_{ARQ} * TGsize}{TransmissionRounds_{RDPAC} * (TGsize + ACsize)}$$

The results corresponding to non-Air Cache ARQ-based protocol were collected from the scenario where the Air Cache size equals 0 (there is no application of the air caching technique, so the RDPAC protocol behaves like a simple ARQ protocol).



	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	0.993	0.97	0.968	0.969	0.992
AC=2	0.925	0.943	0.921	0.939	0.957
AC=4	0.919	0.927	0.864	0.892	0.947
AC=5	0.897	0.899	0.858	0.869	0.889

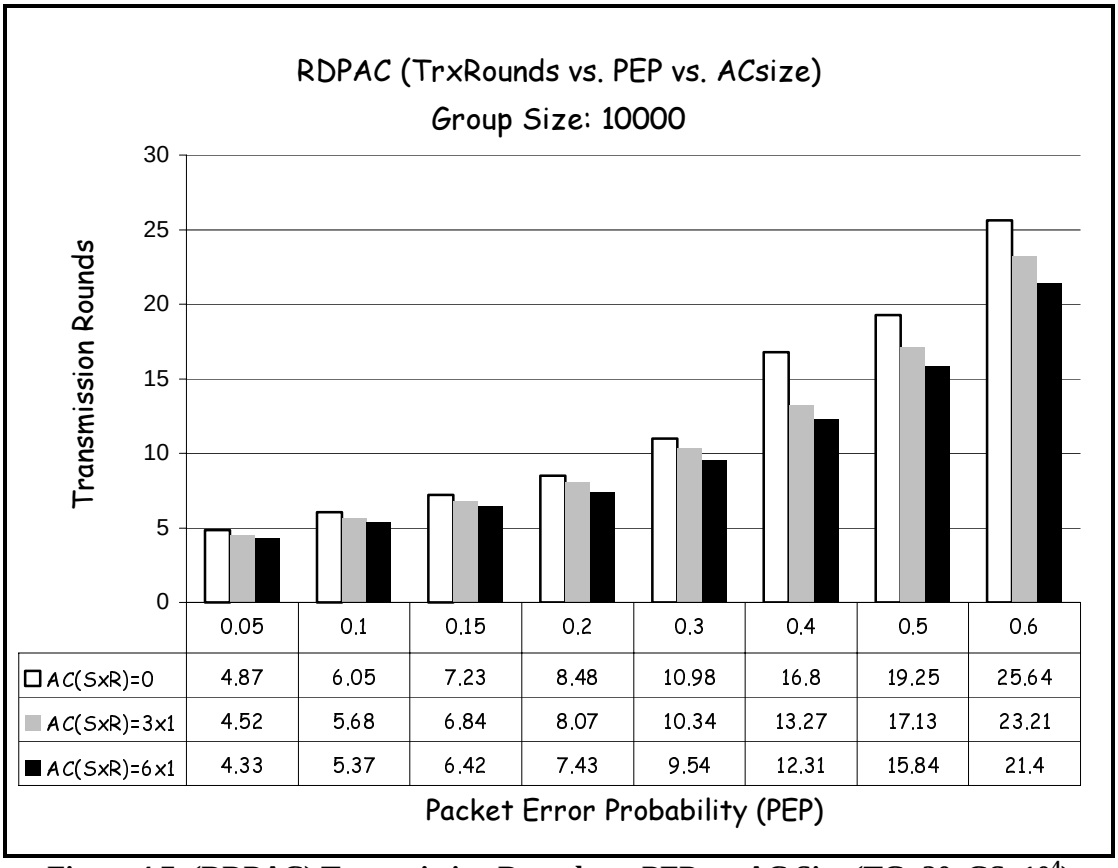
Figure 4.6: (RDPAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

Similar to the performance of UDPAC, RDPAC's delay performance does not seem to compensate for the extra bandwidth usage. That is because even though we apply the air caching technique the Normalized Gain behavior is:

$$Normalized\ \ Gain < 1$$

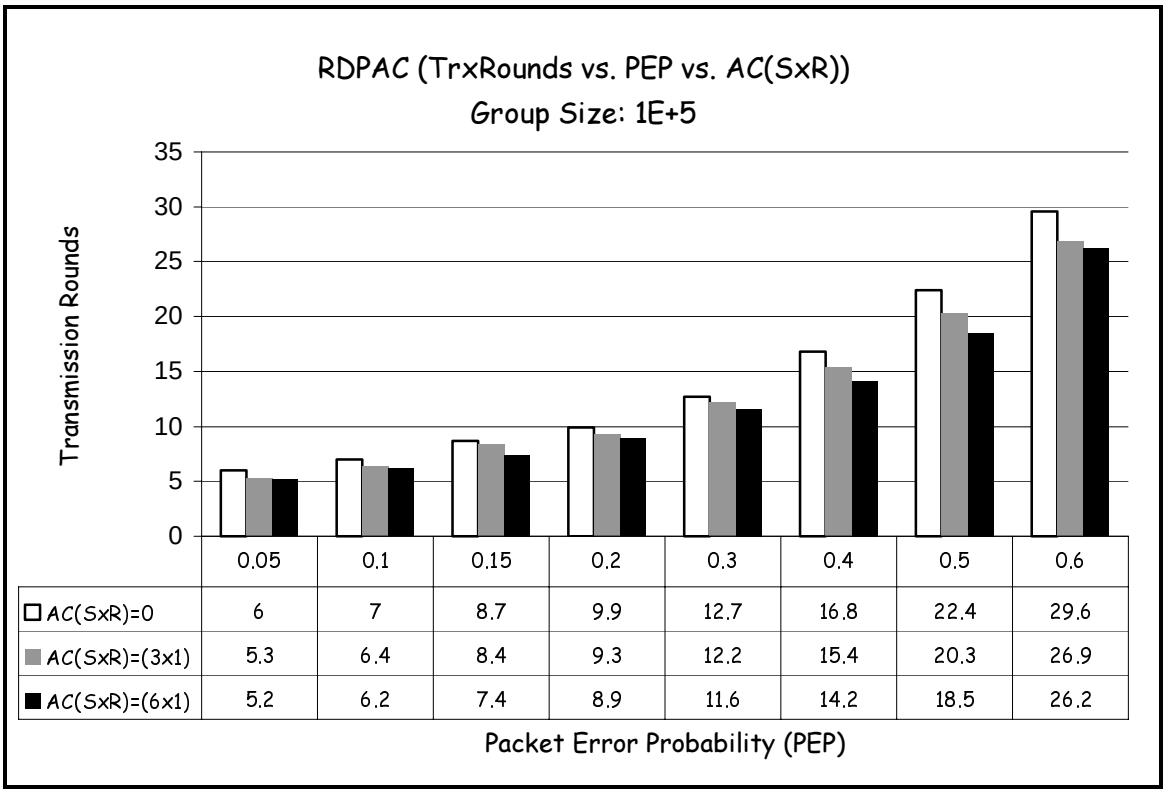
The latter result proves that actually we do not gain anything in terms of the product *BxT*, where B characterizes the bandwidth and T characterizes the Transmission Rounds.

Beyond the delay performance of the RDPAC protocol, we want to exploit the robustness of the protocol. The simulation results of the error behavior of the protocol are shown in the following two graphs, where we demonstrate the performance of the protocol for multicast group sizes 10^4 and 10^5 members, for Air Cache sizes 0 (i.e., no Air Cache), 3 and 6 packets and for various packet error probabilities (PEPs). In both simulated scenarios we assumed that the TG is 20 packets.



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=0	4.87	6.05	7.23	8.48	10.98	16.8	19.25	25.64
AC(SxR)=3x1	4.52	5.68	6.84	8.07	10.34	13.27	17.13	23.21
AC(SxR)=6x1	4.33	5.37	6.42	7.43	9.54	12.31	15.84	21.4

Figure 4.7: (RDPAC) Transmission Rounds vs. PEP vs. AC Size (TG=20, GS=10^4)



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=0	6	7	8.7	9.9	12.7	16.8	22.4	29.6
AC(SxR)=(3x1)	5.3	6.4	8.4	9.3	12.2	15.4	20.3	26.9
AC(SxR)=(6x1)	5.2	6.2	7.4	8.9	11.6	14.2	18.5	26.2

Figure 4.8: (RDPAC) Transmission Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

Obviously, the error behavior of RDPAC does not differ much from the performance of an ARQ protocol. The degradation in performance of the protocol is exponential as the packet error probability (PEP) increases. The latter fact proves the vulnerability of the protocol in error prone environments. On the other hand, this vulnerability does not differ much from the one that represents a non Air Cache ARQ-based protocol, although someone would expect that the extra bandwidth usage (e.g. Air Cache) would improve significantly the robustness of the protocol, but as we can derive from the graphs this not the case, even though we get an improvement of few Transmission Rounds compared to a non Air Cache ARQ-based protocol.

4.2.2.3 Performance Analysis of RDPAC

After having presented the simulation results about RDPAC, we make some comments on its performance and on how valuable is the existence of this protocol.

As it was expected the performance of RDPAC in terms of the required Transmission Rounds and in terms of the Normalized Gain does not differ much from the UDPAC protocol. Speaking about the Transmission Rounds, the RDPAC protocol seems to have slightly better performance than a non Air Cache ARQ-based protocol, and almost the same as UDPAC. The above graphs correspond to a Packet Error Probability equal to 0.2. The Normalized Gain is always below one, which means that RDPAC does not use the bandwidth effectively enough.

Although the Normalized Gain is less than one, this is not a reason for characterizing RDPAC protocol as inefficient. The Normalized Gain metric is closely related to the efficient use of the satellite link's bandwidth. The fact that the Normalized Gain ratio is less than one leads us to the conclusion that the protocol utilizes the available bandwidth in less efficient way compared to a non Air Cache ARQ-based protocol. In environments where the utilization of the available bandwidth is not a crucial factor and the receiving devices have limited memory, processing and power capabilities, then RDPAC and UDPAC are protocols that can be used as an alternative solution to a non Air Cache ARQ-based protocol, with the additional advantage of having better performance than the latter.

RDPAC and UDPAC are two protocols, which focus on environments where the hardware resources are limited and the bandwidth can be underutilized without great impact to the scalability and QoS of the applications that use this bandwidth. Our

question now is how can we design a protocol that has some of the lightweight features of the above two protocols but also presents a considerable improvement in delay performance. One general answer to that question is that we have to use parity packets along with air caching. In the following section we present a protocol based on the combination of Forward Error Correction (FEC) and air caching, called Parity Packets in Air Cache (PPAC) protocol. Our goal is the decrease of the required Transmission Rounds for the reliable delivery of a set of packets (TG) in a multicast group, along with the efficient utilization of the available bandwidth.

4.2.3 Protocol PPAC (Parity Packets in Air Cache)

The common characteristic of the previous two protocols (UDPAC and RDPAC) is the lack of the Forward Error Correction (FEC) technique, so there were no parity packets involved in both the transmissions on channels C_1 (data) and C_2 (Air Cache). As we explained above, this approach has to do with the lightweight characteristics of the protocols in terms of the processing power and buffering space (e.g., memory) at the receiving devices. Although UDPAC and RDPAC are not demanding, we sacrifice their performance in order to achieve that.

On the other hand, we have to design a protocol that has the ability to multicast reliably, a group of data packet to a large group of receivers in a small number of transmission rounds. A protocol like this would be very useful for applications that use the satellite as a medium for the delivery of data to a large number of satellite receivers. Such systems are the DirecTV^{TM} and the DirecPC^{TM}, where the receivers have the

hardware to accommodate a protocol, which requires enough processing power and buffering.

Having as a goal the improvement of the end-to-end delay (e.g. we want to decrease the required Transmission Rounds), we designed the PPAC protocol. PPAC stands for Parity Packets in Air Cache. As its name reveals, PPAC uses a combination of Forward Error Correction and Air Caching. The combination of the two techniques comes into play when the Air Cache is filled with parity packets. The algorithm that describes the PPAC protocol follows:

Step 1: We need to transmit TG data packets. The transmission of the TG data packets will take place on channel C_1. The Air Cache is filled with *h* parity packets, which will be transmitted, on channel C_2. Initially, we transmit concurrently the TG data packets and the *h* parity packets on channels C_1 and C_2, respectively.

Step K: Each node in the multicasting group receives both from C_1 and from C_2. When it receives correctly any TG out of the TG + *h* packets then the TG data packets can be recovered and the corresponding node has completed the reliable reception of the TG data packets and sends a positive ACK to the source. By the time the source has collected positive ACKs equal to the number of participating nodes in the multicast group, then the transmission ends and the required Transmission Rounds are *K-1*. Otherwise, the nodes do not send feedback to the source but just wait for the next transmissions in channels C_1 and C_2 in order to receive the appropriate packets (data or parity) to complete the reception of the TG data packets. During the current transmission round the receiving nodes having the group of the TG data packets incomplete, do not drop the correctly received data or parity packets because they are going to use them in the subsequent

rounds. So, the transmission rounds do not stop. We substitute *K* with *K+1* and we repeat this step.

Although PPAC requires more processing power and buffering, does not need negative feedback from the receivers per transmission round. So, the nodes do not have to access the satellite link, avoiding the high propagation delay. The drawback with this approach is that the group has to remain relatively static (e.g., slow rate of dynamic changes in the group-join/leave/node failures). That is because the source has to know exactly and at any time, the size of the multicast group for the collection of the appropriate number of positive responses for the termination of the reliable transmission.

The most important difference of the PPAC protocol compared to the UDPAC and RDPAC protocols is that PPAC uses Forward Error Correction (FEC), so it takes advantage of the properties of the parity packets, which are transmitted using the Air Cache. Intuitively, the use of parity packets in the Air Cache, which is the fundamental difference of PPAC compared to UDPAC and RDPAC, is expected to boost the performance in terms of required Transmission Rounds but also could create other drawbacks as well. In the next paragraph we describe the pros and cons of the PPAC protocol.

4.2.3.1 Pros and Cons

Before, we get into the details of the presentation and analysis of the simulation results concerning PPAC, let's do some intuitive reasoning, which explains the design and existence of this protocol.

PPAC uses Forward Error Correction (FEC), which is a technique that has already proven successful in boosting the delay performance of reliable multicasting protocols. The novelty of PPAC compared to other reliable multicasting protocols, which use FEC, is the combined use of Air Cache with FEC. This is accomplished by:

1. Filling the Air Cache with parity packets
2. The transmission of the Air Cache happens concurrently with the transmission of the TG data packets (proactive Forward Error Correction-FEC)

Due to the combined use of Air Cache and FEC we expect a significant reduction in the required Transmission Rounds for the reliable delivery of the TG data packets to each one of the members of the multicast group.

One more advantage of PPAC, is that like UDPAC and RDPAC is based on positive ACKs so the nodes do not have to send requests for retransmission to the source. This is important in the case where the source is the satellite, because we can avoid the high propagation delay introduced by the satellite link. Although, the latter characteristic of the protocol seems advantageous in terms of the end-to-end delay, we cannot say the same about the utilization of network resources. If the source had the appropriate feedback per transmission round, it could have reserved the sufficient to the Air Cache bandwidth, without reserving redundant resources, so the available bandwidth could be used more efficiently and splitted more fairly among the ongoing transmissions. In the next chapter where we explore the adaptive Air Cache reliable multicasting protocols, we present a protocol (ASPAC – Adaptive Size Parity Packets in Air Cache) that actually takes advantage of the feedback per transmission round and reconfigures the size of the Air Cache.

Speaking about the disadvantages of the PPAC, an important one is the hardware requirements that the protocol poses to the receiving devices. Those hardware requirements are referred mainly to the processing power and the memory allocated for buffering. The processing power is required for the coding (i.e. combine the correctly received data and parity packets for the reconstruction of the TG data packets). Buffering is required because in each round, each node has to keep the correctly received data and parity packets in order to combine them with the next round's correctly received data and parity packets for the completion of the reliable reception of the TG data packets.

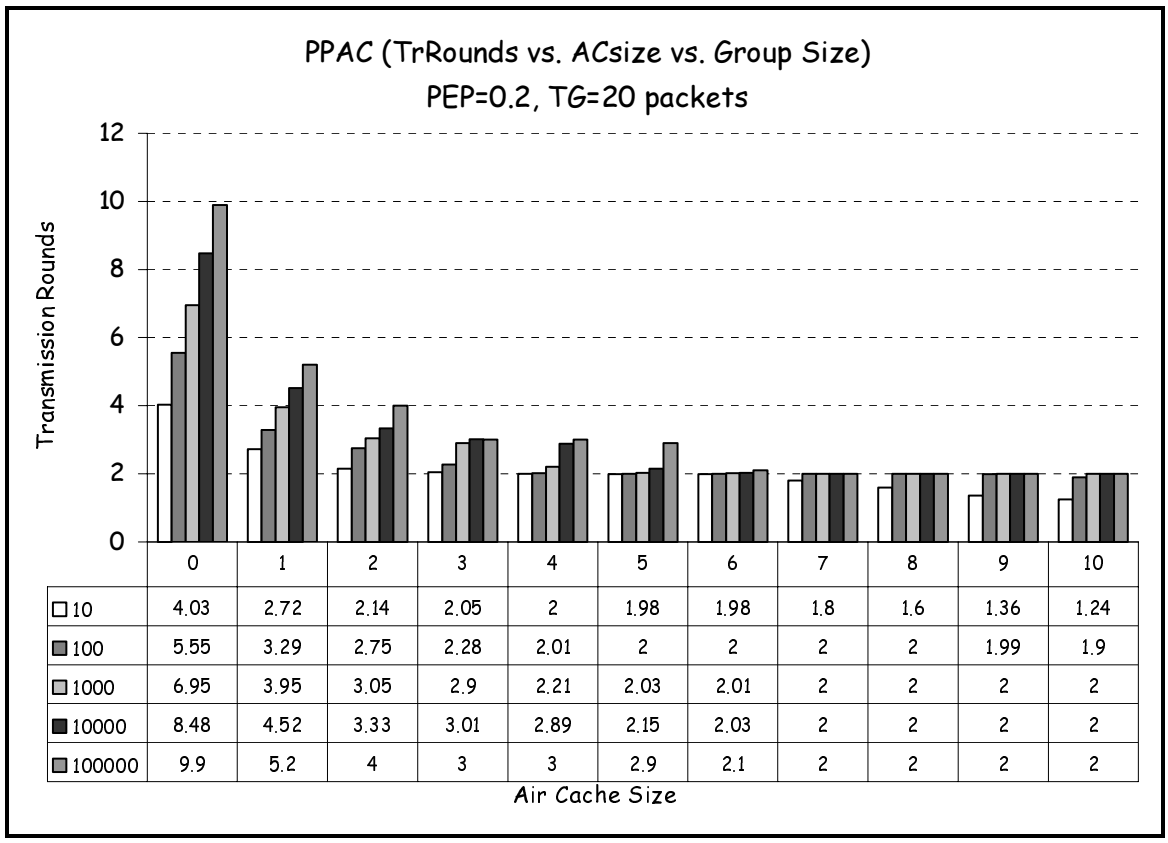
However, the above-mentioned advantages and disadvantages are based on the characteristics and the requirements of the PPAC protocol in terms of its implementation. The complete commentary on the protocol will follow the presentation of the performance results that are demonstrated in the next paragraph.

4.2.3.2 Simulation Results of PPAC

In order to evaluate the performance of the PPAC protocol, we built the simulator described in chapter 3. The results that we collected for various performance metrics by simulating PPAC are presented in this paragraph. We evaluate the protocol in two different dimensions, the first is the delay performance and the second is the robustness of the protocol. The delay performance of the PPAC protocol is characterized from the graphs that present the number of required Transmission Rounds for various *multicast Group Sizes* and various *Air Cache sizes*. Furthermore, the robustness of the protocol is presented by graphs that demonstrate how the delay performance is affected when the

packet error probability (PEP) varies from very low values (i.e. 0.05) to very high values (i.e. 0.6).

Firstly, let's exploit the delay performance of the PPAC protocol. The following graph presents the results that we collected from the simulation of PPAC and represent the number of required transmission rounds for various group sizes and for various Air Cache sizes. We assumed that the TG is 20 data packets and the packet error probability (PEP) is 0.2 in each simulated scenario.



	0	1	2	3	4	5	6	7	8	9	10
10	4.03	2.72	2.14	2.05	2	1.98	1.98	1.8	1.6	1.36	1.24
100	5.55	3.29	2.75	2.28	2.01	2	2	2	2	1.99	1.9
1000	6.95	3.95	3.05	2.9	2.21	2.03	2.01	2	2	2	2
10000	8.48	4.52	3.33	3.01	2.89	2.15	2.03	2	2	2	2
100000	9.9	5.2	4	3	3	2.9	2.1	2	2	2	2

Figure 4.9: (PPAC) Transmission Rounds vs. AC Size vs. Group Size (PEP=0.2, TG=20)

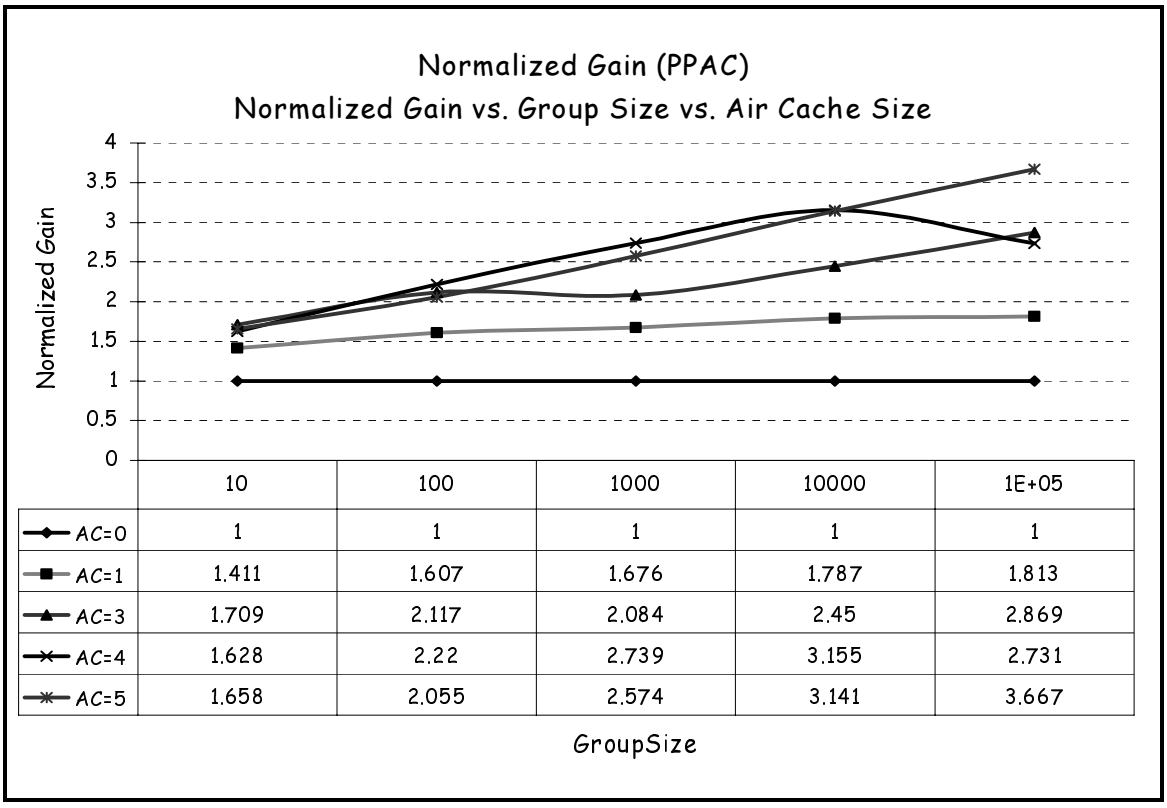
Obviously, the performance of the PPAC protocol is very promising since with Air Cache size equal to 6 parity packets (30% of the size of the C_1, where the regular transmission of the data packets happens), even for very large multicast groups of the order of many thousands members, PPAC requires at most two *Transmission Rounds* in order to deliver

reliably a transmission group (TG) of 20 packets. The improvement in performance is obvious when we compare the results that we collected for Air Cache size 0 packets (i.e., non Air Cache ARQ-based protocol and the results that correspond to it are represented by the first group of columns in the above graph) with the results for Air Cache size equal to 3 packets or more. The improvement for large multicast group sizes is significant and is up to 80% in the case of 10^5 members. For small multicast groups the improvement is between 35% to 50%.

The improvement is due to the use of extra bandwidth as Air Cache for the continuous push of parity packets. The question that arises is if the improvement in performance is sufficient to compensate for the extra bandwidth usage due to air caching. As we did in the cases of UDPAC and RDPAC we are going to use the Normalized Gain metric, which is given from the following ratio:

$$NormalizedGain = \frac{TransmissionRounds_{ARQ} * TGsize}{TransmissionRounds_{PPAC} * (TGsize + ACsize)}$$

By using the above formula, and the results that we collected from the simulation of the protocol we can obtain the following graph that represents the computed Normalized Gain ratio for the various sizes of the Air Cache. The flat line, which represents the Normalized Gain ratio that is equal to one, corresponds to the case of Air Cache size equal to 0 packets (e.g., non Air Cache ARQ-based protocol). This line sets the lower bound for the PPAC's Normalized Gain ratio, in order the performance of the protocol to compensate for the extra bandwidth usage or in other words the protocol to use efficiently the extra bandwidth due to air caching.



	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	1.411	1.607	1.676	1.787	1.813
AC=3	1.709	2.117	2.084	2.45	2.869
AC=4	1.628	2.22	2.739	3.155	2.731
AC=5	1.658	2.055	2.574	3.141	3.667

Figure 4.10: (PPAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

The Normalized Gain ratio for various Air Cache sizes must satisfy the following property in order to be proven that PPAC uses efficiently the extra bandwidth, which is reserved for air caching:

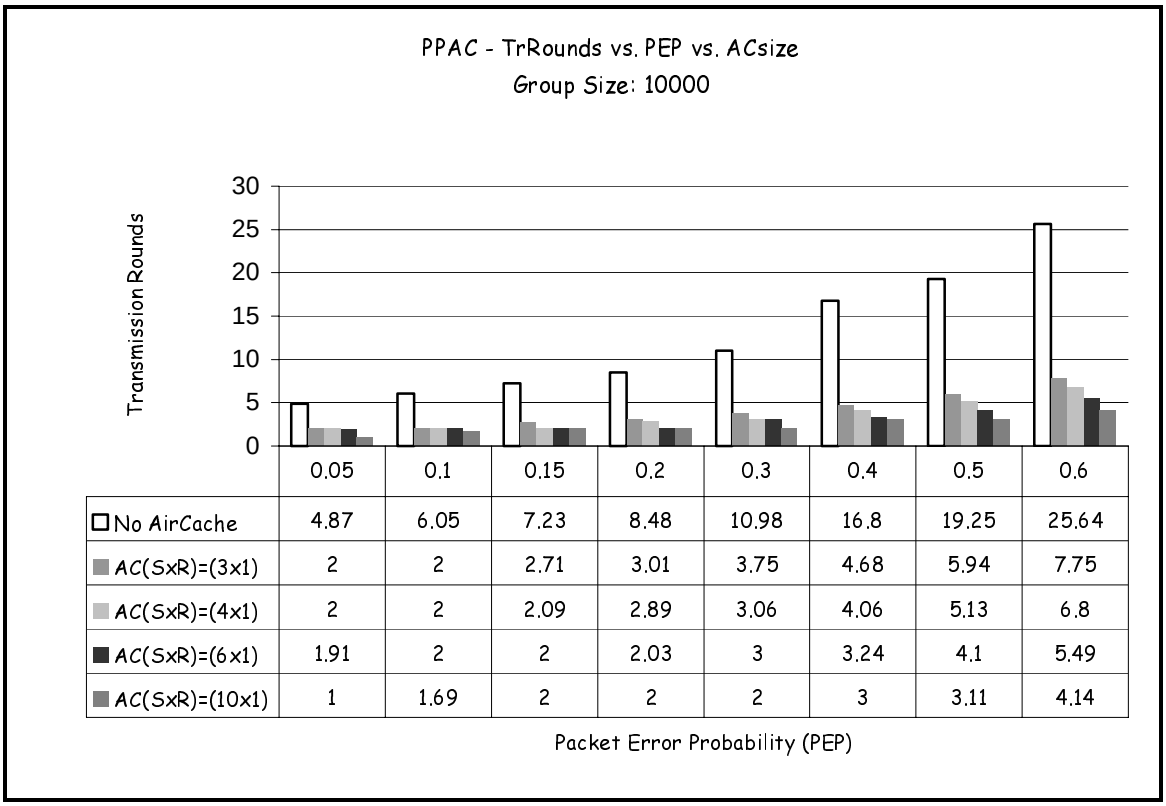
$$Normalized\ \ Gain \geq 1$$

Obviously, from the above graph this condition is satisfied for various Air Cache sizes. So, PPAC seems to be very efficient protocol, since the significant improvements in performance in terms of the required Transmission Rounds metric is sufficient enough to compensate for the extra bandwidth reserved for air caching.

Beyond the delay performance of the protocol we are going to exploit the behavior of the protocol in error prone environments. We used the simulator to collect results about the delay performance of the protocol for various packet error probabilities (PEPs). We

collected results for large multicast group sizes (i.e., 10^4 and 10^5 members) and for various values of packet error probabilities (i.e., 0.05 to 0.6). We want to investigate how the *required Transmission Rounds metric* is affected when the protocol has to deal with large multicast group sizes and the packet error probability (PEP) increases. We expect that for a robust protocol the rate of increase on the number of required transmission rounds is less than the rate of increase on packet error probability. Commenting on the following graphs that represent the error behavior of the PPAC protocol, we can highlight the fact that the rate of increase of the *required Transmission Rounds* is less than the rate of increase of the packet error probability (PEP). So, PPAC is a robust protocol and becomes even more robust when we reserve more bandwidth for air caching.

The following figures present the results that we collected for the required Transmission Rounds metric for multicast group sizes of 10^4 and 10^5 members, for various Air Cache sizes and Air Cache rounds and for packet error probabilities (PEPs) that vary from 0.05 up to 0.6. Again the TG is assumed to be 20 data packets.

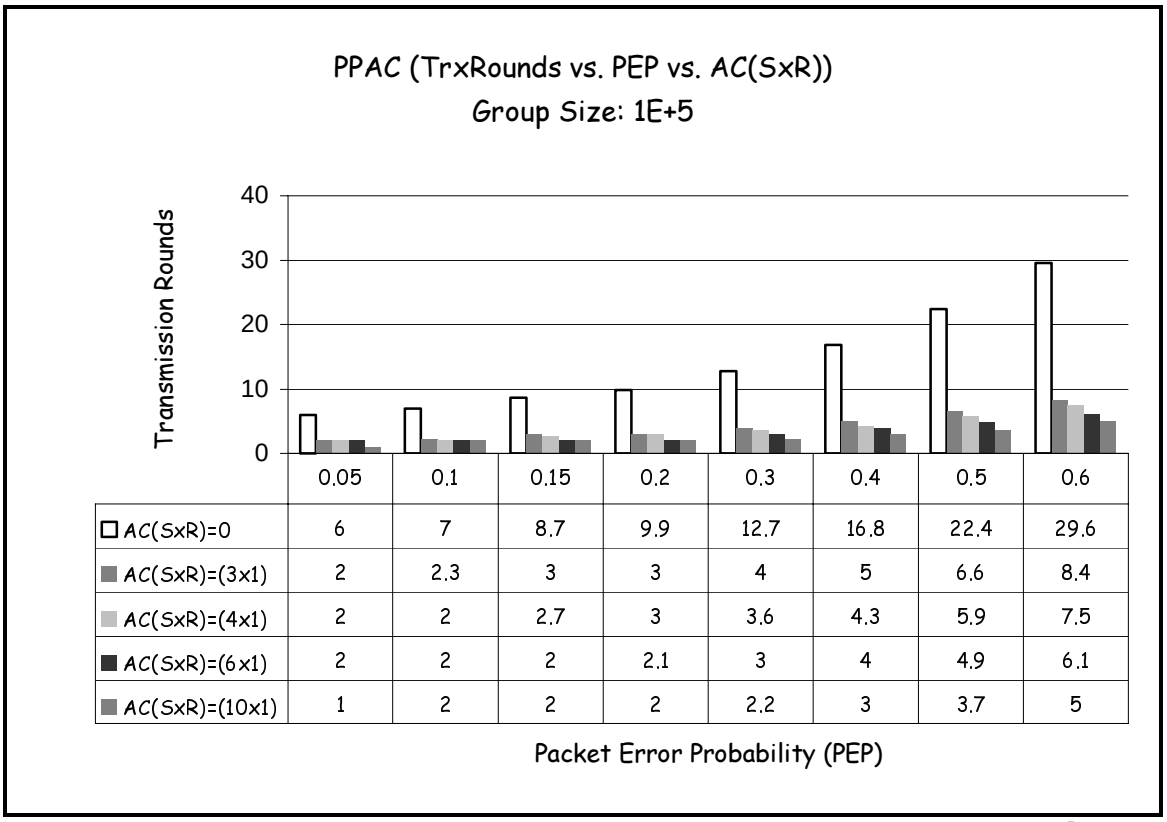


	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
No AirCache	4.87	6.05	7.23	8.48	10.98	16.8	19.25	25.64
AC(SxR)=(3x1)	2	2	2.71	3.01	3.75	4.68	5.94	7.75
AC(SxR)=(4x1)	2	2	2.09	2.89	3.06	4.06	5.13	6.8
AC(SxR)=(6x1)	1.91	2	2	2.03	3	3.24	4.1	5.49
AC(SxR)=(10x1)	1	1.69	2	2	2	3	3.11	4.14

Figure 4.11: (PPAC)Tx Rounds vs. PEP vs. AC size (TG=20, GS=10^4)

The above graph represents the error performance of the PPAC protocol for various Air Cache sizes. Obviously, the larger the Air Cache the more robust the protocol. One important point that we can deduce from the above figure is that in the case where we do not have Air Cache the rate of degradation of the performance is exponential as we increase in constant rate the packet error probability (PEP). The latter observation is not true for the case where the Air Cache size varies from 3 to 10 packets. The increase in the number of required Transmission Rounds does not follow the rate of increase of packet error probability (PEP) but instead it is much less. Also, this can be exploited from the increase of the difference between the performance of a non Air Cache ARQ-based protocol and the PPAC protocol as the PEP increases.

The same observations hold for the case where the multicast group is consisted of 10^5 members. In this case the robustness of the PPAC protocol is even more obvious compared to the previous graph (e.g., multicast group of 10^4 members). That is because of the scalability properties of the PPAC protocol, since the rate of degradation of the performance is much slower compared to the rate of increase of the multicast group size.



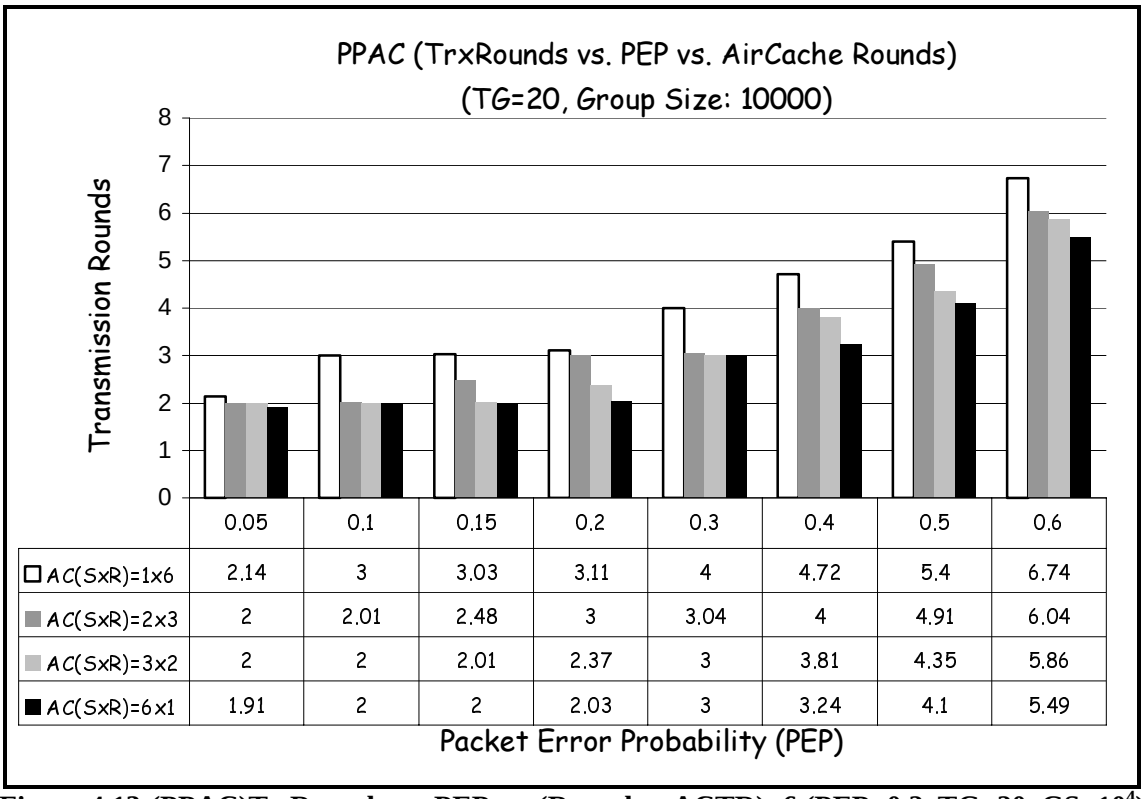
	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=0	6	7	8.7	9.9	12.7	16.8	22.4	29.6
AC(SxR)=(3x1)	2	2.3	3	3	4	5	6.6	8.4
AC(SxR)=(4x1)	2	2	2.7	3	3.6	4.3	5.9	7.5
AC(SxR)=(6x1)	2	2	2	2.1	3	4	4.9	6.1
AC(SxR)=(10x1)	1	2	2	2	2.2	3	3.7	5

Figure 4.12: (PPAC) Transmission Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

The above graph presents the results that we collected for the required Transmission Rounds for various packet error probabilities (PEPs) and for various Air Cache sizes. The multicast group was assumed to be 10^5 members and the transmission group (TG) was assumed to be 20 packets. Again, the robustness characteristics of PPAC have been proven once more through the above-presented results. Furthermore, if we compare the results that we collected for Air Cache size 0, which correspond to a non Air Cache

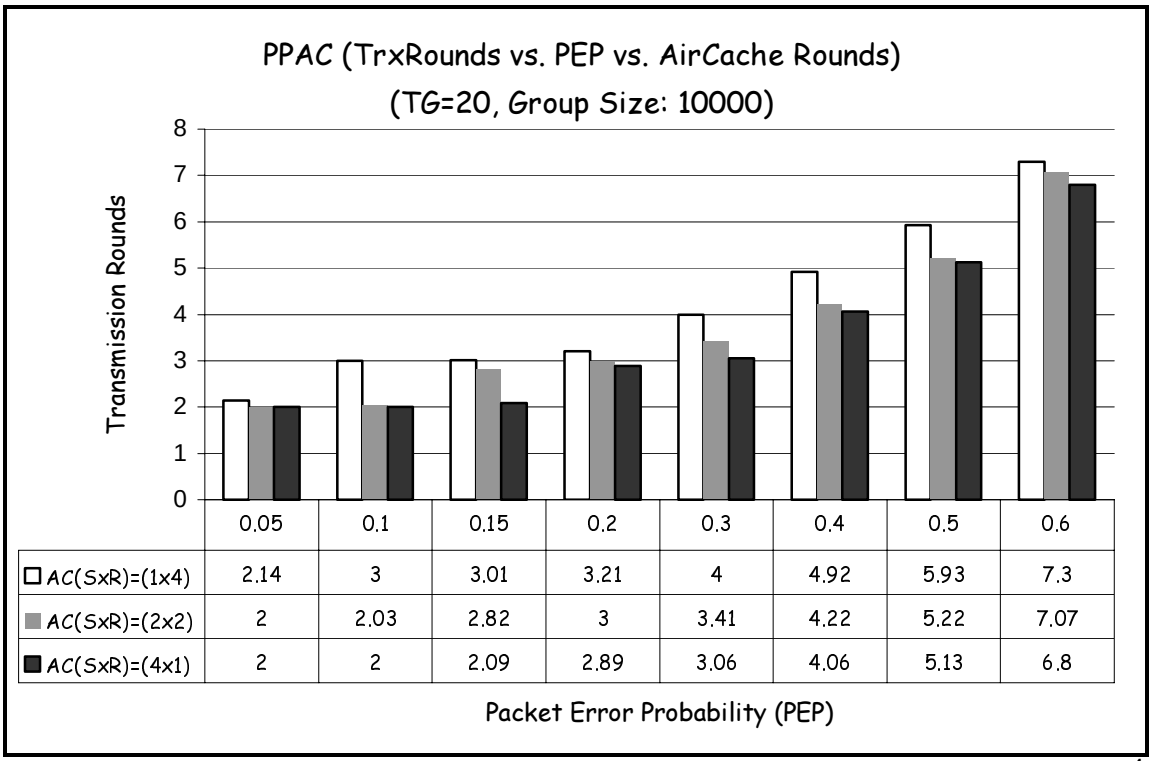
ARQ-based protocol, with the results that correspond to the PPAC protocol (e.g., Air Cache size greater than 0), the performance improvement is obvious once more and it becomes even more obvious as the packet error probability (PEP) gets higher.

The conclusions that we exploited from the two above graphs are relevant to the case where we have single Air Cache round $(TGTR = ACTR)$. The question that arises is when the error behavior is better, when we apply a single Air Cache round or multiple Air Cache rounds on the same amount of reserved bandwidth? The total size of the Air Cache measured in packets, is the product of Air Cache size of a single round multiplied by the number of Air Cache rounds. In order to find answers for the above question and for the effectiveness of using multiple Air Cache rounds, we simulated the PPAC protocol with total Air Cache size equal to 4 and 6 packets, but we varied the Air Cache size of a single Air Cache Transmission Round and the number of Air Cache rounds between the different simulated scenarios. The graphs that we will present are based on those results and are given below, for multicast groups of 10^4 and 10^5 members and for packet error probabilities (PEPs) from 0.05 up to 0.6. The TG was defined to be 20 data packets.



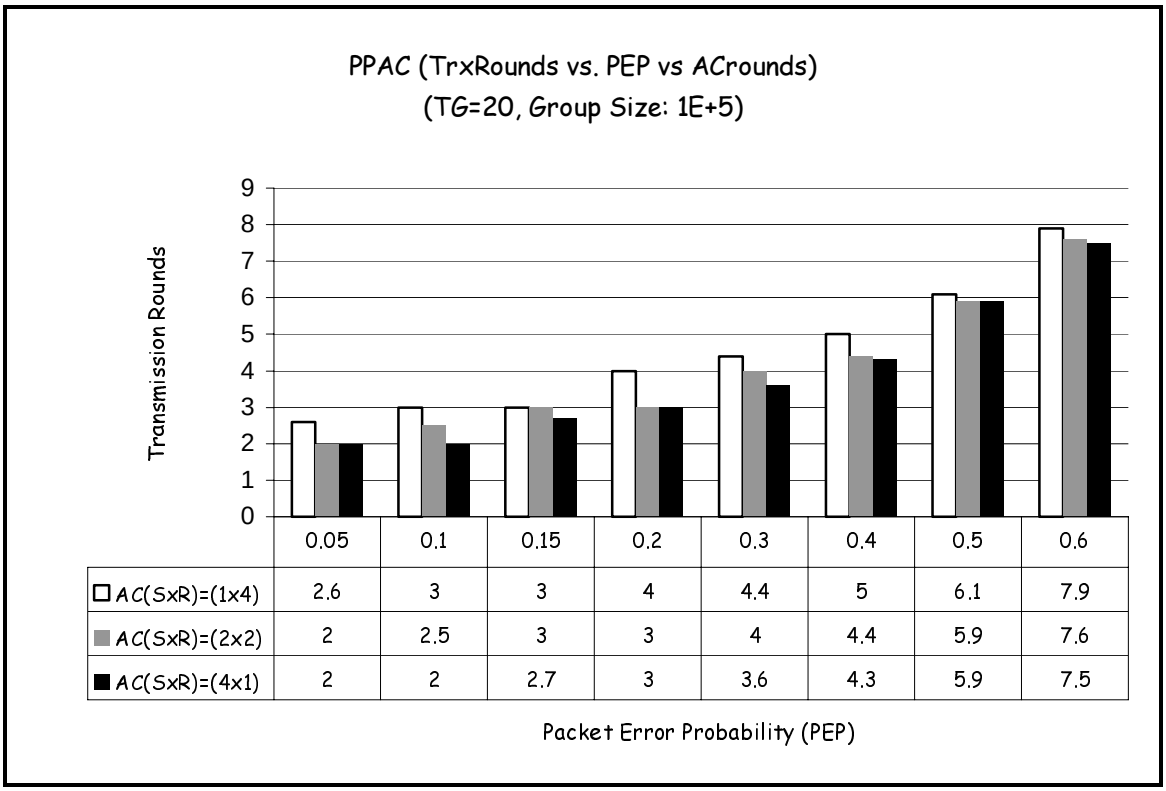
	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=1x6	2.14	3	3.03	3.11	4	4.72	5.4	6.74
AC(SxR)=2x3	2	2.01	2.48	3	3.04	4	4.91	6.04
AC(SxR)=3x2	2	2	2.01	2.37	3	3.81	4.35	5.86
AC(SxR)=6x1	1.91	2	2	2.03	3	3.24	4.1	5.49

Figure 4.13:(PPAC)Tx Rounds vs. PEP vs. (Rounds x ACTR)=6 (PEP=0.2, TG=20, GS=10^4)



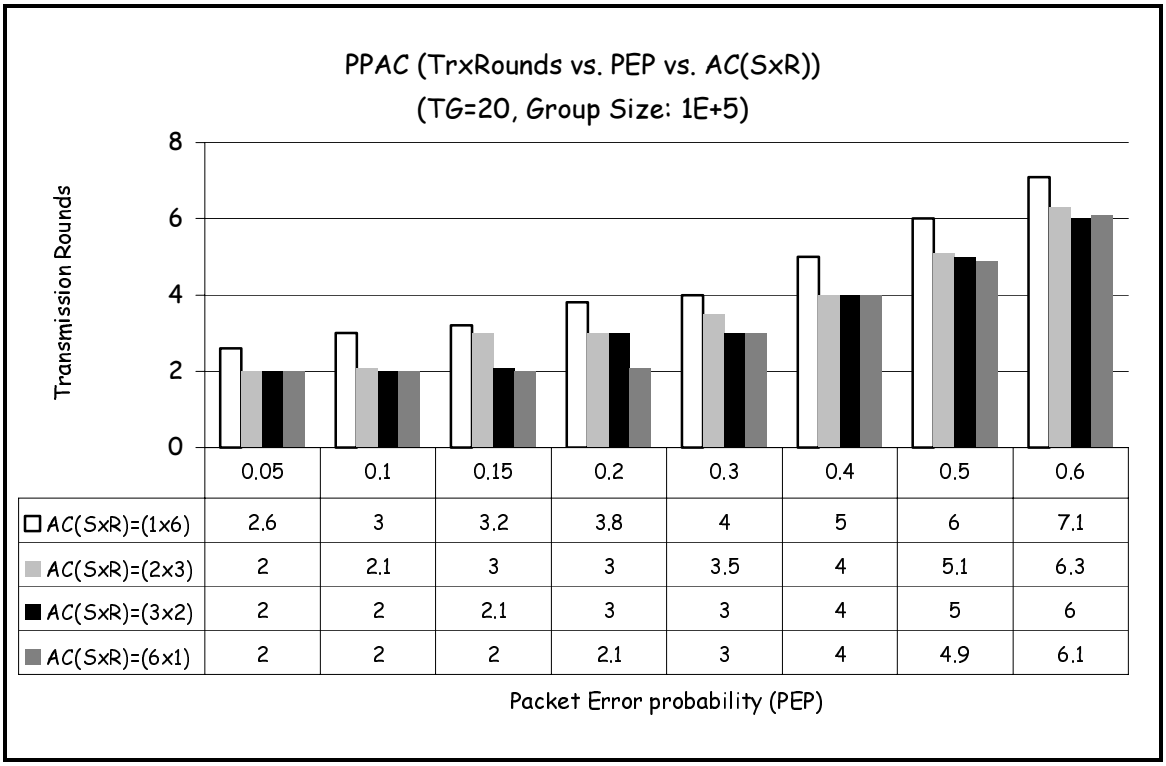
	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=(1x4)	2.14	3	3.01	3.21	4	4.92	5.93	7.3
AC(SxR)=(2x2)	2	2.03	2.82	3	3.41	4.22	5.22	7.07
AC(SxR)=(4x1)	2	2	2.09	2.89	3.06	4.06	5.13	6.8

Figure 4.14:(PPAC)Tx Rounds vs. PEP vs. (Rounds x ACTR)=4 (PEP=0.2, TG=20, GS=10^4)



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=(1x4)	2.6	3	3	4	4.4	5	6.1	7.9
AC(SxR)=(2x2)	2	2.5	3	3	4	4.4	5.9	7.6
AC(SxR)=(4x1)	2	2	2.7	3	3.6	4.3	5.9	7.5

Figure 4.15:(PPAC)Tx Rounds vs. PEP vs. (Rounds x ACTR)=4 (PEP=0.2, TG=20, GS=10^5)



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC(SxR)=(1x6)	2.6	3	3.2	3.8	4	5	6	7.1
AC(SxR)=(2x3)	2	2.1	3	3	3.5	4	5.1	6.3
AC(SxR)=(3x2)	2	2	2.1	3	3	4	5	6
AC(SxR)=(6x1)	2	2	2	2.1	3	4	4.9	6.1

Figure 4.16:(PPAC)Tx Rounds vs. PEP vs. (Rounds x ACTR)=6 (PEP=0.2, TG=20, GS=10^5)

The featured results on the above graphs, exploit the fact that among the different combinations of Air Cache Size and Air Cache Rounds for the same total Air Cache size, the best is the one with a single Transmission Round. Intuitively, we would expect this result because the Air Cache contains parity packets, which have the property to correct any of the data packets. So, transmitting a set of parity packets in multiple rounds weakens the “data healing” ability of the protocol, and the performance degrades. Although the reason of the multiple Air Cache rounds is the self-healing of the packets that are contained in the Air Cache, in the case where the Air Cache contains parity packets, the choice of multiple Air Cache rounds does not give the performance that the protocol presents when we apply a single round Air Cache.

The following paragraph provides us with general conclusions about the performance of the PPAC protocol, based on the results that were presented in this section. This analysis focuses basically on the delay and error performance of the protocol.

4.2.3.3 Performance Analysis for PPAC Protocol

Combining the various results that we got by simulating the PPAC protocol, we collected some very important observations for many aspects of the performance behavior of the protocol. Those aspects are the performance delay and the behavior of the protocol in error prone environments. Along with the latter metrics, in our analysis of the performance behavior of the protocol, we must take into consideration factors like the overhead of the protocol in terms of processing requirements, buffering requirements and messaging. The PPAC protocol seems very scalable and robust, and the delay

performance is much better compared to UDPAC and RDPAC protocols and becomes even better for high *packet error probabilities (PEPs)* and for large *multicast group sizes*.

The advantages of the PPAC protocol are obvious from the significant improvement in the required Transmission Rounds (e.g., compared to the other reliable multicasting protocols) and the very slow degradation rate of performance when the rate of increase of packet error probability (PEP) is constant. Specifying the above mentioned improvement of the required Transmission Rounds metric for large multicast groups (i.e., 10^4, 10^5 members), it is obvious that the reliable delivery of 20 data packets to a multicast group of 10^5 members is not exceeding the 3 transmissions rounds for Air Cache size 3 or more packets, and is not exceeding the 2 transmission rounds for Air Cache size 6 or more data packets (e.g., in scenario we have assumed that the packet error probability (PEP) is 0.2 and there are 20 data packets in the transmission group (TG)). Even though the metric of required Transmission Rounds does not provides us with detailed information for the end-to-end delay in actual time (i.e., *ms* or *secs*), the fact that the required Transmission Rounds are few, has as a consequence that the satellite link is not going to be accessed many times so the members of the multicast group are not going to face the propagation delay of the satellite link, many times. This has as a result that the end-to-end delay is improved using the PPAC protocol. About the error behavior of the PPAC, which is one of the most important advantages of the protocol, the collected results prove that the protocol can be used effectively in error prone environments with high packet error probabilities (PEPs). That is because the performance of the protocol is not affected much from the large increase of the packet error probability (PEP).

Furthermore, the advantages of the PPAC_EB protocol have been highlighted both in the previous and current sections, but what about the potential disadvantages of the protocol. Those disadvantages are recognized in terms of the required processing power, in terms of the required memory for buffering at the receiving devices, and in terms of the messaging overhead. The first two can be categorized as disadvantages when they are compared with the similar requirements for the UDPAC and RDPAC protocols. The increased processing and buffering requirements come from the fact that we use parity packets in the Air Cache. The processing of parity packets demands increased processing power to the receiving entities, in order to decode them and use them to correct the erroneous or corrupted data packets. Also, the generation of the parity packets at the source demands more processing power compared to the case where we do not use FEC. The demanding memory requirements for buffering, arise because of the multiple Transmission Rounds, the devices have to store both the correctly received data and parity packets in each round, in order to combine them with the error-free packets that will be received in subsequent rounds and recover all the data packets of the transmission group.

One of the potential advantages of the protocol due to its non-adaptive nature (i.e., the Air Cache size is constant and the content is parity packets), is that can be feedback free protocol under some assumptions. Those assumptions have to do with the knowledge of the group size from the source entity. So, assuming that the source entity knows the number of group members could stop transmitting data packets when it receives number of positive ACKs equal with the number of the multicast group members. For multicast groups where the membership changes happen rarely, this assumption is valid. If PPAC

could be used as a feedback free protocol then the delay performance would be improved significantly, due to the fact that there would not be feedback, so the upstream (e.g., from the multicast group members to the source) propagation delay would be eliminated. In cases, where the rate of the multicast group membership changes is high, the assumption of the group size knowledge is not valid, so in that case we are restricted to use NACKs in each round. The improvement or degradation of delay performance in either case is a topic that does not affect the required transmission rounds metric, and it needs further investigation.

In the following section we compare the three protocols we presented in this chapter, so we can get a better feeling of their relative performances (i.e., delay and error performances).

4.3 Performance Comparison of the Non-Adaptive Air Cache Protocols

This section is an overview of the results that we collected and the conclusions that we draw for the three non-adaptive reliable multicasting protocols, which we presented in this chapter. Those three protocols (i.e., UDPA, RDPAC and PPAC) can be compared in multiple ways. One very important comparison is related to the delay performance of those three protocols along with the corresponding comparison with other reliable multicasting protocols, which are based on two well-known reliable multicasting techniques, like a non Air Cache ARQ-based reliable multicasting protocol, and a non Air Cache FEC-based reliable multicasting protocol. Another, but of the same importance comparison, is the one related to their error behavior (e.g., robustness). Also, comments

can be done, by comparing the hardware and memory requirements that each protocol poses to the participating devices.

Following the above three stages for comparing the protocols, the rest of this section is separated in three parts, each one corresponding to the tree different comparisons of the protocols. The first one is related to the delay performance, the second one to the robustness and the last one to the hardware and memory requirements.

4.3.1 Comparison of Required Transmission Rounds

In this paragraph we combine the results that we collected for each one of the protocols in order to get an idea of their relative performance, by comparing them to each other and also by comparing them to the already existing protocols like a non Air Cache ARQ based protocol and a non Air Cache FEC based protocol. Already, we have presented the algorithm related to our proposed protocols, so we will briefly describe the algorithm that we used to simulate the non Air Cache ARQ-based and the non Air Cache FEC-based protocols.

The non Air Cache ARQ-based protocol uses constant bandwidth throughout the reliable transmission of the data packets, which belong to the transmission group (TG). In each Transmission Round, the protocol waits for feedback from the multicast group members. If there is a positive number of NACKs then the all the data packets contained in the transmission group are retransmitted during the next Transmission Round, until no NACKs are received from the sender. Obviously, this is a very simple and very primitive protocol and there are not high expectations about its performance.

The non Air Cache FEC-based protocol, it is expected to have much better performance but it is more complicated and more resource (e.g., battery power, processing power, memory) demanding protocol than the non Air Cache ARQ-based protocol. That is because it is based on parity packets, which need further processing for the correct functionality of the protocol (e.g., encoding/decoding). The algorithm of a FEC based protocol that we implemented follows:

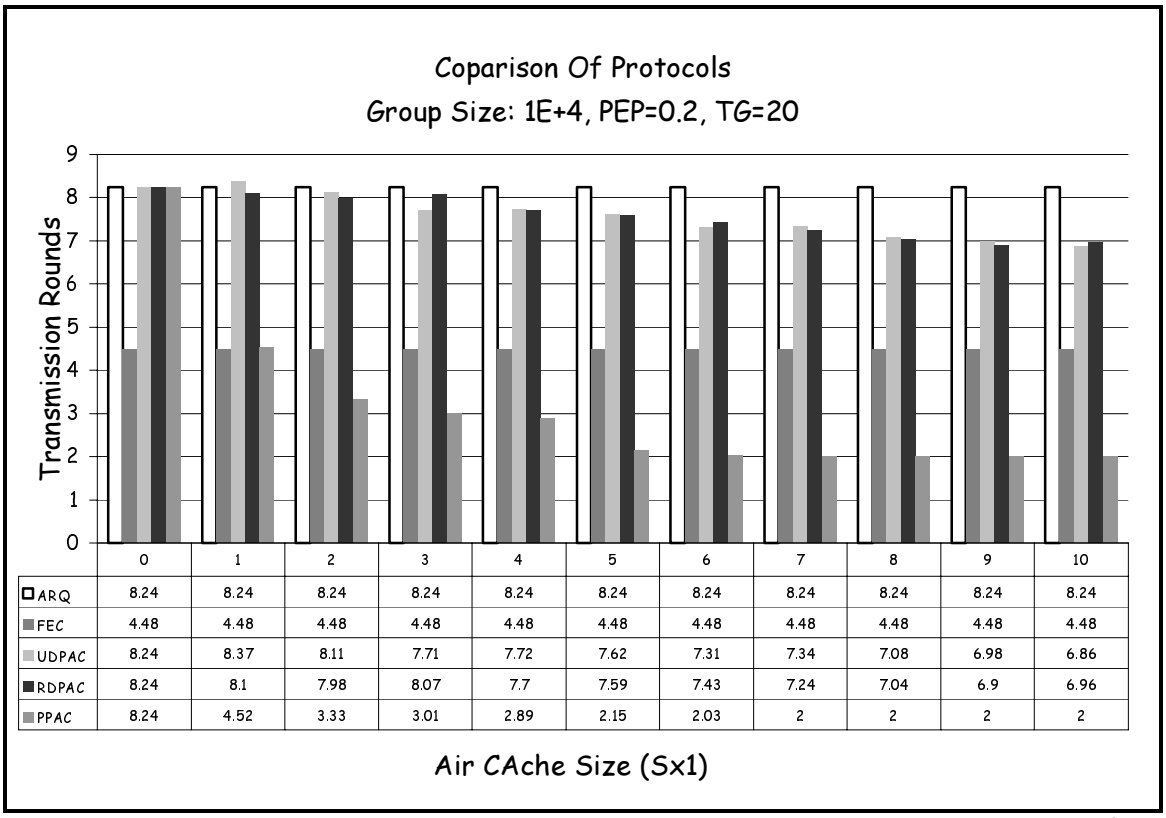
Step I: Transmit the data packets that belong to the Transmission Group, and then wait for feedback

Step II: If the received feedback contains NACKs, then instead of sending data packets, send parity packets, which have the property to correct and recover any of the erroneous or corrupted data packets. Then wait for feedback.

Step III: If there is no feedback then the reliable transmission is completed. Otherwise go back to *Step II*.

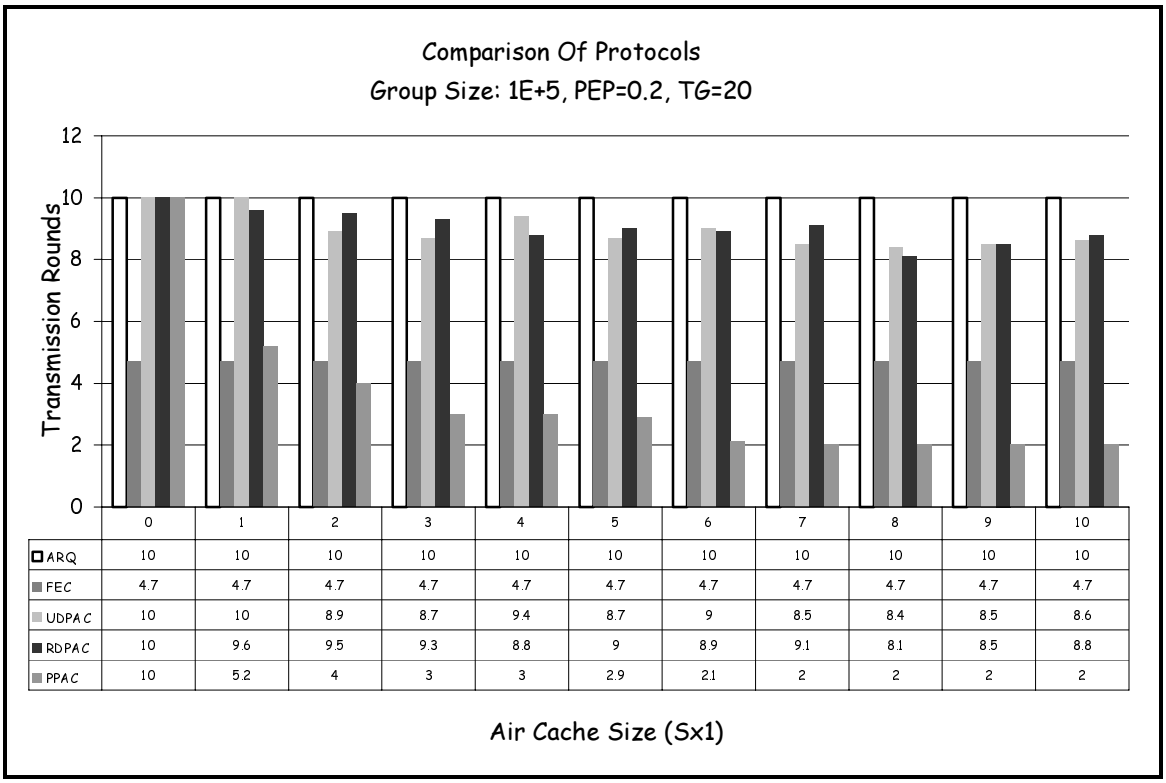
The above algorithm is expected to be very efficient in terms of its performance, due to the use of parity packets and the "healing "properties that those "special" packets have.

The comparison of the five protocols (i.e., UDPAC, RDPAC, PPAC, non Air Cache ARQ-based, non Air Cache FEC-based) is presented in the following graphs. In order to collect the simulation results, we assumed Packet Error Probability (PEP) equal to 0.2 and transmission group (TG) size equal to 20 packets. The first figure corresponds to a multicast group of 10^4 members and the second to a multicast group of 10^5 members.



	0	1	2	3	4	5	6	7	8	9	10
ARQ	8.24	8.24	8.24	8.24	8.24	8.24	8.24	8.24	8.24	8.24	8.24
FEC	4.48	4.48	4.48	4.48	4.48	4.48	4.48	4.48	4.48	4.48	4.48
UDPAC	8.24	8.37	8.11	7.71	7.72	7.62	7.31	7.34	7.08	6.98	6.86
RDPAC	8.24	8.1	7.98	8.07	7.7	7.59	7.43	7.24	7.04	6.9	6.96
PPAC	8.24	4.52	3.33	3.01	2.89	2.15	2.03	2	2	2	2

Figure 4.17:Comparison of ARQ, FEC, UDPAC, RDPAC, PPAC (TRs vs. AC)(GS=10^4)



	0	1	2	3	4	5	6	7	8	9	10
ARQ	10	10	10	10	10	10	10	10	10	10	10
FEC	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7
UDPAC	10	10	8.9	8.7	9.4	8.7	9	8.5	8.4	8.5	8.6
RDPAC	10	9.6	9.5	9.3	8.8	9	8.9	9.1	8.1	8.5	8.8
PPAC	10	5.2	4	3	3	2.9	2.1	2	2	2	2

Figure 4.18: Comparison of ARQ, FEC, UDPAC, RDPAC, PPAC (TRs vs. AC) (GS=10^5)

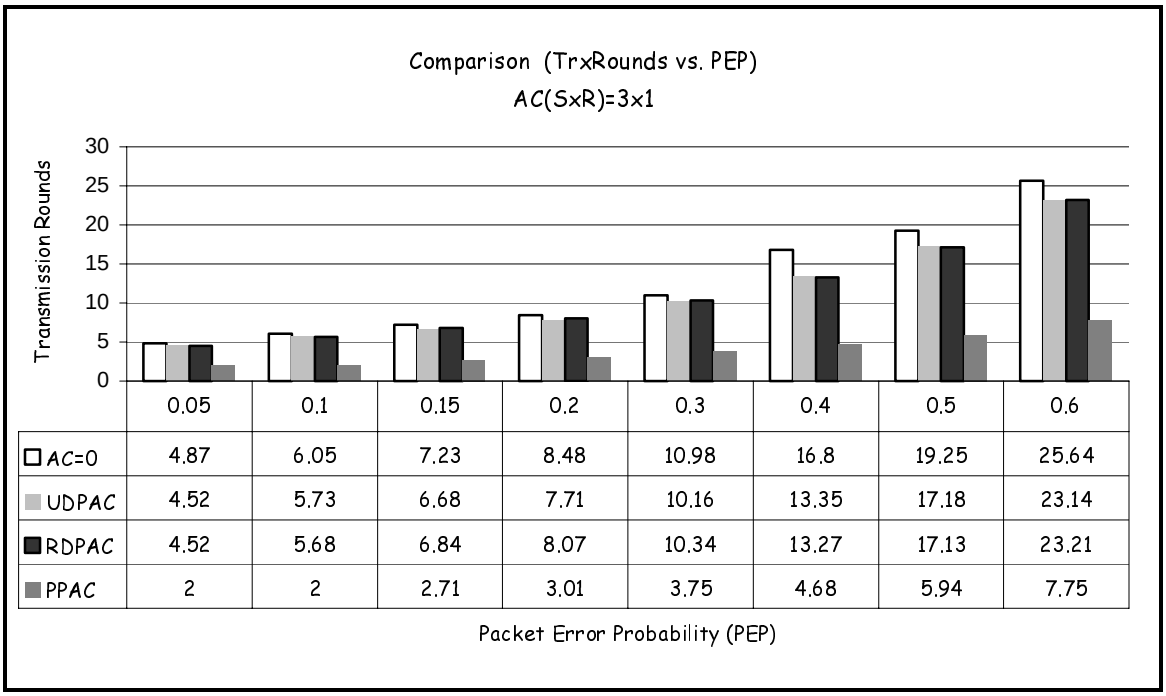
Commenting on the results that were presented on the above two graphs, the non Air Cache ARQ-based protocol has the worst performance compared to the other protocols, which is a result that was expected. The best performance comes from the PPAC protocol, which outperforms the FEC-based protocol, especially for very large multicast group sizes. The improvement in performance is almost 150% when we apply Air Cache size of 6 parity packets and that is a significant improvement with a small increase in the bandwidth usage. One general conclusion is that the parity based protocols (non Air Cache FEC-based and PPAC), outperform the non-parity based protocols and the difference in performance is significant (i.e., up to 400%). Now, if we compare the non-parity based protocols, it is obvious that their performance is not very promising and does not even comes close to the performance of FEC based protocols, but the UDPAC and RDPAC outperform the non Air Cache ARQ based protocol with percentage difference of the order of 16.3%. The UDPAC and RDPAC have almost the same performance despite the fact that the Air Cache of RDPAC protocol is updated more frequently. That is because the Air Cache in both protocols is filled with data packets that are chosen from the transmission group (TG) at random, so the refreshing rate of the Air Cache is not a crucial factor and does not make any difference. If we could use feedback to fill the Air Cache then the refreshing rate is important and makes difference on the performance results. In the next chapter that we refer to the adaptive Air Cache protocols, we explore the latter case.

Concluding this paragraph, the most important results that someone has to remember is that the PPAC protocol performs the best among the non-adaptive Air Cache protocols, and the improvements can be up to 400% compared to non-parity based protocols and up

to 150% compared to other FEC-based protocols. The reason of existence of non-parity based protocols despite of their low performance, is the fact that are simple and the hardware and memory requirements are not so demanding as in the case of FEC based protocols.

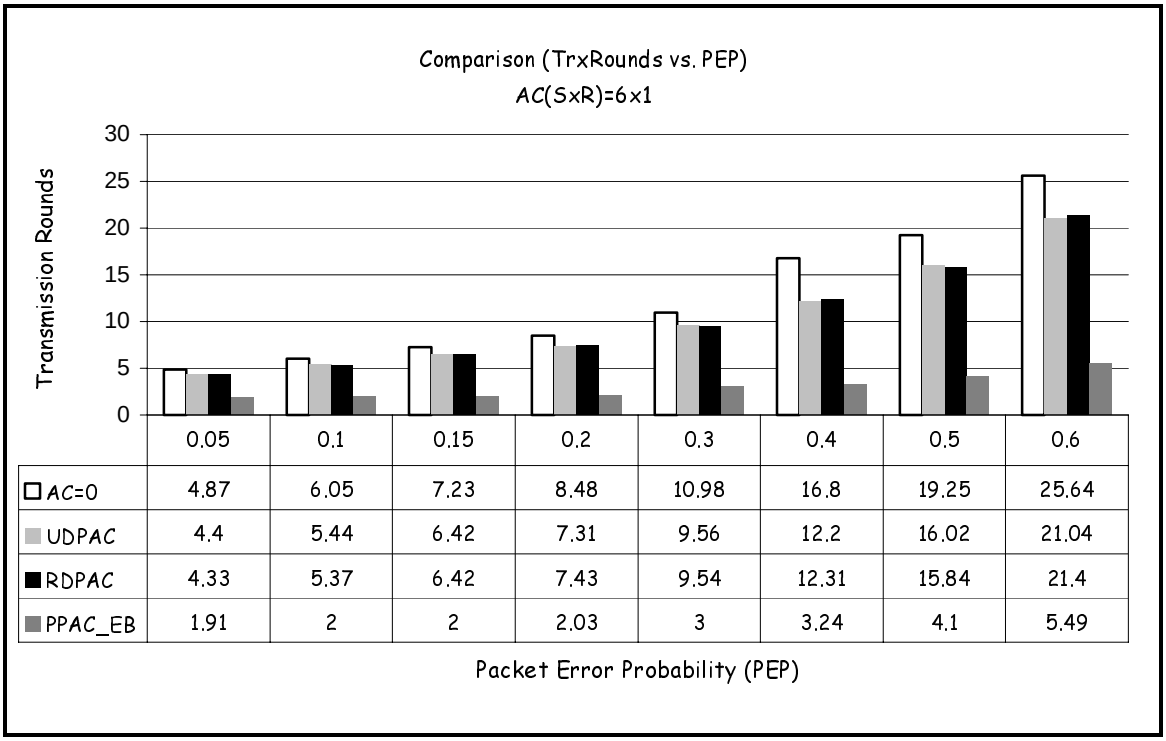
4.3.2 Error Behavior Comparison of Non-Adaptive Protocols

This paragraph is dedicated to the error performance comparison of the three non-adaptive Air Cache protocols along with the error performance of the non Air Cache ARQ-based protocol. We combine results that we have already individually presented in the previous sections, in order to get an idea of how those protocols behave with respect to the packet error probability (PEP) compared to each other. The following two graphs represent results that we collected by simulating the three non-adaptive Air Cache protocols and the non Air Cache ARQ-based one, for various packet error probabilities (PEPs) that vary from 0.05 to 0.6, for transmission group size of 20 packets (TG=20), for a multicast group of 10^4 members and for Air Cache size of 3 and 6 packets respectively. The two graphs are presented below and are followed by the observations and conclusions concerning the relative error performance of the protocols.



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC=0	4.87	6.05	7.23	8.48	10.98	16.8	19.25	25.64
UDPAC	4.52	5.73	6.68	7.71	10.16	13.35	17.18	23.14
RDPAC	4.52	5.68	6.84	8.07	10.34	13.27	17.13	23.21
PPAC	2	2	2.71	3.01	3.75	4.68	5.94	7.75

Figure 4.19: Comparison of ARQ, UDPAC, RDPAC, PPAC (TRs vs. PEP) (TG=20, AC=3)



	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6
AC=0	4.87	6.05	7.23	8.48	10.98	16.8	19.25	25.64
UDPAC	4.4	5.44	6.42	7.31	9.56	12.2	16.02	21.04
RDPAC	4.33	5.37	6.42	7.43	9.54	12.31	15.84	21.4
PPAC_EB	1.91	2	2	2.03	3	3.24	4.1	5.49

Figure 4.20: Comparison of ARQ, UDPAC, RDPAC, PPAC (TRs vs. PEP) (TG=20, AC=6)

Obviously, the FEC-based Air Cache protocol (PPAC) has the best error performance and seems to be the most robust compared to the others. Observing the two graphs above, we can highlight the fact that although the required Transmission Rounds for the non-FEC based protocols are increased exponentially, in the case of PPAC protocol the rate of increase of the corresponding metric is much less compared to the UDPAC, RDPAC and non Air Cache ARQ-based protocols. Comparing the error performance of the non-FEC based protocols, the UDPAC and RDPAC retain a slightly better performance from the non Air Cache ARQ-based protocol, but the rate of increase of the required Transmission Rounds metric is almost the same. The latter fact provides us with the expected conclusion that the UDPAC and RDPAC protocols behave the same with the non Air Cache ARQ-based protocol as the packet error probability (PEP) increases. That is because the RDPAC and UDPAC protocols are based on the ARQ error recovery technique, with the extra addition of the data Air Cache, which results in this slight improvement in performance.

4.3.3 Hardware and Memory Requirements of the Protocols

Throughout the presentation of the proposed protocols we referred on the hardware and memory requirements that those protocols pose to the participating devices, as part of their advantages or disadvantages. In this paragraph we focus on those requirements, which provides us with some extra information, such as in which network environment each one of the proposed protocols could be used. Those comments are useful from the point of view that many proposed reliable multicasting protocols fail when they come to the phase of practical implementation.

The RDPA and UDPAC protocols are non-FEC based protocols, so the received packets beyond the usual Cyclic Redundancy Check (CRC check), do not require decoding or further processing. So, the receiving devices are not required to have high processing power, beyond the necessary for handling the network communications. On the other hand, PPAC is a FEC-based protocol, so the involved parity packets require further processing compared to the ARQ-based protocols, where only data packets are involved in the transmission. The receiving devices need to have extra processing power for the processing of the parity packets (i.e. the parity packets are decoded along with the correctly received data packets) in an acceptable amount of time. The extra processing power that is required by the FEC-based protocols is significant compared to the power that RDPAC and UDAPC use to perform the protocol's functions. Also, the sender entity is required to have extra processing power to produce the parity packets in an acceptable amount of time otherwise the delay of the parity packets' production process, because of the lack of processing power, is going to be added to the end-to-end delay. Obviously, the PPAC protocol is not going to be as efficient as it looks like to be in environments where the receiver devices do not have the appropriate processing power (like networks of palm devices). That was the reason of proposing protocols like UDPAC and RDPAC, which sacrifice performance for the protocol's simplicity and low processing power requirements. Even though those protocols do not have nice performance characteristics as the FEC-based protocols, they present better performance than a non Air Cache ARQ-based protocol.

Furthermore, except the processing power requirements there are also the memory requirements from each one of the protocols. The memory is required for buffering

purposes because all the protocols (UDPAC, RDPAC, PPAC) operate in multiple transmission rounds and in each round the buffered packets from the previous ones are used, in order all the packets that belong to the transmission group (TG) to be collected or recovered. In RDPAC and UDPAC protocols the correctly received packets that have been collected in each transmission round are buffered until the whole transmission group (TG) has been received correctly. After the correct reception of the TG, the buffer space is freed in order to be used for the new reliable transmission. In the case of PPAC, the buffer requirements are higher, due to the fact that except from the correctly received data packets, the correctly received parity packets have to be buffered, too. Obviously, the Air Cache FEC-based protocol (PPAC) has more memory requirements compared to the RDPAC and UDPAC protocols. The last observation along with the one about the processing power requirements, provides us with the conclusion that even though RDPAC and UDPAC are protocols that do not have very good performance compared to FEC-based protocols, they can be used in network environments where the communication entities are lightweight in terms of processing power and memory specifications. The PPAC protocol can be used in network environments where the devices are more powerful and have more available memory and the result of its use will be the high level performance of the protocol. So, the choice of what reliable multicast protocol to be used is a trade off between the required performance and the specifications of the participating network entities.

Chapter 5: Adaptive Air Cache RM Protocols

5.1 Introduction

In this chapter we introduce the idea of adaptive Air Cache protocols. After we had presented in the previous chapter the non-adaptive Air Cache protocols and their performance characteristics, we reached in some general conclusions about these protocols. The RDPAC and UDPAC protocols do not perform well because of the random selection of the data packets that will consist the Air Cache. The PPAC protocol even though it seems to have very promising performance characteristics, due its static management of the network resources that it reserves (i.e., extra bandwidth dedicated to the Air Cache) it seems that utilizes more than the necessary bandwidth for air caching, in order to achieve this performance. In order, to improve the performance of the data Air Cache protocols (RDPAC and UDPAC) and to minimize the unnecessary bandwidth usage for all the protocols, including PPAC, we introduce the adaptive Air Cache reliable multicasting protocols, which are modified versions of the existing non-adaptive Air Cache reliable multicasting protocols.

5.2 Overview of the Proposed Adaptive Air Cache Protocols

The reason for the design of the adaptive Air Cache protocols becomes obvious from the previous paragraph. Our goals are to improve the performance of the data Air Cache reliable multicasting protocols, to minimize the extra bandwidth that is used for air caching, and to preserve the performance that the parity Air Cache protocols present, but

by minimizing at the same time the extra bandwidth that is used for air caching. There are three classes of adaptive Air Cache reliable multicasting protocols. The classification is based on which parameters of the Air Cache are adapted. We can adapt the size of the Air Cache (i.e., minimization of the bandwidth dedicated to the Air Cache) or the content of the Air Cache (i.e., improve the delay performance) or beyond the individual adaptation of size or content, there is the case of adapting both in order to get the best performance along with the minimal required usage of the network resources. The three classes of adaptive Air Cache reliable multicasting protocols are:

- Content Adaptation of Air Cache Reliable Multicasting Protocols
- Size Adaptation of Air Cache Reliable Multicasting Protocols
- Hybrid Adaptation of Air Cache Reliable Multicasting Protocols

For each one of the above classes we propose a reliable multicasting protocol, we explore its performance behavior and we compare it with the corresponding static Air Cache reliable multicasting protocol. The three protocols that we present here are:

- Adaptive Content Data Air Cache (ACDAC) protocol

The corresponding static Air Cache protocols of ACDAC protocol are the RDPAC and UDPAC. Using the feedback in each transmission round we adapt the content of the Air Cache, in contrast to the UDPAC and RDPAC protocols where the Air Cache is filled by randomly selecting data packets from the transmission group (TG). Our goal is to design a protocol that has better performance than the RDPAC and UDPAC protocols, and we expect that ACDAC is going to be such a protocol

- Adaptive Size Parity Air Cache (ASPAC)

The corresponding static Air Cache protocol of ASPAC protocol is the PPAC protocol. Both protocols use parity packets to fill the Air Cache in each transmission round. The adaptation happens in terms of the size of Air Cache. So, the goal here is to design the adaptive version of PPAC, which will present the same performance levels, with the minimal required extra bandwidth usage.

- Hybrid Adaptive Data Air Cache (HADAC)

The corresponding static Air Cache protocols of HADAC protocol are the UDPAC and RDPAC protocols. The adaptation takes place on both the content and the size of the Air Cache. Actually, HADAC is an extended version of ACDAC because in HADAC beyond the content adaptation, we adapt also the size of the Air Cache. Our goal is to improve the delay performance of the data Air Cache protocols along with the conservation of bandwidth resources.

The above protocols belong to the adaptive Air Cache reliable multicasting protocols. Their detailed description and their performance evaluation follows.

5.3 Adaptive Content Data Air Cache (ACDAC)

The Adaptive Content Data Air Cache (ACDAC) protocol is the adaptive version of the UDPAC and RDPAC protocols. The similarity of ACDAC with those static Air Cache reliable multicasting protocols exists because ACDAC uses data packets from the transmission group (TG) to fill the Air Cache (e.g., like UDPAC and RDPAC, ACDAC is an ARQ-based protocol). The difference arises because UDPAC and RDPAC fill their Air Cache with data packets chosen at random from the transmission group, but ACDAC

takes advantage of the feedback from the receivers and fills the Air Cache with the data packets, which are the most requested in each transmission round. Intuitively, we expect that the performance of the static Air Cache protocols is going to be improved significantly because in ACDAC the data packets that fill the Air Cache are not chosen randomly but the data packets that have been corrupted the most at the receivers are the most favorable to consist the Air Cache contents that will be transmitted in the next round. Obviously, ACDAC is characterized as adaptive because the content of Air Cache is chosen based on the feedback from the receivers in each transmission round. The way that the protocol utilizes the feedback and fills the Air Cache is presented in the next section.

Furthermore, and beyond the expected advantages in performance, the protocol has an important disadvantage compared to the corresponding static data Air Cache reliable multicasting protocols. In UDPAC and RDPAC, the Air Cache is filled at random and that is why these protocols do not require any feedback information from the receivers for that operation. On the other hand, ACDAC utilizes the feedback from the receivers in order the appropriate data packets to be selected for the Air Cache. Even though the number of required Transmission Rounds is expected to decrease, the sender has to wait for feedback from the receivers (i.e., the requests for retransmission have to be transmitted through the satellite link – large propagation delay), and in this scenario we face also the problem of acknowledgement implosion at the receiver. An elegant way to bypass or eliminate this problem is to wait for a time window and based on the partial feedback taken so far to proceed on the Air Cache content adaptation procedure. In this study we assume that we collect all the feedback from the receivers, so the performance

results can be assumed as the upper bound of the performance that the protocol can potentially achieve.

In the following paragraph we present the algorithm of the ACDAC protocol along with the algorithm that demonstrates how the Air Cache is filled and refreshed per transmission round with data packets. The presentation of the simulation results follow and then, we analyze them in order to find out what are the strong and weak points of the protocol.

5.3.1 ACDAC's algorithm

The algorithm, which represents the functionality of the Adaptive Content Data Air Cache (ACDAC) protocol, is presented in two phases. The first phase corresponds to the communication or transport part of the protocol between the sender and the receivers and the second phase demonstrates how the ACDAC utilizes the received feedback for the refresh of the content of the Air Cache. So, the two phases that characterize the full functionality of ACDAC are:

ACDAC (Adaptive Content Data Air Cache)

(Communication Protocol)

Step I: Transmit the TG in C_1 and transmit the randomly chosen Air Cache contents in C_2.

Step K (K>1): If there are no requests for retransmission then the TG has been received correctly from each one of the multicast group members, the algorithm stops here and the number of required Transmission Rounds for the completition of the reliable transmission is *K-1*.

If there are requests for retransmission then we retransmit the TG in C_1 and the updated Adaptive Air Cache contents in C_2. How the contents of the Air Cache are refreshed per Transmission Round is described from the following algorithm, in figure 2. We substitute *K* with *K+1* and we repeat this step.

Above we presented the algorithm that corresponds to the communication part of the protocol. Obviously, the above algorithm is similar to the algorithm that describes the UDPAC and RDPAC protocols. The difference of ACDAC from the latter non-adaptive protocols, which characterizes also the adaptive properties of ACDAC, has to do with the refresh of the Air Cache with data packets from the transmission group (TG), based on the feedback from the receivers. The algorithm for the update of the Air Cache follows.

Rules for Updating the Air Cache Content

Step I: The Air Cache is filled with randomly selected data packets from the TG.

Step K (K>1): Taking into consideration the feedback, the Air Cache is not anymore consisted from randomly selected data packets (e.g., like in UDPAC or in RDPAC), but there is an algorithm that specifies the selection process. Assume that the number of distinct requested data packets is *DRQpacks* and the Air Cache size per transmission round is *ACsize*. There are two different scenarios that can be applied for the update of the contents of the Air Cache. The selection of the scenario to be applied is based on the comparison of *DRQpacks* with the *ACsize*.

Scenario I: if ($DRQpacks \geq ACsize$) then we select the *ACsize* packets that have been most requested for retransmission from the members of the multicast

group and we fill the Air Cache of the following transmission round with those.

Scenario II: if (*DRQpacks* < *ACsize*) then we fill the Air Cache with the *DRQpacks* requested for retransmission packets. The remaining (*ACsize -DRQpacks*) spots in the Air Cache are filled with the (*ACsize – DRQpacks*) most requested packets in the current round. In this case, we could resize the Air Cache but we have assumed that in ACDAC protocol we adapt only the content an not the size of the Air Cache. So, the size of the Air Cache remains the same throughout the reliable transmission.

The above described two phases of the algorithm present the functionality of the ACDAC protocol in detail. In the following paragraphs we are going to simulate the described protocol in order to evaluate it in terms of its delay performance and its robustness. Intuitively, we expect that ACDAC is going to have better performance behavior than UDPAC and RDPAC but we do not expect that this protocol is going to behave better than any FEC based protocol because it uses data packets for the recovery of the corrupted or erroneous data packets at the receivers (e.g., as we defined ACDAC protocol is an ARQ-based protocol). The effectiveness of an ARQ-based protocol is limited (i.e., each data packet can be correct only by retransmitting reliably itself) compared to the effectiveness of the FEC-based protocols, where the corrupted or erroneous data packets can be potentially recovered by any of the correctly received parity packets.

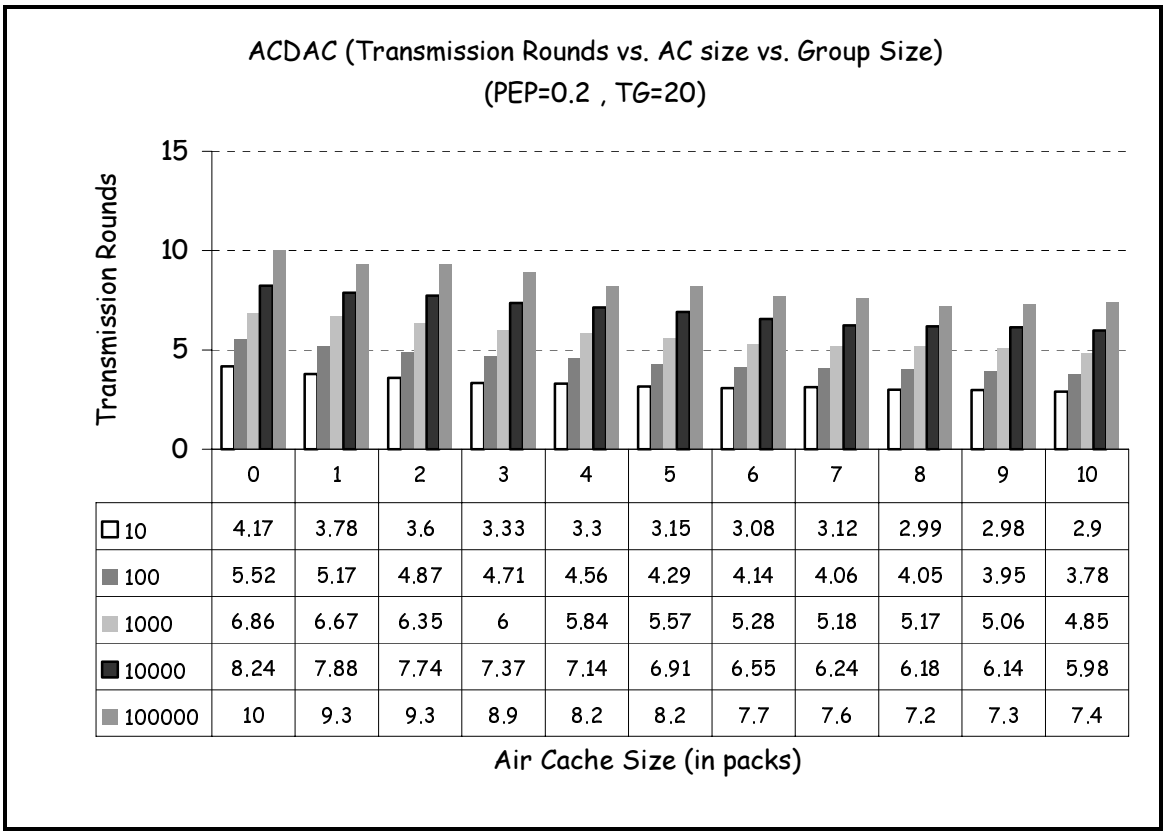
5.3.2 Simulation Results

In this section we present the results that we got after simulating the protocol described above (e.g., ACDAC). It is important to answer the following questions based on the collected results from the simulation of ACDAC:

- What is the delay performance of the protocol measured in *required Transmission Rounds* for the reliable delivery of a set (TG) of data packets?
- Is the improvement in performance of the protocol adequate enough to compensate for the extra bandwidth, which is reserved for air caching, compared to the performance of the non Air Cache ARQ-based reliable multicasting protocols?
- How robust is the protocol in error prone environments?

In this section we present the appropriate results in order to answer those questions. By the end of the next section we will be able to answer the above questions and draw general conclusions about the effectiveness of the protocol and its advantageous and non-advantageous characteristics.

The first set of results is relevant to the delay performance of the protocol, and more specifically to the number of *Required Transmission rounds* for the reliable delivery of a set of data packets (TG) to each one of the members of the multicast group. The following graph represents the behavior of the *required Transmission Rounds* metric for various group sizes and various Air Cache sizes. The Air Cache varies from 0 to 10 packets and the size of the multicast group from 10 to 10^5 members. The following results were collected for packet error probability (PEP) equal to 0.2 while the size of the transmission group was assumed to be 20 data packets. So the results are:



	0	1	2	3	4	5	6	7	8	9	10
10	4.17	3.78	3.6	3.33	3.3	3.15	3.08	3.12	2.99	2.98	2.9
100	5.52	5.17	4.87	4.71	4.56	4.29	4.14	4.06	4.05	3.95	3.78
1000	6.86	6.67	6.35	6	5.84	5.57	5.28	5.18	5.17	5.06	4.85
10000	8.24	7.88	7.74	7.37	7.14	6.91	6.55	6.24	6.18	6.14	5.98
100000	10	9.3	9.3	8.9	8.2	8.2	7.7	7.6	7.2	7.3	7.4

Figure 5.1: (ACDAC) Transmission Rounds vs. AC Size vs. Group Size (PEP=0.2, TG=20)

Obviously, the performance of the protocol does not improve as we increase the Air Cache size. Furthermore, if we compare the results that we collected for no Air Cache (e.g., the Air Cache size equals to 0 packets), to the results that correspond to Air Cache of size 10 packets, the difference is not significant even for large group sizes. Note here that the corresponding results for Air Cache size 0 (no Air Cache) correspond to a non Air Cache ARQ-based protocol. The improvement in performance is not more than 33% for an increase of 50% in bandwidth usage. So, we suspect, based on those results, that the protocol uses inefficiently the extra bandwidth in order to improve its performance. Of, course these results were expected based on the fact that ACDAC is not an FEC-based protocol and it just uses plain data packets from the TG to correct the

corresponding erroneous or corrupted packets at the receiving end, like an ARQ-based protocol does.

The question that arises at this point is how we can determine if the improvement in performance compared to the performance of a non Air Cache ARQ-based protocol is sufficient to compensate for the extra bandwidth usage. So, for fairness in comparison between the different protocols, we need a metric that combines the performance of the protocol and the bandwidth that the corresponding protocol uses. Following the same procedure as we did for the class of non-adaptive Air Cache protocols in the previous chapter, we are going to use the *Normalized Gain* metric. The formula of the metric is the same as the one already presented in the previous chapter and is given from the following ratio:

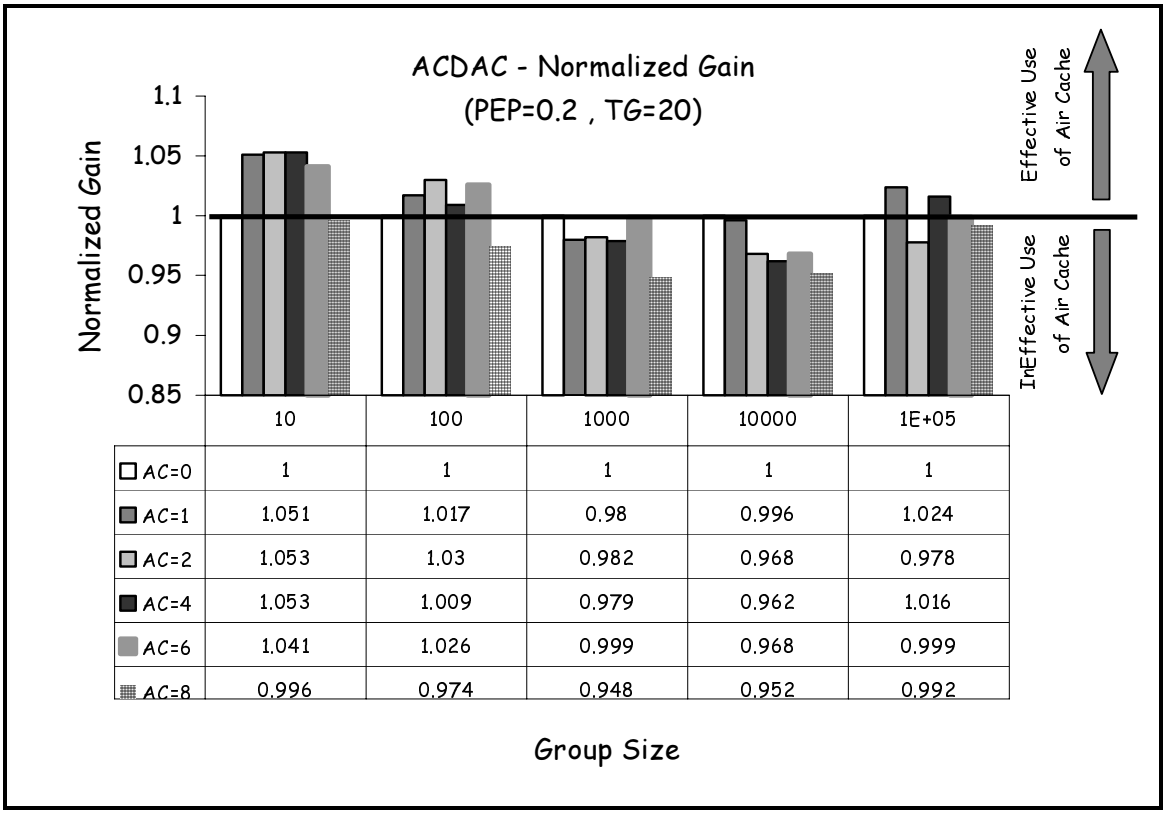
$$NormalizedGain = \frac{TransmissionRounds_{nonAirCache-ARQ} * TGsize}{TransmissionRounds_{ACDAC} * (TGsize + ACsize)}$$

The results for Air Cache of size 0 packets correspond to a non Air Cache ARQ-based protocol, as it is obvious from the communication algorithm of ACDAC, when we eliminate the transmission in C_2 channel (e.g., Air Cache). So, we are using those results (e.g., Air Cache of size 0 packets) for the numerator of the Normalized Gain ratio. The condition, which determines the efficiency of the protocol in terms of the utilization of the extra bandwidth, is:

$$\boxed{Normalized\ Gain \geq 1}$$

The above condition is self explanatory, since we want the product of $B \times T$ (e.g., *Bandwidth* x *required Transmission Rounds*) for the ACDAC protocol with $Air\ Cache\ size > 0$ packets to be less than the corresponding product of the non Air Cache ARQ-based (Air Cache size = 0 packets) protocol. We require the above condition to hold because we want the improvement in performance to be sufficient enough to compensate for the bandwidth reserved for air caching.

Applying the above ratio and utilizing the results that we collected for the *required Transmission Rounds* metric, we present in the following graph the Normalized Gain ratio values that we computed for various Air Cache sizes (e.g., 0 to 10 packets) and for various multicast group sizes (e.g., 10 to 10^5 members). The packet error probability (PEP) is defined to be equal to 0.2 and the TG is consisted of 20 data packets in each one of the simulated scenarios for the collection of the following results.



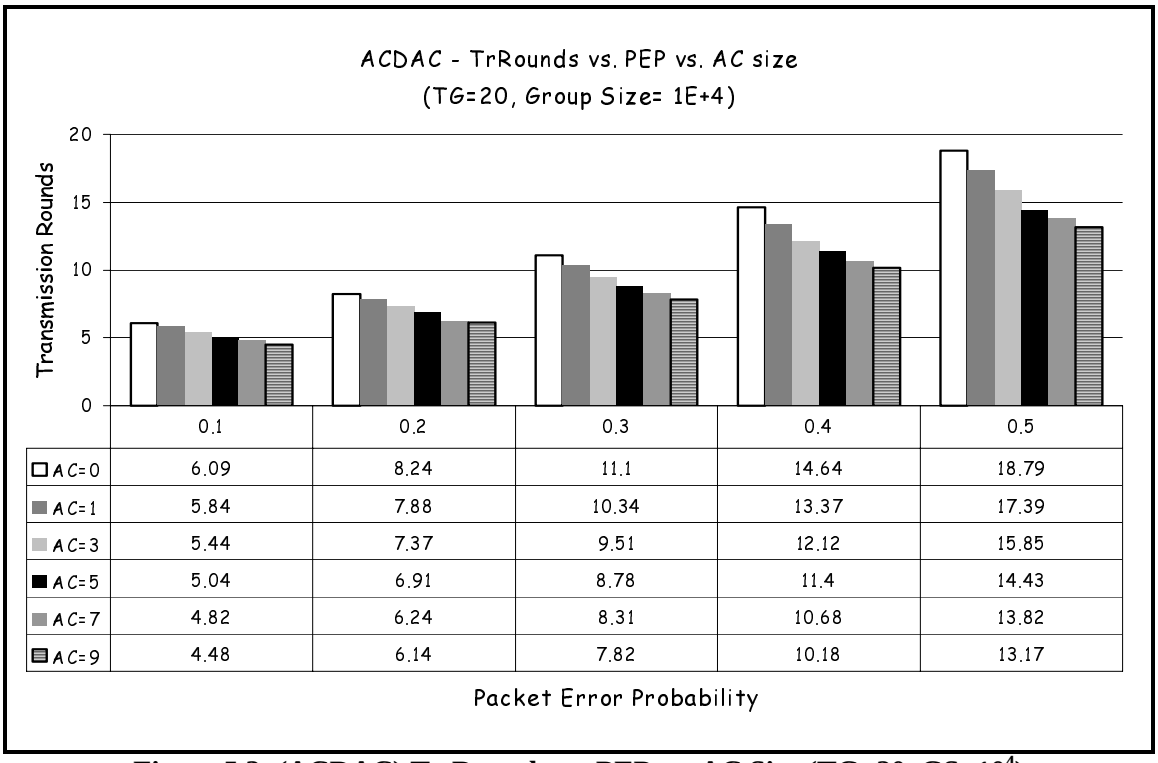
	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	1.051	1.017	0.98	0.996	1.024
AC=2	1.053	1.03	0.982	0.968	0.978
AC=4	1.053	1.009	0.979	0.962	1.016
AC=6	1.041	1.026	0.999	0.968	0.999
AC=8	0.996	0.974	0.948	0.952	0.992

Figure 5.2: (ACDAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

Taking into account both the above results and the above condition concerning the Normalized Gain ratio, it is obvious that ACDAC is not using very efficiently the extra bandwidth that reserves for air caching. On the other hand the computed Normalized Gain ratios are fluctuating around the bound value of one for the various simulated scenarios (i.e., various group sizes and various Air Cache sizes). Based on the latter observation and on the fact that ACDAC protocol improves the required Transmission Rounds metric compared to a non Air Cache ARQ-based protocol, we can conclude that ACDAC can be of practical use. Of course, the improvement in performance is not sufficient enough to compensate 100% for the extra bandwidth, but the protocol has other advantages such as the lightweight hardware requirements (e.g., processing power, memory for buffering) that poses to the participating network entities, compared to the corresponding requirements of the reliable multicasting protocols that are FEC-based.

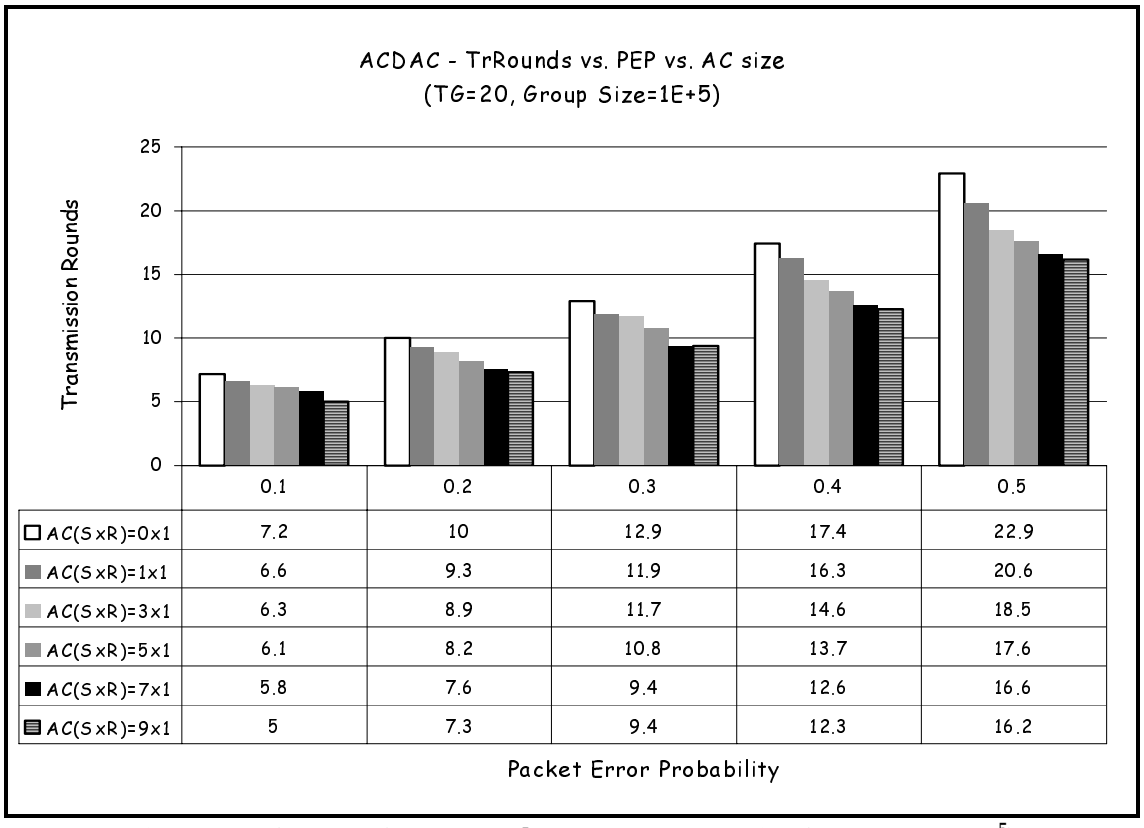
Furthermore, we want to explore the robustness of ACDAC in error prone environments. So, in order to evaluate the protocol's performance in terms of the *required Transmission Rounds* for various packet error probabilities (PEP), we simulate the protocol for packet error probabilities (PEPs), which vary from 0.1 up to 0.6 for each transmitted packet. The results are shown to the graphs below (the first graph corresponds to multicast group size of 10^4 members and the second graph corresponds to multicast group size of 10^5 members). By observing those two graphs, which represent the error behavior of the protocol for various multicast group sizes, we draw the conclusion that the ACDAC protocol is not very robust, since the rate at which the *required Transmission Rounds* metric increases as we increase in constant rate the PEP, is nearly exponential. The latter statement is even more obvious from the second graph, which

represents the case where the multicast group has 10^5 members. This result was expected due to the fact that ACDAC is not a FEC-based Air Cache reliable multicasting protocol. So, even though the Air Cache content is selected based on feedback, as opposed to RDPAC and UDPAC protocols, the performance of the protocol in error prone environments degrades fast as the packet error probability (PEP) gets higher. Beyond the latter general conclusion for the robustness characteristics of the protocol, we need to compare it in terms of the error performance with the corresponding non-adaptive Air Cache reliable multicasting protocols, in order to get an idea of how the non-random selection of the data packets that constitute the Air Cache, affects the performance of the protocol. We have to do this study because if the difference in performance is not significant then there is no need to wait for feedback from the sender, a choice that will affect the timing delay of the end to end reliable delivery of the transmission group (TG). That study will be presented in a subsequent section, where we compare the adaptive and non-adaptive Air Cache reliable multicasting protocols.



	0.1	0.2	0.3	0.4	0.5
AC=0	6.09	8.24	11.1	14.64	18.79
AC=1	5.84	7.88	10.34	13.37	17.39
AC=3	5.44	7.37	9.51	12.12	15.85
AC=5	5.04	6.91	8.78	11.4	14.43
AC=7	4.82	6.24	8.31	10.68	13.82
AC=9	4.48	6.14	7.82	10.18	13.17

Figure 5.3: (ACDAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^4)



	0.1	0.2	0.3	0.4	0.5
AC(SxR)=0x1	7.2	10	12.9	17.4	22.9
AC(SxR)=1x1	6.6	9.3	11.9	16.3	20.6
AC(SxR)=3x1	6.3	8.9	11.7	14.6	18.5
AC(SxR)=5x1	6.1	8.2	10.8	13.7	17.6
AC(SxR)=7x1	5.8	7.6	9.4	12.6	16.6
AC(SxR)=9x1	5	7.3	9.4	12.3	16.2

Figure 5.4: (ACDAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

Since we have presented the simulation results that correspond to ACDAC, we will comment further those results and will draw general conclusions about the performance behavior of the protocol. Some of those conclusions have already been mentioned on this section but in the following one we will put everything together in order to characterize and evaluate the protocol based on its operation characteristics, its hardware requirements and its performance results.

5.3.3 Performance Analysis

In this paragraph we evaluate and comment on the results demonstrated above. The following general conclusions about the performance behavior of the protocol have been stated briefly in the previous section as short comments on the graphs presented.

The performance behavior of the protocol in terms of the *required Transmission Rounds* is not very promising, because the performance improvement compared to the performance of a non Air Cache ARQ-based protocol is not significant to compensate for the extra bandwidth dedicated to air caching. The improvement is as high as 2 Transmission Rounds in the case, where we have 10^5 members in the multicast group, the packet error probability (PEP) is 0.2, the transmission group (TG) size is 20 packets and the Air Cache size is 10 packets, which in this scenario corresponds to the 50% of the bandwidth used for the data packets transmission. The improvement in performance in the latter case compared to the large amount of extra bandwidth used is neither significant nor sufficient to compensate for the reservation of this bandwidth for air caching. Also, the latter conclusion is obvious from the values of the Normalized Gain ratio, which are not always greater than one, but fluctuate between 0.95 and 1.05.

About the robustness characteristics of the protocol, we have already made some very general comments, which include also the fact that the ACDAC's performance in error prone environments is not very promising, since the increase in the required Transmission Rounds follows almost an exponential pattern, as the PEP gets higher.

On the other hand, we have to highlight the advantages of the protocol because this protocol was not designed to have the best performance but was designed to be lightweight in terms of the hardware and memory specifications that poses to the participating network entities. Similarly to the RDPAC and UDPAC protocols, the ACDAC is not a FEC-based protocol, so it does not require extra processing capabilities for the encoding (at the sender) and the decoding (at the receivers) of the parity packets for the error recovery of the corrupted or erroneous data packets. About the memory requirements the advantages are the same as the two corresponding data Air Cache non-adaptive reliable multicasting protocols (UDPAC and RDPAC). The participating devices do not have to store extra packets (parity packets) for future processing. For example, the receiving devices is not required to have extra buffering (memory) space to store the correctly received parity packets and the sender does not have to store the parity packets, which are produced from the encoding process, until the end of the reliable transmission. So, the conclusion we draw from all the above advantages, in correlation with the fact that ACDAC has better performance compared to the other protocols of its class (i.e., data Air Cache reliable multicasting protocols or ARQ-based protocols), it is a very useful protocol for network environments where the receiving and/or sending devices are not very powerful in terms of processing power and do not have much available memory for buffering. For example, the handheld devices have characteristics like the latter ones.

Also, the processing power is related closely, to the power that a device uses, so all the wireless devices need to operate in order to consume as low power as possible, so the protocols that target on networks of those devices have to trade off the low power consumption with the processing power. The latter is one more reason that explains the importance of protocols like ACDAC.

5.4 Adaptive Size Parity Air Cache (ASPAC)

Having explored the Adaptive Content Data Air Cache protocol (ACDAC) we need to incorporate adaptation features in the protocols, which use parity Air Cache. Obviously, in the case that we use parity packets to fill the Air Cache we cannot have content adaptation due to the fact that each one of the parity packets can recover each one of the data packets. So, there is no need to adapt the content of the Air Cache because each one of the parity packets has equivalent effect on the recovery process. Another parameter that can be adapted is the size of the Air Cache. Actually, using the feedback we can transmit only the appropriate number of parity packets per transmission round, in order to utilize the minimal possible bandwidth while we try to preserve the performance of the protocol, which is a FEC-based, on high levels.

Furthermore, we have to take into consideration that this new protocol (ASPAC) is the successor of the PPAC protocol. Backtracking to the previous chapter, PPAC has very promising performance characteristics, even though it belongs to the non-adaptive class of the Air Cache reliable multicasting protocols. Our new adaptive Air Cache protocol should be designed carefully in order to retain those promising performance

characteristics in combination also with the minimal required bandwidth usage characteristics due to the Air Cache size adaptation.

The protocol is called Adaptive Size Parity Air Cache (ASPAC) in accordance with the onomatology that we have developed for the adaptive Air Cache protocols. The name characterizes two of the properties of the protocol, the first one is that the protocol uses parity packets in the Air Cache and the second one is that the parameter of the Air Cache, which is adapted using the feedback from the receivers, is the size.

In the following sections and in similar fashion with the presentation of other protocols, we are going to present the algorithm of ASPAC protocol, then we will give out sets of simulation results concerning the performance of the protocol and finally, we will analyze the results in order to gain further insight about the advantages and disadvantages of the protocol. In the following section we start the exploration of ASPAC from the presentation of its algorithm.

5.4.1 Adaptive Size Parity Air Cache (ASPAC) Protocol's Algorithm

ASPAC belongs to the parity Air Cache reliable multicasting protocols like the Parity Packets in Air Cache (PPAC) protocol, which we have already presented in chapter 4. The only difference of ASPAC compared to the PPAC is the size adaptation feature of the Air Cache in ASPAC. The two protocols have a lot of similarities; one of those is located on the algorithmic level where the communication part of the protocols is almost the same. The only difference in the algorithm of those two protocols is the extra algorithm of ASPAC, which controls the adaptation of the size of the Air Cache. We present the algorithm of ASPAC in two phases, as we did with the previous adaptive Air

Cache reliable multicasting protocol. The first phase is dedicated to the communication part of the protocol and the second phase is dedicated to the size adaptation of the Air Cache, which specifies the bandwidth that will be reserved for the Air Cache per transmission round based on the feedback from the receivers. We are going to present those two phases individually, starting from the communication phase. The algorithmic description of the ASPAC protocol follows:

ASPAC (Adaptive Size Parity Air Cache)

Phase I (Communication Protocol)

Step I: The behavior of ASPAC in the initial round is the same as in PPAC protocol.

There are two concurrent transmissions going on. In C_1 we have the transmission of the TG data packets and in C_2 we have the transmission of the *max(ACsize)* parity packets. In this initial round, the Air Cache is filled with the maximum number of parity packets (e.g. *max(AC size)* – max(Air Cache size)- *max(ACsize)* is the maximum number of packets that will be allowed in the Air Cache per Transmission Group Transmission Round (TGTR), throughout the reliable transmission).

Step K (K>1): If there are no requests for retransmission then all the members of the multicast group have received the TG data packets correctly, so the algorithm stops here and the number of the required Transmission Rounds for the reliable transmission of the TG data packets is *K-1*.

Otherwise, if there are requests for retransmission, then depending on the number of the distinct requested data packets in the current round; we adapt the size of the Air Cache. How this is done is described below, where we describe the second

phase of the algorithm (e.g., refresh of the Air Cache). After the Air Cache is filled with the appropriate number of parity packets, the concurrent transmission resumes on the two channels C_1 and C_2, where the transmissions of the TG data packets and this of the Air Cache happen, respectively. We update *K* by substituting it by *K+1* and we repeat this step.

After the presentation of the communication protocol, which is not differ much from the corresponding algorithm of the PPAC protocol, we describe how the size adaptation of the Air Cache happens in each transmission round. The following part of the ASPAC's algorithm is the novel one, and establishes the ASPAC as an adaptive Air Cache protocol. So, the size adaptation of the Air Cache is done based on the following algorithm:

ASPAC (Adaptive Size Parity Air Cache)

Phase II (Size Adaptation of the Air Cache)

Step I: The Air Cache is filled with the maximum number of parity packets that can be accommodated from the Air Cache in any of the Transmission Group Transmission Rounds. We assume that this number is *max*(*ACsize)*.

Step K (K>1): If we assume that the number of distinct data packets from the TG that have been requested for retransmission in the round *K-1* is $DRQpacks_{K-1}$ and the maximum number of packets that the Air Cache can accommodate in each transmission round is *max(ACsize)*, then the size of the Air Cache in round K, which is symbolized as $ACsize_K$, is specified from the following formula:

$$ACRsize_K = \min(DRQpacks_{K-1}, \max(ACsize))$$

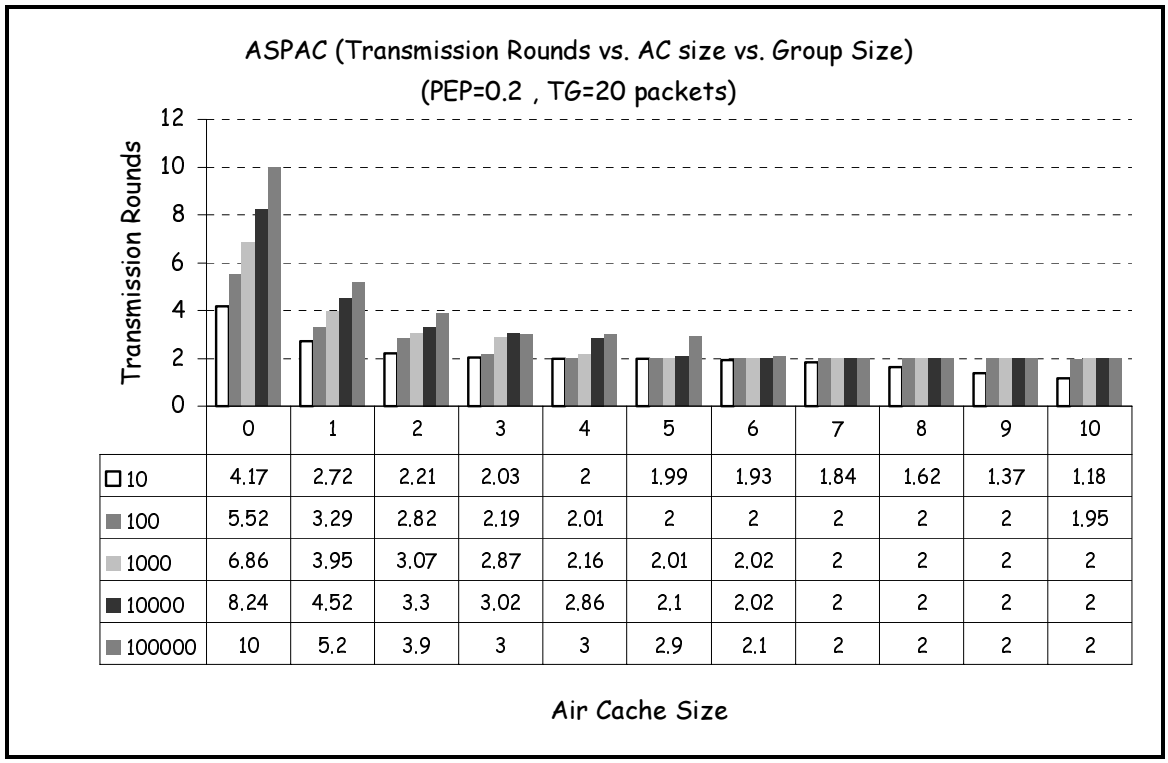
We followed the above algorithm in order to simulate ASPAC. The details of the simulator were described in chapter 3, and the results that we collected about ASPAC are presented in the following section. Actually, our goal is to evaluate the protocol in terms of its performance, its efficiency on the utilization of the available bandwidth and its robustness.

5.4.2 Simulation Results

Using the same methodology with the one that we followed when we were presenting the simulation results of the previous protocols, we will demonstrate the results that we collected by simulating the Adaptive Size Parity Air Cache (ASPAC) protocol. Our goal is to evaluate the protocol mainly in terms of its delay characteristics and its robustness, and also we want to examine if the protocol utilizes efficiently the extra bandwidth that is reserved for air caching. The latter is going to be exploited by applying the Normalized Gain ratio, which we defined in the previous chapter, on the ASPAC protocol. In order to explore and the other important characteristics (e.g., end-to-end delay, robustness) of the protocol we collected simulation results concerning the required Transmission Rounds metric, for various group sizes and packet error probabilities (PEPs).

Initially, we will present results that are representative of the delay performance of the protocol. The basic performance metric is the *required Transmission Rounds* metric, which represents the average number of Transmission Rounds that are required for the reliable delivery of a group of data packets (e.g., transmission group-TG). We simulated ASPAC assuming that the packet error probability (PEP) is 0.2 and the transmission

group (TG) is consisted of 20 data packets. The varying parameters in those simulated scenarios were the multicast group size, which varies from 10 to 10^5 members, and the maximum Air Cache size (*max(ACsize*)), which varies from 0 to 10 packets. We used the term "maximum Air Cache size" because the size of the Air Cache does not remain constant throughout the reliable transmission but changes depending on the feedback from the members of the multicast group, as that was described in the previous section. The only parameter that we can define for the Air Cache size is the maximum number of parity packets that can be accommodated in each transmission round. The following graph presents the set of results that we collected in order to gain an insight for the delay performance of the protocol.



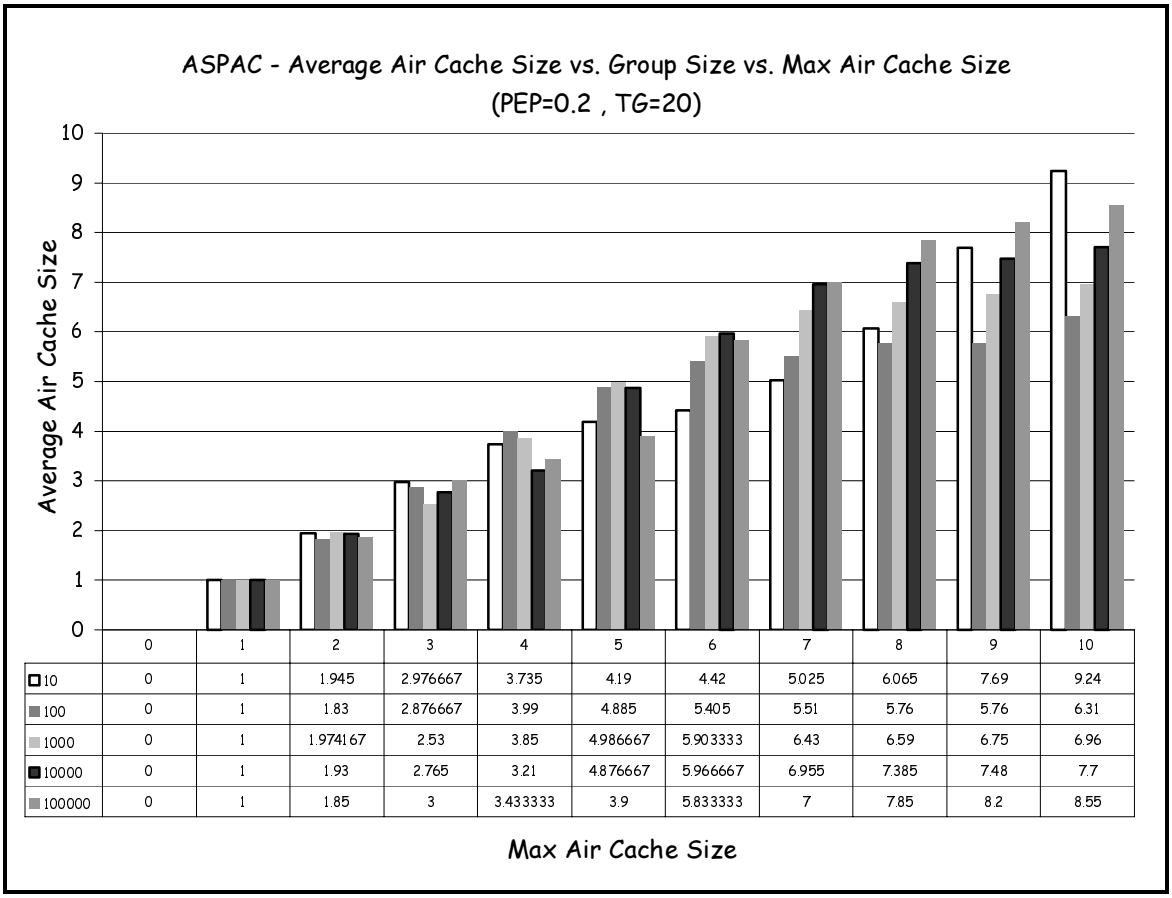
	0	1	2	3	4	5	6	7	8	9	10
10	4.17	2.72	2.21	2.03	2	1.99	1.93	1.84	1.62	1.37	1.18
100	5.52	3.29	2.82	2.19	2.01	2	2	2	2	2	1.95
1000	6.86	3.95	3.07	2.87	2.16	2.01	2.02	2	2	2	2
10000	8.24	4.52	3.3	3.02	2.86	2.1	2.02	2	2	2	2
100000	10	5.2	3.9	3	3	2.9	2.1	2	2	2	2

Figure 5.5:(ASPAC) Transmission Rounds vs. AC Size vs. Group Size (PEP=0.2, TG=20)

The results that are shown on the above figure are very promising as we expected to be, since ASPAC is very similar with the PPAC protocol, which appears also to have very promising performance characteristics. So, even though we adapt the Air Cache size and we expect that the Air Cache of ASPAC will utilize less bandwidth per transmission round compared to the corresponding scenario for PPAC, the results that we collected do not differ from the results that we collected for PPAC protocol. It seems that we do not have any degradation in performance even though the Air Cache utilizes less bandwidth, due to the size adaptation of the Air Cache per transmission round. Furthermore, and looking at the results separately from the other proposed protocols, ASPAC has very good performance and it seems very scalable. The latter can be extracted from two facts; the first is that if we define Air Caches, which can accommodate more than 25% of packets that the TG can accommodate, the number of required Transmission rounds does not change significantly, and the second and most important fact is that the rate of performance improvement as we increase the Air Cache size is not affected from the various multicast group sizes.

As we mentioned before, the performance of ASPAC was expected, so we just had to verify that, through the simulation results. The main characteristic of the protocol is the adaptation of the Air Cache size, so we would like to know what is the average gain in bandwidth per transmission round, due to the adaptation process by assuming different maximum Air Cache sizes. Those results are presented in the following graph, which represents the average Air Cache size used by ASPAC per transmission round, for various multicast group sizes and various maximum Air Cache sizes. The packet error probability (PEP) was assumed 0.2 throughout the corresponding to this graph simulated

scenarios and the transmission group (TG) was defined to be consisted of 20 data packets.

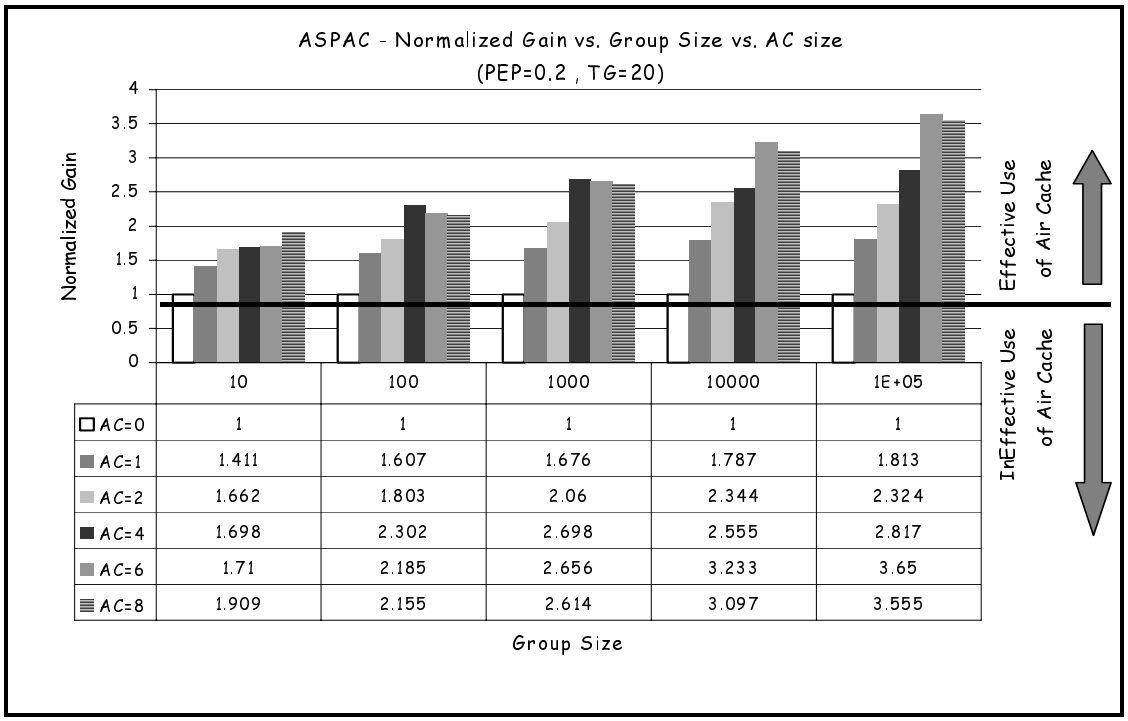


	0	1	2	3	4	5	6	7	8	9	10
10	0	1	1.945	2.976667	3.735	4.19	4.42	5.025	6.065	7.69	9.24
100	0	1	1.83	2.876667	3.99	4.885	5.405	5.51	5.76	5.76	6.31
1000	0	1	1.974167	2.53	3.85	4.986667	5.903333	6.43	6.59	6.75	6.96
10000	0	1	1.93	2.765	3.21	4.876667	5.966667	6.955	7.385	7.48	7.7
100000	0	1	1.85	3	3.433333	3.9	5.833333	7	7.85	8.2	8.55

Figure 5.6: (ASPAC) Avg. AC Size vs. Group Size vs. Max AC Size (PEP=0.2, TG=20)

Observing carefully the above graph we can draw some very important conclusions. As we can see the average Air Cache size utilized by the protocol per transmission round does not follow a standard pattern as we vary the multicast group size and the maximum Air Cache size. Instead, as we vary the maximum Air Cache size and the multicast group size, the results do not follow a standard pattern but the important observation is that ASPAC uses in average less bandwidth for air caching per transmission round compared to the non-adaptive version of the protocol (PPAC). The important point drawn from the previous two graphs is that the performance of the ASPAC protocol in terms of the

required Transmission Rounds is almost identical to the performance of PPAC, despite the fact that the average Air Cache size used is always less or equal than the Air Cache size used from the PPAC protocol per transmission round. So, in ASPAC we managed to retain the very good performance characteristics of the PPAC protocol using less extra bandwidth. Intuitively and based on the latter result, we expect that the Normalized Gain ratio related to ASPAC protocol will present better results in terms of how efficient the available bandwidth is utilized compared to the Normalized Gain ratio related to the PPAC protocol. Since, the performance of the PPAC protocol is sufficient enough to compensate for the extra bandwidth utilization, we predict that the same will apply to ASPAC since the performance is similar to PPAC and additionally ASPAC utilizes less bandwidth for air caching per transmission round. We prove that this prediction is true by presenting the values of the Normalized Gain ratio for the ASPAC protocol. Those values are shown on the following graph.



	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	1.411	1.607	1.676	1.787	1.813
AC=2	1.662	1.803	2.06	2.344	2.324
AC=4	1.698	2.302	2.698	2.555	2.817
AC=6	1.71	2.185	2.656	3.233	3.65
AC=8	1.909	2.155	2.614	3.097	3.555

Figure 5.7: (ASPAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

The Normalized Gain ratio values given above prove that our prediction is correct, since those values lay, in all cases (e.g., for various maximum Air Cache sizes and for various multicast group sizes) above the value of 1. By observing the formula from which the above results were computed,

$$NormalizedGain = \frac{TransmissionRounds_{ARQ} * TGsize}{TransmissionRounds_{ACDAC} * (TGsize + ACsize)}$$

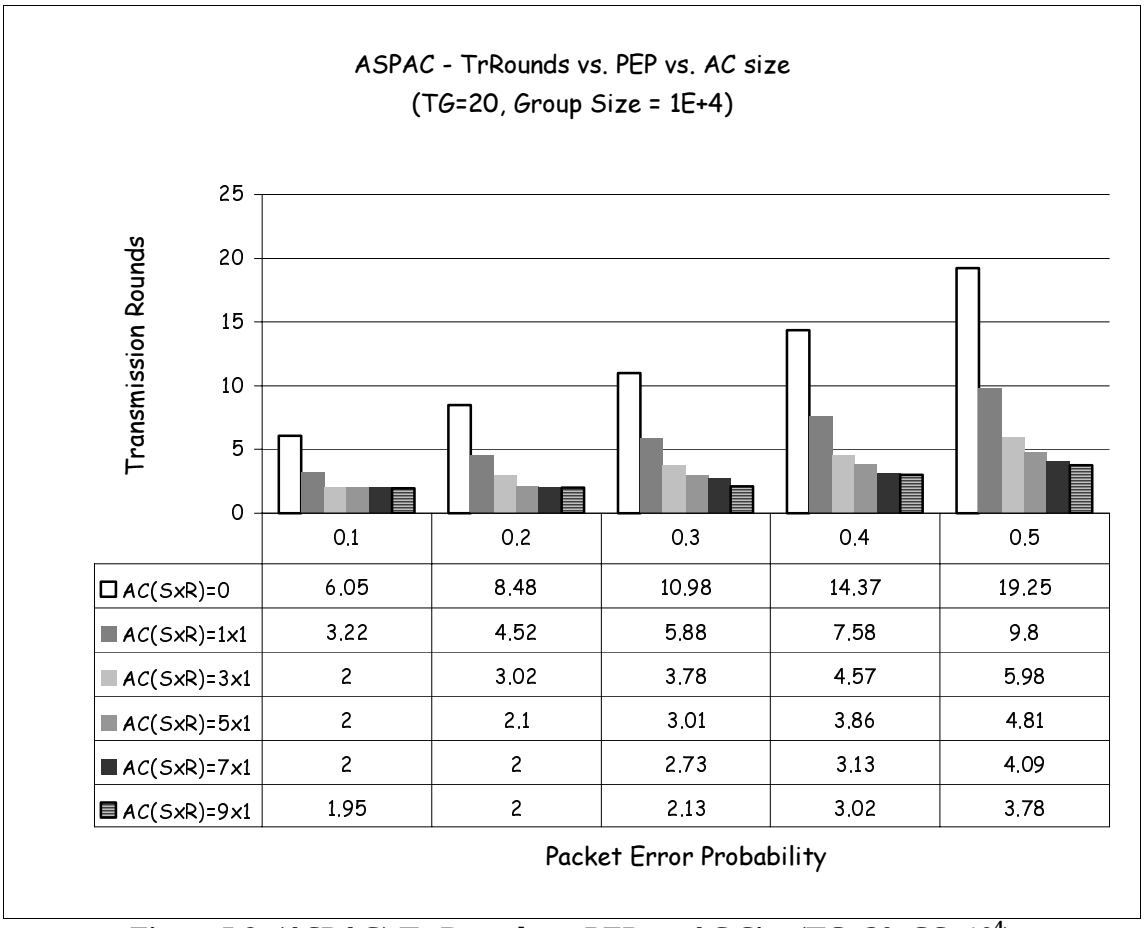
it is obvious that in order the protocol's performance to compensate for the extra bandwidth utilization, the following condition must hold:

$$\boxed{Normalized\ Gain \geq 1}$$

This condition holds for ASPAC and is obvious by observing the above graph. So, the protocol has been proven efficient in terms of the available bandwidth utilization and effective in terms of the performance, in cases where the network has flat hierarchy and we want to apply reliable multicast communications on this network architecture. The conclusion that refers to the better efficiency of the ASPAC protocol compared to the PPAC protocol is going to be revisited when we compare the designed protocols.

On our way to exploit all the aspects of the ASPAC's performance, we will evaluate the robustness of the protocol in the following paragraphs. We have proven through different simulation scenarios that the protocol presents very promising performance characteristics, but what about its performance behavior for various and especially for larger values of packet error probabilities (PEPs). We collected all the above results for the required Transmission Rounds and the Normalized Gain ratio by assuming the packet error probability (PEP) is 0.2 in each corresponding simulation scenario. In the following

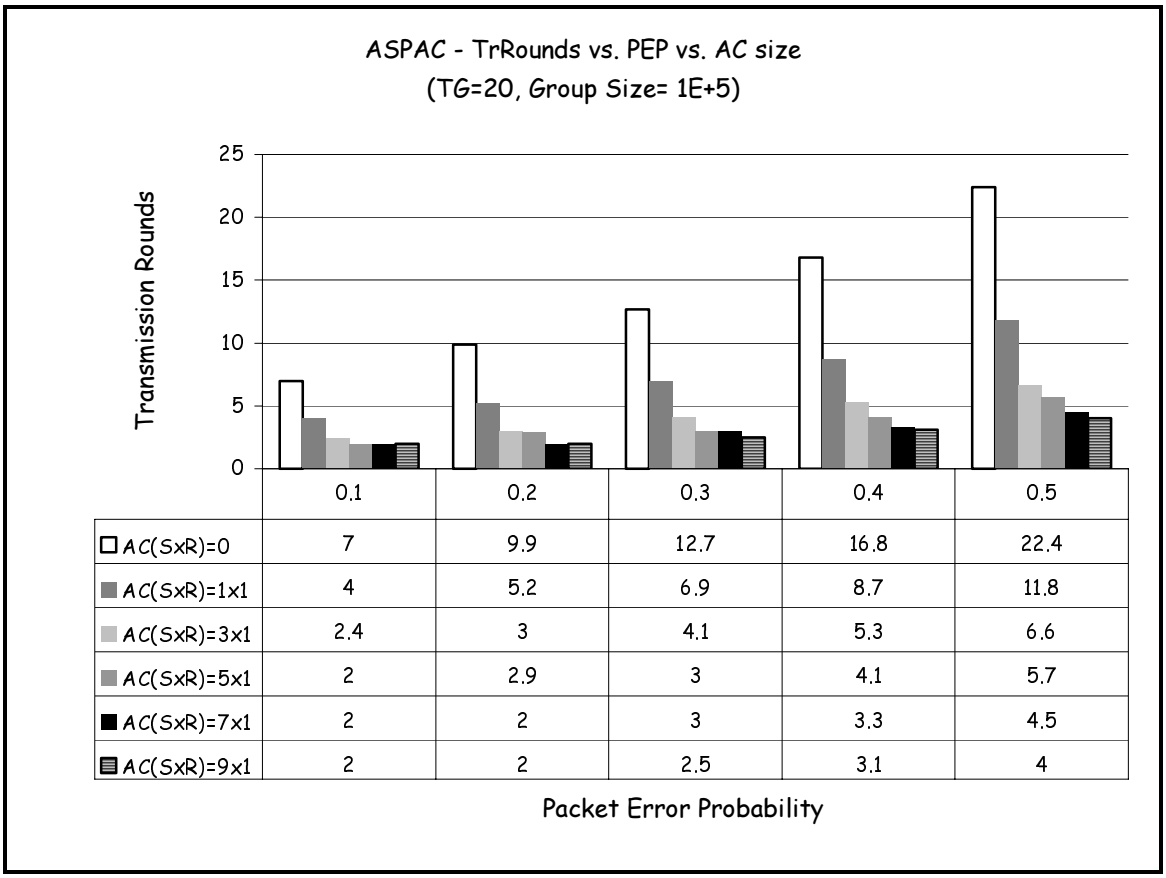
scenarios we vary the packet error probability (PEP) having as a goal to observe the performance of the protocol when the PEP is low but most importantly when it takes high values because the satellite links in a real world scenario, present high PEP. This study is important because of the fact that the satellite links are characterized from increased noise levels compared to the other wireless links. So, it is expected that in a real world scenario the PEP is going to be high. So, we evaluate the behavior of ASPAC protocol under different values of PEP, which vary from 0.1 to 0.5. The following graph presents the number of required Transmission Rounds for various PEPs and Air Cache sizes. The multicast group was assumed to be consisted of 10^4 members. The second graph presents similar results but when the multicast group has 10^5 members. In both cases the protocol is proven to be extremely robust even for high packet error probabilities (PEPs). The effectiveness and robustness of the protocol are obvious when we compare the first column of each group of results with the rest of the columns of the same group, since the first column corresponds to the non Air Cache scenario. So, beyond the robustness of the protocol, this graph presents also the effectiveness of the protocol, which was an expected result, since the Air Cache is filled with parity packets for applying forward error correction (FEC) to each one of the members of the multicast group. One more fact that we can export from the following graphs by comparing the non Air Cache scenario with the rest of the scenarios, is that when there is not air caching applied, the performance seems to degrade exponentially compared to the cases where air caching is applied. In the latter scenarios the performance degrades linearly or even more slowly. The latter statement is obvious from the following graphs when you draw the lines that connect the top of the columns in each different case.



	0.1	0.2	0.3	0.4	0.5
AC(SxR)=0	6.05	8.48	10.98	14.37	19.25
AC(SxR)=1x1	3.22	4.52	5.88	7.58	9.8
AC(SxR)=3x1	2	3.02	3.78	4.57	5.98
AC(SxR)=5x1	2	2.1	3.01	3.86	4.81
AC(SxR)=7x1	2	2	2.73	3.13	4.09
AC(SxR)=9x1	1.95	2	2.13	3.02	3.78

Figure 5.8: (ASPAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^4)

Similar robustness characteristics have been already identified at the PPAC protocol, and as it was expected, ASPAC's robustness characteristics do not surprise us, since this protocol is based on the same techniques with the PPAC protocol. The only difference appears only on how ASPAC manages the extra bandwidth for more efficient resource usage.



	0.1	0.2	0.3	0.4	0.5
AC(SxR)=0	7	9.9	12.7	16.8	22.4
AC(SxR)=1x1	4	5.2	6.9	8.7	11.8
AC(SxR)=3x1	2.4	3	4.1	5.3	6.6
AC(SxR)=5x1	2	2.9	3	4.1	5.7
AC(SxR)=7x1	2	2	3	3.3	4.5
AC(SxR)=9x1	2	2	2.5	3.1	4

Figure 5.9: (ASPAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

Quantitatively, the results above show that in cases where the number of packets that the Air Cache can accommodate exceeds the 30% of the number of data packets sent per transmission round; the required Transmission Rounds do not exceed the number 6 even in the cases where the PEP is 0.6 (e.g. more that the half packets are corrupted). Focusing on the last column of the above graph we can observe that parity Air Cache gives a significant boost to the performance of the reliable multicast protocols in flat hierarchy networks, since for the non Air Cache protocols in the case of a multicast group of 10^5 members and PEP=0.5 the average number of Required Transmission rounds is 22.5 compared to the 5.7 required Transmission Rounds in average, in the case where we apply the ASPAC protocol with Air Cache size equal to 25% of the regular transmission

channel size (TG=20 packets/Air Cache size=5 packets). The robustness of the ASPAC protocol is unquestionable and reaches the levels of the PPAC's robustness characteristics even though ASPAC uses less extra bandwidth per transmission round in average for air caching compared to PPAC. So, ASPAC has been proven efficient, robust with the extra advantage that uses less bandwidth in average dedicated to air caching per transmission round. The following paragraph wraps up the performance characteristics of the ASPAC protocol as they have been exported from the simulation results presented throughout this paragraph.

5.4.3 Performance Analysis

In this paragraph we sum up our observations about the performance characteristics of the ASPAC protocol, we refer also to its advantages and disadvantages as these have been pointed out from the previously presented simulation results. Although, we have made comments about the behavior of the protocol, it is considered helpful to give an overview of the protocol's pros and cons based on metrics like delay performance, robustness, efficient use of bandwidth, power and processing requirements and complexity of the protocol.

The comments on this paragraph are not going to differ much from the corresponding comments of the PPAC protocol. That is because the ASPAC protocol is based on the same operation principals with the PPAC protocol (e.g., parity Air Cache). The only difference is the way those two protocols are handling the bandwidth, which is reserved for air caching. The PPAC protocol is using a constant bandwidth for the Air Cache per transmission round, in contrast to ASPAC protocol, which is using a variable size Air

Cache based on the number of requests for retransmission per transmission round. As we have already seen, the ASPAC protocol utilizes in average less bandwidth for air caching per transmission round, without affecting the delay performance of the protocol, as we observed on the results presented in the previous paragraphs.

Wrapping up, ASPAC is a very efficient and scalable protocol with very promising delay characteristics, since we can transmit reliably a group of packets in very large multicast groups (i.e., hundreds of thousands of members) in a small number of transmission rounds, compared to the corresponding number of rounds, which are required from non Air Cache protocols that are solely based on ARQ or FEC. Apart from the delay performance characteristics of the ASPAC protocol, we can also refer to the robustness of the protocol. Based on the above simulation results ASPAC seems very robust even in cases where the PEP is as high as 0.5. The decrease in delay performance as the PEP increases is very slow (e.g., less than linear decrease). The latter fact is one advantage of the protocol and can be proven crucial in the real world application of the protocol, since the satellite links, which are this research target, have high PEP (packet error probability). Observing the robustness characteristics of the ASPAC protocol it is more than obvious that the delay performance of the protocol hardly changes even though the PEP increases in a constant rate. The latter two characteristics of the ASPAC protocol, which have to do with the delay performance and the robustness, are indicative for the success of this protocol.

So, what are the new advantages of the ASPAC protocol that PPAC lacks? This question is trivial if we consider the different way the extra bandwidth, which is dedicated for air caching, is treated from the ASPAC protocol. In PPAC the Air Cache is

filled with a constant number of parity packets in each transmission round (e.g., non adaptive Air Cache size). In ASPAC the Air Cache size does not remain the same per transmission round, but instead the Air Cache is filled with the necessary number of parity packets, up to a maximum number (*max(ACsize)*). Acting in such a way, we lower the average bandwidth utilization per transmission round since there are transmission rounds where we do not have to fill the Air Cache with the maximum allowable number of packets. Practically then, the ASPAC protocol uses in average less bandwidth for air caching per transmission round than the corresponding PPAC protocol but without making any sacrifice in delay performance. We proved the latter statement by using the Normalized Gain ratio of ASPAC protocol and observing that behaves better than the corresponding ratio of PPAC. This result is used as an indication that the ASPAC protocol is more efficient and as effective as the PPAC protocol.

Of course, nothing is coming for free, since the improvement in average Air Cache bandwidth consumption per transmission round of ASPAC compared to the PPAC protocol is because the ASPAC protocol collects and processes requests for packet retransmissions from the multicast group members per transmission round, in contrast to the operation of the PPAC protocol, which does not require feedback to function correctly and efficiently. This fact is very important since there would be an extra delay introduced between the retransmission rounds because of the collection and processing of feedback. So, one of the cons of the protocol is that it requires feedback from the receivers, which is a very costly procedure for both the receivers and the sender (e.g. especially in satellite links because of the increased propagation delay and in some cases because the satellite link is not duplex (extra hardware is needed for duplex satellite

communication)). If this process of collection and processing of feedback can be done in a more efficient way then this protocol is preferable than PPAC, otherwise PPAC is the protocol of choice.

Finally, one more of the cons of the protocol that also exists in the PPAC protocol is the fact that the protocols like the PPAC and ASPAC that use parity packets in the Air Cache, base the recovery of the erroneous or corrupted packets on the forward error correction (FEC) technique. Although, this technique is much more effective than the ARQ technique, it requires more powerful receiver devices since the recovery of the corrupted packets is based on the decoding of parity packets in combination with the correct received data packets. The decoding process requires more processing power and consecutively more energy for a successful data packet recovery compared to the techniques that are based on ARQ, where no decoding is needed since the recovery of the data packets is done by the retransmission of the actual data packets. The latter conclusion is important when we refer to networks that are consisted from devices that are not so powerful in processing power and they are operate with limited life time batteries. So, the life of those devices is expected to decrease faster when we use protocols that are based on forward error correction (FEC) (e.g., because of the demanding encoding/decoding procedures). Furthermore, one more issue is the buffering requirements, since the FEC technique as it is used in ASPAC and PPAC need enough buffering space for both the correctly received data packets and the correctly received parity packets until the correct reception or recovery of the complete group of data packets. On the other hand, even though the latter two facts about the requirements of the ASPAC protocol seem to eliminate its use, when we have to deal with networks of

devices that do not have the processing power, the battery life and the memory (e.g., for buffering), we have to take into consideration the technological achievements of the recent years where we can have small devices with huge processing capabilities, we operate on the lowest necessary power and having large amounts of buffering space. The latter observation is a fact that proves the usefulness of the protocols like ASPAC, because since we overcome the issues that limit the utilization of those protocols we can apply them to improve the reliable multicast communications between the sender and a large number of receivers.

In the next section we present one more reliable multicasting protocol, which is not based on FEC but instead is based on ARQ combined with the Air Cache technique. We will use the same techniques that we used for the previous protocols, for the evaluation of the protocol in terms of its delay performance, efficiency and robustness. The protocol is called Hybrid Adaptive Data Air Cache (HADAC) protocol and is presented in the following paragraphs along with its performance evaluation.

5.5 Hybrid Adaptive Air Cache (HADAC)

The evaluation of the reliable multicasting protocols that belong the class of the adaptive Air Cache reliable multicasting protocols is completed with the HADAC protocol, which is described in this section. The protocol, which is named after its basic characteristics (e.g., following the onomatology that we established from the previous chapter), is called Hybrid Adaptive Data Air Cache Protocol (HADAC). So, the three most important characteristics of HADAC are:

- The Air Cache is filled with data packets

- The size of the Air Cache is dynamic, based on the feedback from the multicast group
- The content of the Air Cache is adapted based on the feedback from the multicast group

How the adaptation is done, we will find out when we describe the protocol in more detail.

Our choice to adapt both the size and content of the protocol was originated from the results that we collected for the Adaptive Content Data Air Cache (ACDAC) protocol. When we evaluated the ACDAC protocol we concluded that the protocol behaves much better than the UDPAC and RDPAC protocols, which do not adapt the Air Cache content based on feedback but instead they fill the Air Cache per transmission round by randomly selecting data packets from the TG. Also, all of these protocols, included the ACDAC, use constant size Air Cache. The UDPAC and RDPAC protocols are not relying on dynamic adaptation of the Air Cache size because there is not feedback information from the multicast group, information that is available in the case ACDAC protocol but in this case is used only for the content adaptation. So, in order to get the performance of ACDAC protocol along with the minimum required Air Cache bandwidth consumption per transmission round, we have to adapt the Air Cache size along with the content adaptation of the Air Cache. Based on our intuition we expect that HADAC protocol will have the performance characteristics of ACDAC protocol but with less in average extra bandwidth consumption per transmission round for air caching.

In the following paragraphs we present the HDPAC algorithm and the performance evaluation of the protocol as we did for the previously presented protocols. The

performance evaluation is based on metrics like the required Transmission Rounds for the reliable reception of all the data packets from each one of the members of the multicast group, the robustness of the protocol in environments that present low and high packet error probability (PEP). Also, one more very crucial characteristic of the protocol that we have to elaborate on is the scalability of the HADAC. All these questions are going to be investigated and answered in the best possible way through the simulation results that we collected and present in this section. At the end of the section we wrap up the pros and cons of the protocol based on the performance results and the hardware requirements of HADAC. The latter are specified based on the memory and processing power requirements of the error recovery techniques that the HADAC is based on (e.g., air caching, ARQ).

5.5.1 HADAC's algorithm

As we mentioned above the main characteristics of the protocol is the concurrent adaptation of the size and content of the data Air Cache. The data packets, which constitute the Air Cache, are selected in the same fashion as in ACDAC protocol, based on the feedback collected from the multicast group members, in each transmission round. Those data packets are packets that are transmitted in C_1 where the regular transmission of the TG is taking place. The goal is the recovery of the data packets that have been corrupted or corrupted the most[4] in the previous transmission round. The latter characteristic is also part of ACDAC protocol, but the difference of ACDAC compared to

[4] In the case where the number corrupted data packets exceeds the maximum number of packets that the Air Cache can accommodate in each transmission round, the most corrupted packets have highest priority on the selection process

the protocol described here (HADAC) is that we may release the bandwidth, which we reserved for air caching, when this is redundant. We operate in such a fashion in order to lower the average consumption of bandwidth for air caching (channel C_2) per transmission round, compared to ACDAC protocol, which uses a constant Air Cache size in each transmission round. How the HADAC protocol operates is described from the following algorithmic steps. Also, we describe how the Air Cache is refreshed and how the size and content are adapted in each transmission round based on the feedback from the multicast group members.

HADAC (Hybrid Adaptive Data Air Cache)

Phase I (Communication Protocol)

Step I: The initial round is similar to UDPAC, RDPAC and ACDAC. We have two concurrent transmissions going on channels C_1 and C_2. In C_1 happens the regular transmission of the TG data packets and in C_2 we have the transmission of the data Air Cache packets. The Air Cache is filled with *max(ACsize)* data packets randomly chosen from the transmission group (TG).

Step K (K>1): If there are not requests for retransmission then the transmission of the TG data packets ends, because all the members of the multicast group have received the TG reliably. In this case the number of the required Transmission Rounds for the reliable delivery of TG data packets is *K-1*.

Otherwise, the two channels C_1 and C_2 are utilized again. The former (e.g. C_1) is used again for the regular transmission of the TG data packets and the latter (e.g. C_2) is used for the transmission of the updated Air Cache contents. The Air Cache is updated based on the collection of requests for retransmission that

are referred on the transmission of round *K-1*. How the Air Cache is refreshed based on the feedback from the members of the multicast group is described from the following algorithm. We update *K* by setting it equal to *K+1* and we repeat this step.

The algorithm for the update of the Air Cache in each transmission round is based on the feedback from the previous transmission round and is described below. In this category of protocols (e.g. Hybrid Adaptation of the Air Cache Protocols) the Air Cache is adapted both in content and size. The description of how the adaptation process is realized, follows:

Content and Size Adaptation of the Air Cache

Phase II (Update of the Air Cache)

*Step I***:** On this round the Air Cache is filled in the same fashion as in UDPAC, RDPAC and ACDAC. The Air Cache is filled with the maximum allowable number of data packets (*max(ACsize)*) that can be transmitted in a single Transmission Group Transmission Round (TGTR). The contents of the Air Cache are randomly selected from the data packets of the transmission group (TG).

Step K (K>1): Since in this round we can collect feedback from the members of the multicast group we can adapt properly the content and the size of the Air Cache. The size is adapted in the same fashion as in ASPAC. If we assume that the number of packets that have been requested for retransmission is $DRQpacks_{K-1}$ and the maximum allowable number of packets in the Air Cache is *max(ACsize)*,

then the size of the Air Cache in round *K*, which is symbolized as $ACsize_K$,will be:

$$ACsize_K = \min(DRQpacks_{K-1}, \max(ACsize))$$

The Air Cache size is determined as follows:

- Case 1: (if $ACsize_K = \min(DRQpacks_{K-1}, \max(ACsize)) = \max(ACsize)$)

  The Air Cache will have size and will contain the $\max(ACsize)$ most requested data packets from the multicast group members.

- Case 2: (if $ACsize_K = \min(DRQpacks_{K-1}, \max(ACsize)) = DRQpacks_{K-1}$)

  The Air Cache will have size $DRQpacks_{K-1}$ and will contain all the requested data packets. The number of the distinct data packets of the TG that have been requested for retransmission in the previous round is $DRQpacks_{K-1}$.

After having described the algorithmic details of the protocol that describe its two-phase operation (e.g., communication phase and update of the Air Cache phase), we will present some very important performance results that we collected after simulating the HADAC protocol. Those results correspond to the required Transmission Rounds metric, the robustness and the scalability of HADAC.

5.5.2 Simulation Results

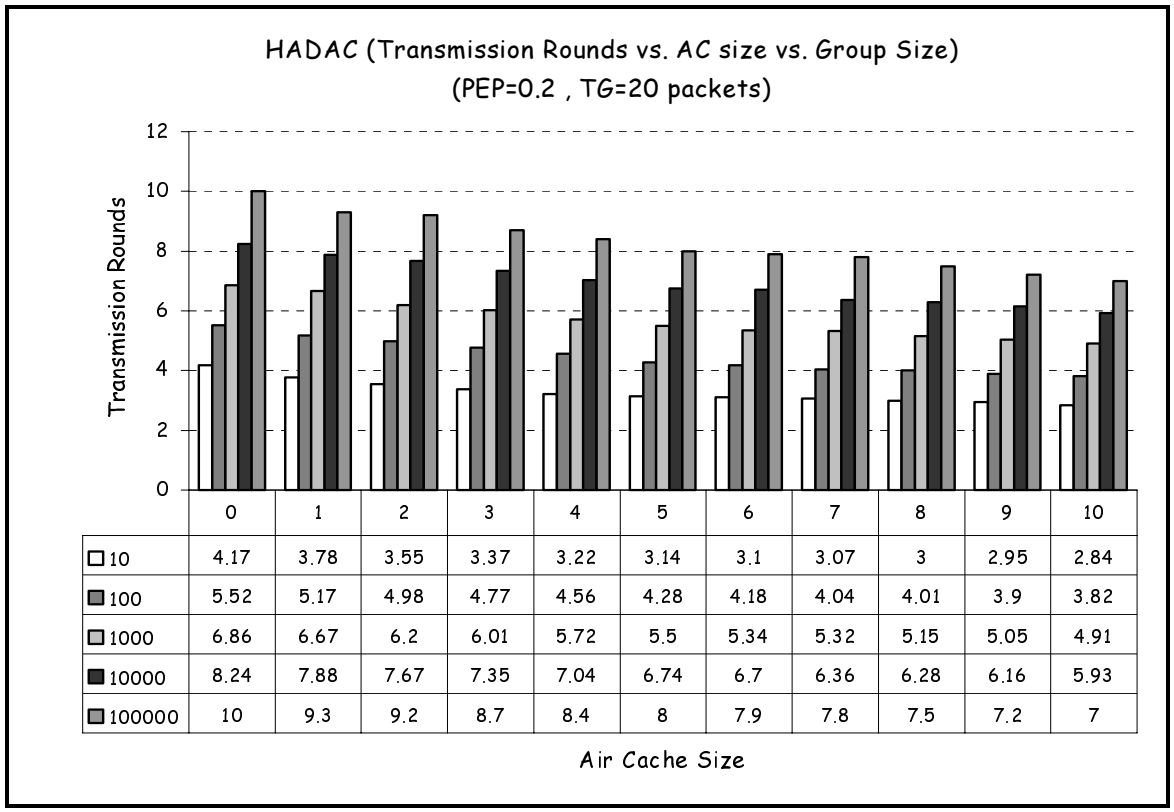
Following the same procedure as we did with the rest of the proposed protocols, we will evaluate HADAC protocol based on the values of different metrics that we collected by simulating the protocol. Those metrics have to do with:

- The efficiency of the protocol in terms of the required Transmission Rounds for the complete and reliable transmission of the TG.
- The robustness that the protocol presents when the PEP increases
- The scalability characteristics that the protocol presents when it operates for large multicast groups of sizes in the range of $O(10^5)$ members.

Intuitively, we expect that the performance characteristics of the HADAC protocol will not differ much from the performance of the ACDAC, which has already been presented earlier in this chapter (paragraph 5.2). The latter intuitive comment arises from the algorithmic similarities that those two protocols present. The only difference is that the HADAC protocol is expected to utilize in average less extra bandwidth for air caching per transmission round compared to ACDAC, which uses a constant size Air Cache in each transmission round. So, the HADAC is expected to have similar performance characteristics with the ACDAC protocol, with the extra advantage that uses less extra bandwidth per transmission round. The latter two intuitive statements about the performance of HADAC protocol, point out that we will get better results for the Normalized Gain ratio than the corresponding results we got for the ACDAC protocol. We will find out if our intuition is correct in the following paragraphs, where we present the results we collected for the various performance metrics.

Initially we will present and evaluate the results corresponding to the required Transmission Rounds for the complete and reliable transmission of TG data packets. We assumed the packet error probabilities (PEPs) are independent and constant throughout the reliable transmission and equal to 0.2 for each data packet either this is transmitted in C_1 (e.g., transmission of the TG data packets) or C_2 (e.g. transmission of the data Air

Cache). The TG is consisted of 20 data packets. We simulated the behavior of the HADAC protocol for various maximum Air Cache sizes (*max(ACsize)*) (e.g., 0 to 10 data packets) and various multicast group sizes (e.g.,10 to 10^5 multicast group members). By collecting the results of the various simulation scenarios, we can evaluate the performance of the protocol in terms of the required Transmission Rounds for the reliable delivery of TG data packets to each one of the members of the multicast group. By varying the parameters, which correspond to the maximum available Air Cache size and the multicast group size, we will also collect valuable information about the scalability characteristics of the proposed protocol for various Air Cache size. Comments referred to the required Transmission Rounds metric and scalability characteristics of HADAC are given after the following graph, which presents the result that we collected, for the parameter values mentioned above.



	0	1	2	3	4	5	6	7	8	9	10
10	4.17	3.78	3.55	3.37	3.22	3.14	3.1	3.07	3	2.95	2.84
100	5.52	5.17	4.98	4.77	4.56	4.28	4.18	4.04	4.01	3.9	3.82
1000	6.86	6.67	6.2	6.01	5.72	5.5	5.34	5.32	5.15	5.05	4.91
10000	8.24	7.88	7.67	7.35	7.04	6.74	6.7	6.36	6.28	6.16	5.93
100000	10	9.3	9.2	8.7	8.4	8	7.9	7.8	7.5	7.2	7

Figure 5.10: (HADAC) Tx Rounds vs. AC Size vs. Group Size (PEP=0.2, TG=20)

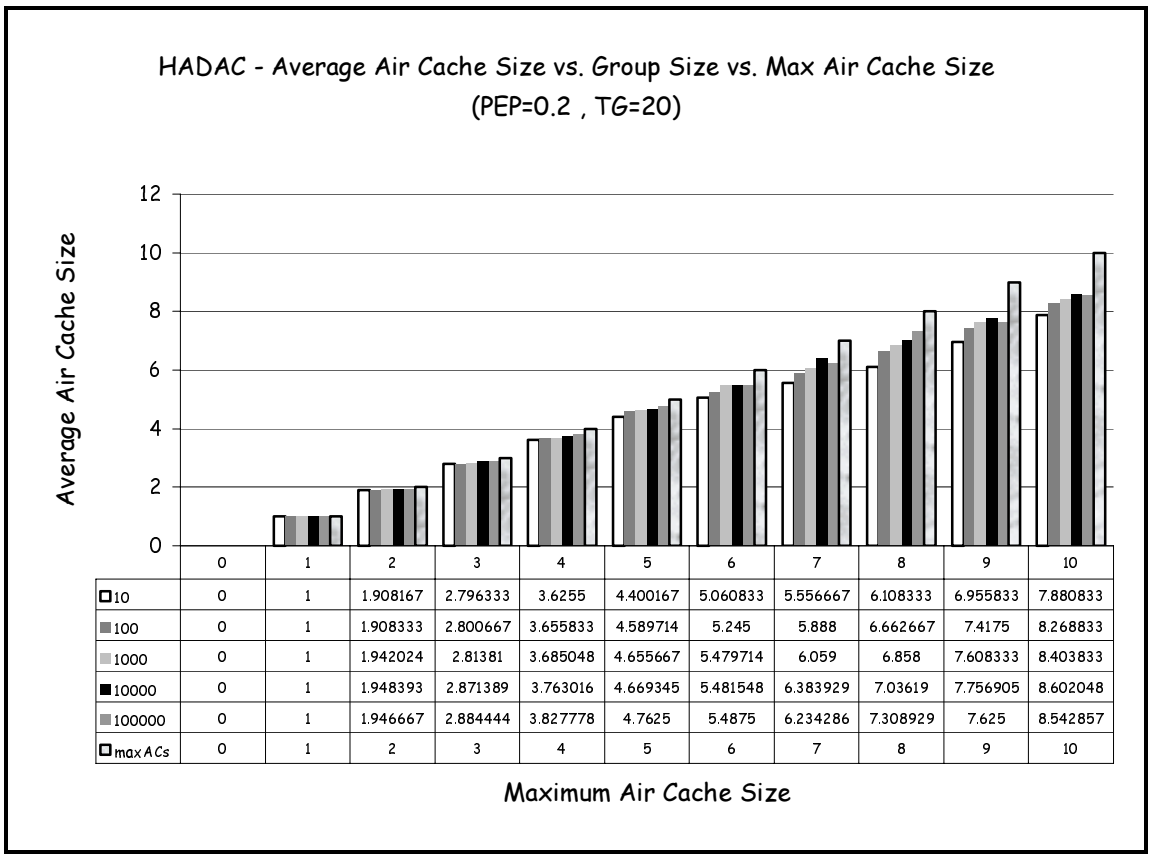
Analyzing, the above results, we can conclude that the protocol presents some very nice characteristics compared to the protocols that belong in the class of those that are ARQ-based and not FEC-based. Focusing on the arithmetic results corresponding to the required Transmission Rounds for various Air Cache sizes and various multicast group sizes, we observe that they do not refer much from the results we collected, by simulating the ACDAC protocol under the same assumptions and the same parameter values. The latter observation is important because the Adaptive Content Data Air Cache (ACDAC) protocol performs better than a non Air Cache ARQ-based protocol, which utilizes only C_1 for the regular transmission of the data packets and the retransmission of the TG for the recovery of the corrupted or erroneous packets at the receivers. On the other hand, the Air Cache size that ACDAC reserves per transmission round is constant and this might result in inefficient utilization of the bandwidth dedicated to C_2 per transmission round (e.g. the number of requested data packets is less than the Air Cache size, so a part of the bandwidth dedicated to Air Cache will not utilized). The fact that HADAC protocol utilizes the appropriate amount of bandwidth for air caching per transmission round and has similar performance characteristics with ACDAC, demonstrates the improvement of the protocol in terms of the combination of Air Cache size utilization and number of required Transmission Rounds. Also, by further commenting on the above graph, we can come into some important conclusions about the scalability of the protocol. As, we increase the size of the multicast group the number of the required Transmission Rounds increases accordingly. More detailed comments related to the above graph will be given in the next paragraph, where we analyze the performance of the protocol based on the results that will be presented throughout this paragraph.

Moving ahead to the presentation of the results for the different performance metrics, we present here the results that we collected for the average consumption of the extra bandwidth dedicated to Air Cache per transmission round. Throughout the simulations we assumed that the packet error probability (PEP) is constant and equal to 0.2 (PEP) for each packet, either this is transmitted to C_1 or to C_2. The packet error probabilities (PEPs) follow the uniform distribution and they are independent of each other. In order to collect the following results we varied the maximum allowable Air Cache size per transmission round and also we varied the multicast group size from 10 to 10^5 members in order to study if we can get any savings on the average Air Cache size utilization even in cases where the multicast group size is very large. We want to find out what is the average size of the Air Cache per transmission round *(avgACs)*, compared to the case where the maximum allowable size of the Air Cache per transmission round *(maxACs)* is constant throughout the reliable transmission. We expect that the relation between those two variables is going to be as follows:

$$avgACs < \max ACs$$

In the case, where t those two variables are equal, HADAC does not differ from ACDAC, since there are no savings in extra bandwidth utilization and the required Transmission Rounds performance of HADAC does not differs from the corresponding performance of ACDAC protocol. So, we are expecting that the above inequality will always hold for the Air Cache bandwidth utilization in HADAC, in order this protocol to be an improved version of ACDAC. We will find out if the latter is true by analyzing the results that we collected for the average Air Cache size utilization, which are presented in the following figure. Those results along with the corresponding results for the required Transmission

Rounds metric indicate the more efficient utilization of the Air Cache bandwidth in HADAC compared to ACDAC



	0	1	2	3	4	5	6	7	8	9	10
10	0	1	1.908167	2.796333	3.6255	4.400167	5.060833	5.556667	6.108333	6.955833	7.880833
100	0	1	1.908333	2.800667	3.655833	4.589714	5.245	5.888	6.662667	7.4175	8.268833
1000	0	1	1.942024	2.81381	3.685048	4.655667	5.479714	6.059	6.858	7.608333	8.403833
10000	0	1	1.948393	2.871389	3.763016	4.669345	5.481548	6.383929	7.03619	7.756905	8.602048
100000	0	1	1.946667	2.884444	3.827778	4.7625	5.4875	6.234286	7.308929	7.625	8.542857
maxACs	0	1	2	3	4	5	6	7	8	9	10

Figure 5.11: (HADAC) Avg. AC Size vs. Group Size vs. Max AC Size (PEP=0.2, TG=20)

The last column in each group of columns specifies the maximum allowable Air Cache size (*maxACs*) for the corresponding group of columns. Those groups of columns represent the average Air Cache size utilization for a specific maximum Air Cache size. By comparing that last column with the rest of the columns (e.g., each column represents different multicast group size) in each group of results, we get an insight about the average savings in the Air Cache bandwidth utilization per transmission round. Some general observations about the above results related to the *avgACs* are:

1. The larger the maximum allowable Air Cache size, the larger the savings on the Air Cache bandwidth utilization.

2. The larger the multicast group, the lower the average savings on the Air Cache bandwidth utilization.
3. For very small *maxACs* there are almost no savings on Air Cache bandwidth utilization. For example, if we have small *maxACs* (e.g., ACs=1 or 2) then $avgACs \cong \max ACs$.

The above three general statements describe the amount of savings that we get on Air Cache bandwidth utilization when we adapt the size of the Air Cache based on the feedback from the multicast group per transmission round. The latter results indicate that the nice properties of the protocol in terms of its combined performance (e.g.,*RequiredTRounds* x *ExtraBW*) are obvious when the Air Cache size is larger compared to the TG size. Observing more closely the above group of results, we conclude that we can get significant bandwidth savings by adapting the Air Cache size when:

$$ACsize \geq 0.6 \times TGsize$$

The latter observation indicates that for $ACsize < 0.6 \times TGsize$ the protocol's performance characteristics are similar to the ACDAC protocol's performance characteristics, and the adaptation in size does not result in any significant decrease on the average Air Cache bandwidth utilization, but instead adds to the protocol complexity due to the adaptation of the Air Cache size per transmission round. More comments are given in the following paragraph when we analyze the performance results that we collected for HADAC.

After having presented the results, which correspond to the average Air Cache size per transmission round due to the adaptation of the Air Cache size per transmission

round, we have to evaluate the protocol in terms of its efficiency. In other words we would like to find out if the performance gain that we get is sufficient enough to compensate for the bandwidth this protocol utilizes compared to the performance of a non Air Cache ARQ-based protocol, which operates only on C_1, and the Air Cache size is always 0 packets. We are going to evaluate HADAC, following the same procedure as the one we have followed throughout this thesis for the rest of the proposed protocol, by using the Normalized Gain ratio:

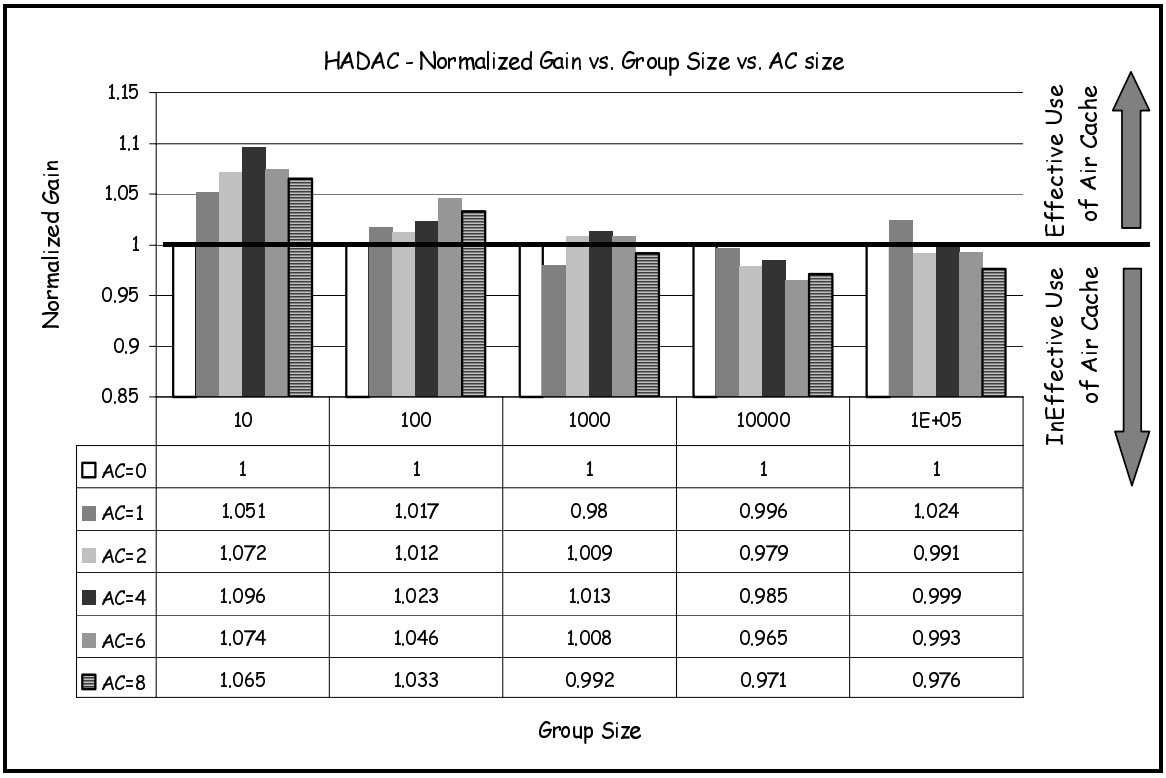
$$NormalizedGain = \frac{TransmissionRounds_{ARQ} * TGsize}{TransmissionRounds_{ACDAC} * (TGsize + ACsize)}$$

and checking if the following condition:

$$\boxed{Normalized\ Gain \geq 1}$$

holds. As we explained earlier, the above condition shows that the performance gain related to the required Transmission Rounds metric is sufficient enough to compensate for the extra bandwidth utilization due to air caching. The latter study is based on the Normalized Gain ratio because the product (*BW(in packs)* x *requiredTRounds*) on the numerator represents the normalized performance of a non Air Cache ARQ-based protocol and the denominator represents the normalized performance of the HADAC protocol. We know (e.g., see figure 5.11) that the HADAC protocol requires less transmission rounds but more bandwidth, so the ratio of the normalized performances will provide us with the answer to the question of the efficiency of the protocol. The following graph presents the values of the Normalized Gain ratio for various maximum allowable Air Cache sizes and various multicast group sizes. Equivalent to the parameter

values assumed at the previous experiments, the packet error probability (PEP) is 0.2 for each transmitted packet in C_1 or C_2.



	10	100	1000	10000	1E+05
AC=0	1	1	1	1	1
AC=1	1.051	1.017	0.98	0.996	1.024
AC=2	1.072	1.012	1.009	0.979	0.991
AC=4	1.096	1.023	1.013	0.985	0.999
AC=6	1.074	1.046	1.008	0.965	0.993
AC=8	1.065	1.033	0.992	0.971	0.976

Figure 5.12: (HADAC) Normalized Gain vs. Group Size vs. AC Size (PEP=0.2, TG=20)

On the above graph the black thick line where the Normalized Gain ratio equals to 1 marks the boundary, which separates the area where the protocol is efficient (e.g., the performance compensates for the extra bandwidth usage) and the area where the performance of the protocol fails to compensate for the extra bandwidth usage. The efficient area is the one above the black thick line ($Normalized\ \ Gain > 1$) and is where we expect the values of the Normalized Gain ratio for the HADAC protocol to lay. Observing the above results, it is obvious that the HADAC protocol has efficiency problems. As the multicast group size increases from 10 to 10^5 members, the Normalized Gain ratio decreases and drops below the critical value of 1. The values of the ratio show the tendency to drop below 1, when the multicast group size becomes greater or equal to

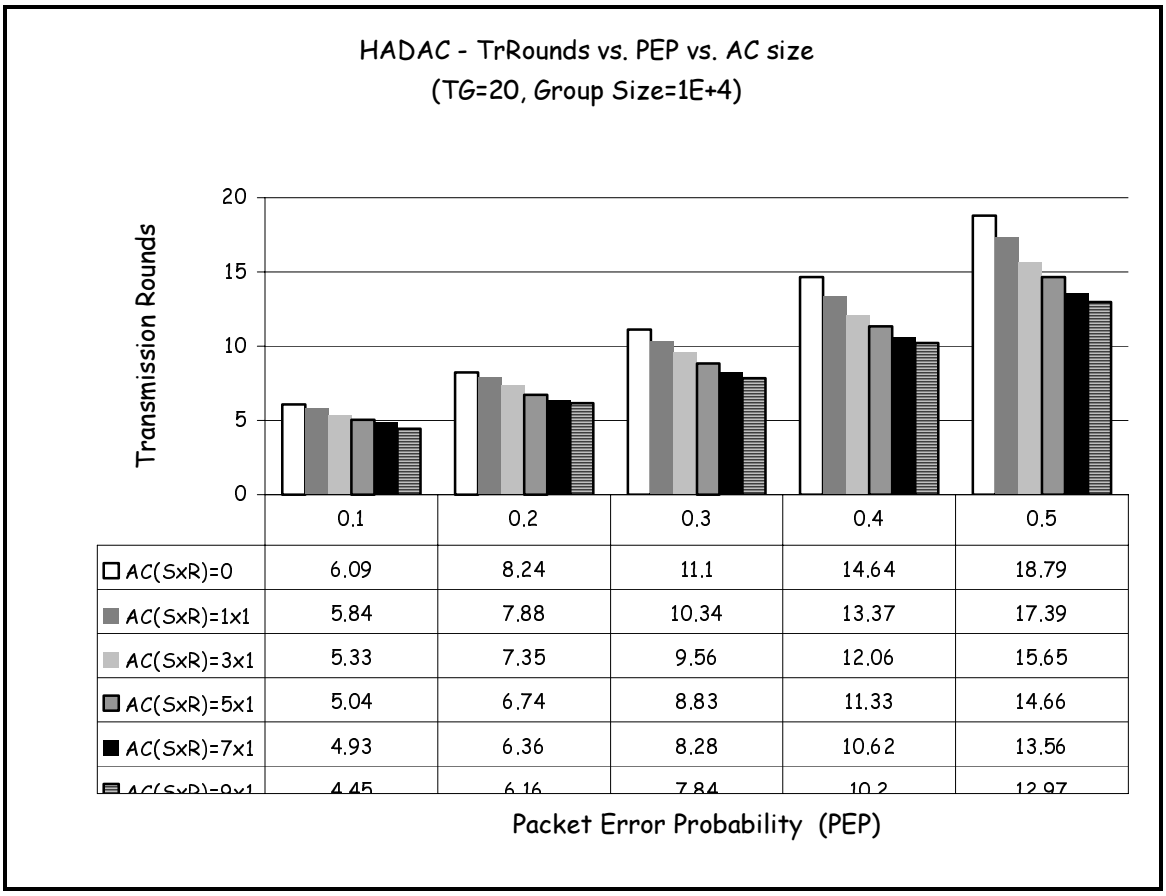
10^3 members, which is the size of the multicast group where the Normalized Gain ratio is almost at the boundary of the efficient area. Even though those results prove that there are cases where the protocol does not exhibit efficiency, especially for large multicast groups, on the other hand analyzing the results beyond the strictly mathematical framework, and looking at them from another point of view, we could ask the question "How inefficient is our protocol?". The answer to the latter question is critical in convincing us that the HADAC protocol can be used in practice and can be taken into consideration in real world scenarios. Again, interpreting the results presented on the above graph we conclude that the inefficiency[5] that this protocol presents in some cases is very small and does not exceed the 3.5% (e.g., The minimum value that the Normalized Gain ratio takes is 0.965). In most of the scenarios presented above, the inefficiency is of the order of 1% and less. Now, that we have an idea of the HADAC's inefficiency, we can say that the protocol can be taken into consideration in a real world scenario, since the number of required Transmission Rounds for the reliable transmission of the TG data packets, is much less that the corresponding number in the case where we have a non Air Cache ARQ-based protocol. So, if we want to complete the reliable multicast transmission of the TG data packets in less required Transmission Rounds without worrying for the negligible inefficiency of the protocol (e.g., due to the bandwidth utilized for air caching) then HADAC is the protocol of choice in the class of the ARQ-based protocols. So, as a final remark we highlight the fact that the HADAC protocol might be inefficient in some cases but this inefficiency is not important when we have

[5] We measure the inefficiency as the percentile of how far is the value of the Normalized Gain factor from the value of 1. So, the calculation is done like this: $\left[\left(1-NormalizedGain\right)*100\%\right]$.

enough available bandwidth to consume, because the number of required Transmission Rounds drops significantly compared to a non Air Cache ARQ-based protocol.

Up to this point we presented the performance results that have to do with the required Transmission Rounds metric, which are related to the delay performance of the protocol. Those results gave us a first insight about the scalability of the protocol. Furthermore, we presented the results about the average Air Cache bandwidth utilization, which indicate that the adaptation of the Air Cache size is effective in reducing the average bandwidth utilized by the Air Cache per transmission round. Finally, we demonstrated how effectively the HADAC protocol uses the extra bandwidth to decrease the required Transmission Rounds for the reliable transmission of TG data packets. One aspect of the protocol that we have not taken into consideration yet and is crucial in the case of the high error rate satellite links is the robustness of the protocol. We want to explore the performance behavior of HADAC for various packet error probabilities (PEPs) and especially, in cases where the packet error probability (PEP) is high. In order to prove that HADAC is robust we have to show that the decrease in the performance of the HADAC protocol is consistent with the increase in packet error probability (PEP), and in a way that the protocol still can deliver acceptable performance. We will get an answer to that by analyzing the results that will be presented in the following two graphs. Those results have been collected from simulating HADAC for various values of packet error probability (PEP)(e.g., we vary PEP from 0.1 to 0.5). Following the same pattern with previous experiments we assume that the packet error probabilities are independent and equal for each transmitted packet in C_1 or C_2. The y-axis represents the required Transmission Rounds metric and the x-axis represents the various values of PEP. We

collected the simulation results for various Air Cache sizes (i.e., 0,1, 3, 5, 7, 9 packets) and by assuming in each simulated scenario that the TG is consisted of 20 packets. We also, include in the graph the results that correspond to a non Air Cache ARQ-based protocol (e.g., the Air Cache size is 0) in order to compare the behavior of the HADAC protocol with a non Air Cache ARQ-based protocol. This comparison will help us more in studying and understanding, if air caching is advantageous for the robustness of the HADAC protocol. The first graph presents the results we collected for a multicast group size of 10^4 members and the second one presents the results that we collected for a multicast group of 10^5 members. In such a way, we will find out if the protocol still retains its performance for high PEP and large multicast group sizes. The two graphs along with the corresponding comments follow:



	0.1	0.2	0.3	0.4	0.5
AC(SxR)=0	6.09	8.24	11.1	14.64	18.79
AC(SxR)=1x1	5.84	7.88	10.34	13.37	17.39
AC(SxR)=3x1	5.33	7.35	9.56	12.06	15.65
AC(SxR)=5x1	5.04	6.74	8.83	11.33	14.66
AC(SxR)=7x1	4.93	6.36	8.28	10.62	13.56
AC(SxR)=9x1	4.45	6.16	7.84	10.2	12.97

Figure 5.13: (HADAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^4)

The above results correspond to the case where the multicast group size is consisted of 10^4 members. We varied the packet error probability (PEP), which as we mentioned is equal for each transmitted packet in any of the two transmission channels (C_1 or C_2), from 0.1 to 0.5. The first column in each group of columns represents the required Transmission Rounds in the case where the Air Cache size is 0 (e.g., non Air Cache ARQ-based protocol). The benefits of the Air Cache in high bit error rate links are obvious when we observe the results for packet error probability (PEP) equal to 0.4 or 0.5 and for Air Cache sizes that do not exceed the 50% of the size of the primary channel (C_1). For example in the case where the PEP is 0.5 and the maximum allowable Air Cache size is specified to be 9 packets (45% of C_1, which contains 20 packets) the

performance gain is translated to an improvement of almost 6 transmission rounds, which in terms of percentile is almost 33% improvement in required Transmission Rounds for the complete reliable transmission of 20 data packets to a multicast group of 10^4 members. So, the first observation is that the air caching technique boosts the robustness of the ARQ-based reliable multicasting protocols. The latter conclusion is true based on the comparison of the results that correspond to HADAC protocol and those that correspond to a non Air Cache ARQ-based protocol in a highly erroneous environment (e.g. where PEP is 0.4 or 0.5). Furthermore we want to explore the robustness of the HADAC protocol as we vary the Air Cache size and the multicast group size in order to study how the performance of the HADAC protocol is affected as we move from low PEP to high PEP. The results that we collected provide us with the appropriate information to study, analyze and draw conclusions about the robustness of the protocol. From the two graphs that correspond to the robustness of HADAC, we analyze the performance behavior of the HADAC protocol as we move from low PEP environments to high PEP ones. The rate of degradation of the protocol's performance is less than linear and this observation is indicative of the robustness of the protocol. The mathematical proof of the latter observation (e.g., the rate of performance degradation is less than linear as we increase the PEP) is sketched below:

Sketch of proof: If we get the first two values of the required Transmission Rounds metric that correspond to the same value of Air Cache size and multicast group size for PEP 0.1 and 0.2 respectively we can construct a line that is described from the function:

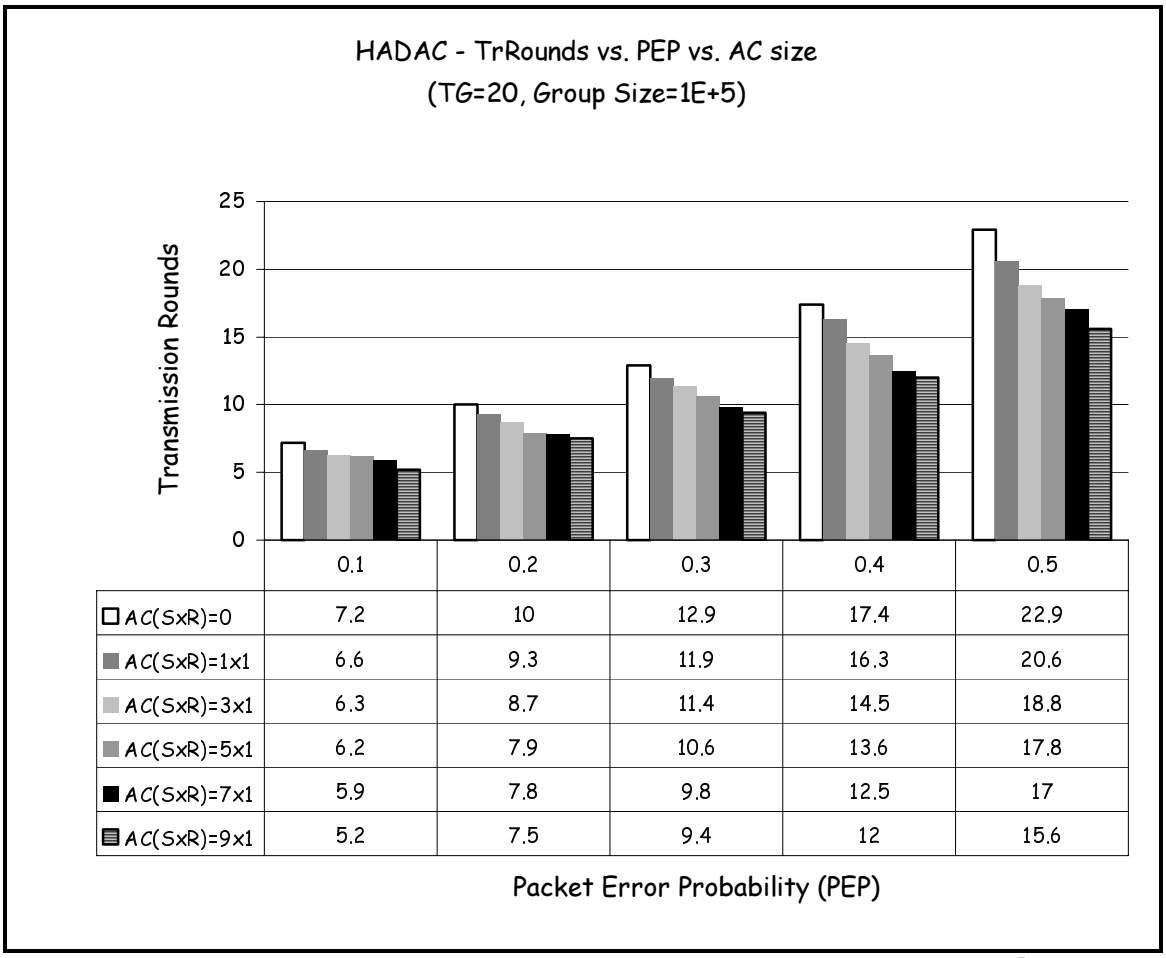
$$RTfunc \equiv \mathrm{Re}\, qTRounds(PEP) = a * PEP + b \text{ where a,b constants and a,b} \in \mathbb{R}$$

Then if we compute the value of required Transmission Rounds (*RTfunc*) for higher values of PEP using the above function, and compare this value to the *RTsim value,* which is the corresponding, value taken from the simulation results then we will observe that:

$$RTsim \leq RTfunc$$

in all cases where the PEP is high ($PEP \geq 0.3$). The latter result shows that the degradation rate in performance of HADAC is slow and proves that the protocol is robust even in cases with high PEP and large multicast group sizes, so the protocol presents promising robustness characteristics among the protocols that belong to the class of ARQ based reliable multicasting protocols.

The properties for the robustness of the HADAC protocol that we highlighted in the previous paragraph hold also for the case where the multicast group size has 10^5 members, as this can be concluded from the results, which are presented in the following graph. The observation that the robustness properties of the HADAC protocol hold even if the multicast group size increases significantly; proves that the protocol can operate with acceptable performance even for large group sizes in highly erroneous environments. Those robustness characteristics of the protocol along with its performance characteristics and the nearly efficient utilization of the Air Cache bandwidth prove that the HADAC protocol is an interesting, well-behaved and very promising protocol among the class of protocols that are ARQ-based.



	0.1	0.2	0.3	0.4	0.5
AC(SxR)=0	7.2	10	12.9	17.4	22.9
AC(SxR)=1x1	6.6	9.3	11.9	16.3	20.6
AC(SxR)=3x1	6.3	8.7	11.4	14.5	18.8
AC(SxR)=5x1	6.2	7.9	10.6	13.6	17.8
AC(SxR)=7x1	5.9	7.8	9.8	12.5	17
AC(SxR)=9x1	5.2	7.5	9.4	12	15.6

Figure 5.14: (HADAC) Tx Rounds vs. PEP vs. AC Size (TG=20, GS=10^5)

Furthermore, if we compare the results that we collected for HADAC with the corresponding results that we collected for a non Air Cache ARQ-based protocol, when the PEP is high, we will observe that the performance improvement in terms of the required Transmission Rounds when the multicast group has 10^5 members is greater compared to the case where the multicast group size has 10^4 members. For example when the multicast group is consisted of 10^4 members, the PEP is 0.5 and the Air Cache size is 9 (45% of the TG size) the HADAC requires almost 6 Transmission Rounds less than a non Air Cache ARQ-based protocol. But in the case where the multicast group is consisted of 10^5 members, the PEP is 0.5 and the Air Cache size is 9, HADAC requires

almost 7.5 less transmission rounds for the reliable delivery of 20 data packets. The latter observation is one more proof of the fact that air caching boosts both the performance and the robustness of the HADAC protocol.

Based on the above simulation results we gained an insight of the HADAC's performance behavior and we showed that it could be useful if it is applied in real world scenarios. In the following paragraph we are going to give an overview of the results corresponding to the delay, scalability, efficient utilization of Air Cache bandwidth and robustness of the protocol and we will make some comments on how those results are affected in various network environments, when those environments are characterized by parameters like the PEP, the multicast group size and the Air Cache size. Actually, what follows is a collection of the results that have been already presented in this section, in order to get a more complete view of the performance behavior of HADAC protocol.

5.5.3 Performance Analysis

As we have already stated, in this paragraph we are going to demonstrate the advantages and disadvantages of the HADAC protocol based on the observations that correspond to the results that we collected throughout the simulations for a variety of performance metrics, as they were presented in the previous paragraph.

Initially, we demonstrate a collection of advantages that the HADAC protocol presents based on the simulation results. The group of results that demonstrates some of the advantages of the HADAC protocol are analyzed and compared in the context of the performance that the class of the ARQ-based protocols, typically present.

As we mentioned HADAC is very similar to the ACDAC protocol and also is the adaptive counterpart of the UDPAC and RDPAC protocols. Based on these similarities the advantages of the HADAC protocol are related to the performance behavior and the characteristics of those protocols (e.g., ACDAC, RDPAC, UDPAC). Compared to the latter protocols, HADAC presents the best behavior among the class of ARQ-based Air Cache reliable multicasting protocols. This result was expected in terms of the comparison of HADAC with its non-adaptive counterparts (UDPAC and RDPAC), since the Air Cache in HADAC is updated based on the feedback from the receivers, so it contains the most appropriate data packets for the recovery of the erroneous or corrupted data packets at the receiving ends, in contrast to UDPAC and RDPAC where there is no utilization of feedback for the update of the Air Cache, which is refreshed by randomly selecting packets from the TG. The question that arises is why HADAC is considered to have better performance than ACDAC, since in both the protocols the Air Cache content is updated based on the same algorithm. The answer to the latter question is based on the size adaptation of the Air Cache in the case of HADAC. Even though both protocol present the same performance characteristics in terms of the required Transmission Rounds, HADAC, due to the size adaptation of the Air Cache, can achieve this performance by utilizing in average less bandwidth per transmission round for air caching compared to the corresponding bandwidth utilized by ACDAC, since the latter is not adapting the size of the Air Cache, but it keeps it constant throughout the reliable transmission.

Apart from the comparison of HADAC with the rest of the proposed Air Cache ARQ-based protocols, we examined the performance of HADAC protocol compared to a non

Air Cache ARQ-based protocol. The performance of HADAC protocol is better than the performance of a non Air Cache ARQ-based protocol in terms of the required Transmission Rounds metric, which is a result that was expected due to the application of the adaptive air caching in HADAC.

Since we have observed that HADAC presents better performance compared to the rest of the protocols that belong to the ARQ-based class of protocols, the question that arises is if this performance is sufficient enough to compensate for the bandwidth, which is utilized per transmission round for air caching. In order to answer the last question we used the Normalized Gain ratio, as we did for each one of the proposed Air Cache reliable multicasting protocols. Even though we expect that the following condition

$$Normalized\ Gain \geq 1$$

will hold in order to be proven that HADAC utilizes efficiently the Air Cache bandwidth and the improvement in performance is sufficient enough to compensate for the bandwidth consumption due to air caching, the values of Normalized Gain ratio for the HADAC protocol are not always satisfy the above condition. The latter general observation, even though it is an indication, does not prove that HADAC utilizes inefficiently the Air Cache bandwidth, since the values of the Normalized Gain ratio are fluctuating around the critical value of one. Based also, on the fact that the performance of HADAC is better compared to a non Air Cache ARQ-based protocol, HADAC will be the protocol of choice when the available bandwidth is not the critical factor.

Compared to ACDAC protocol, HADAC presents similar performance characteristics, but the factor that establishes HADAC as better in the class of adaptive Air Cache ARQ-based protocols is the adaptation in size of the Air Cache, which results

in less bandwidth utilization from the Air Cache per transmission round. So, HADAC can achieve the same performance with ACDAC but with less bandwidth used for air caching.

Finally, the most important advantage of HADAC protocol is the processing power and memory requirements that HADAC poses to the participating network entities. From this point of view, HADAC is a lightweight protocol since those requirements are not very demanding and devices that are not very powerful and do not have large memory available for buffering can apply HADAC. The latter observation is very important because in some cases the important factor for the survivability of the network is not the performance but the power consumption of the applied protocol. Also, other important factors are the size and the cost of the participating devices, so the lightweight protocols are preferable, even though may not have very promising performance characteristics.

Since we presented the advantages of HADAC, we will mention the disadvantages that this protocol presents by comparing it to the non-adaptive Air Cache reliable multicasting protocols, to the non Air Cache reliable multicasting protocol and to the FEC-based protocols. By comparing the performance of HADAC with the non-adaptive ARQ-based protocols, it is obvious that even though we apply adaptation of the content of the Air Cache and we get performance improvement, this improvement is not so high. The problem is that HADAC utilizes the feedback from the receivers in each transmission round in order to adapt the content and size of the Air Cache, so the protocol has to wait in each transmission round for the collection of feedback, which is an action that will penalize the end-to-end delay compared to the non-adaptive protocols, which do not have to wait much before the initiation of the next transmission round. How

important is this disadvantage can be studied only by collecting results that have to do with actual time (e.g., *ms*, *secs*) for the end-to-end delay.

Compared to the FEC-based protocols, HADAC like all the ARQ-based protocols presents less promising performance characteristics, since there are no parity packets involved in the transmission and correction of the data packets. Each erroneous or corrupted packet can be recovered only by the retransmission of the corresponding data packet compared to the FEC-based protocols where each parity packet can correct any of the erroneous or corrupted data packets at the receiving ends.

Concluding, HADAC is a protocol that can be used in real world scenarios, where the important factor is not so much the performance but the specifications of the participating devices and the survivability of the network.

5.6 Comparison of the Proposed Protocols

In this section we present graphs, which demonstrate some comparison results that correspond to the performance of:

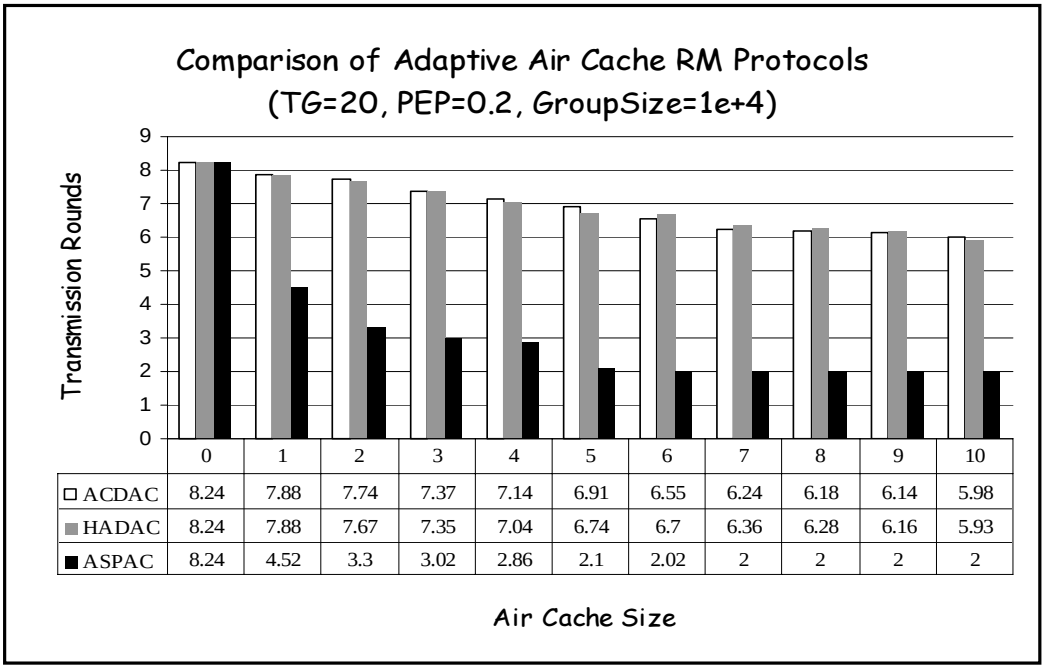
- The adaptive Air Cache reliable multicasting protocols
- The Air Cache reliable multicasting protocols (adaptive and non-adaptive)

Even though, throughout this chapter we highlighted the differences that the proposed adaptive Air Cache reliable multicasting protocols present relative to each other and relative to the non-adaptive Air Cache reliable multicasting protocols, in this paragraph we give some graphs in order to demonstrate explicitly and in a more concrete way, those differences.

In the following paragraphs we present a couple of graphs related to the comparison between the adaptive Air Cache reliable multicasting protocols and a couple of graphs that correspond to the comparison of the latter protocols to the non-adaptive Air Cache reliable multicasting protocols.

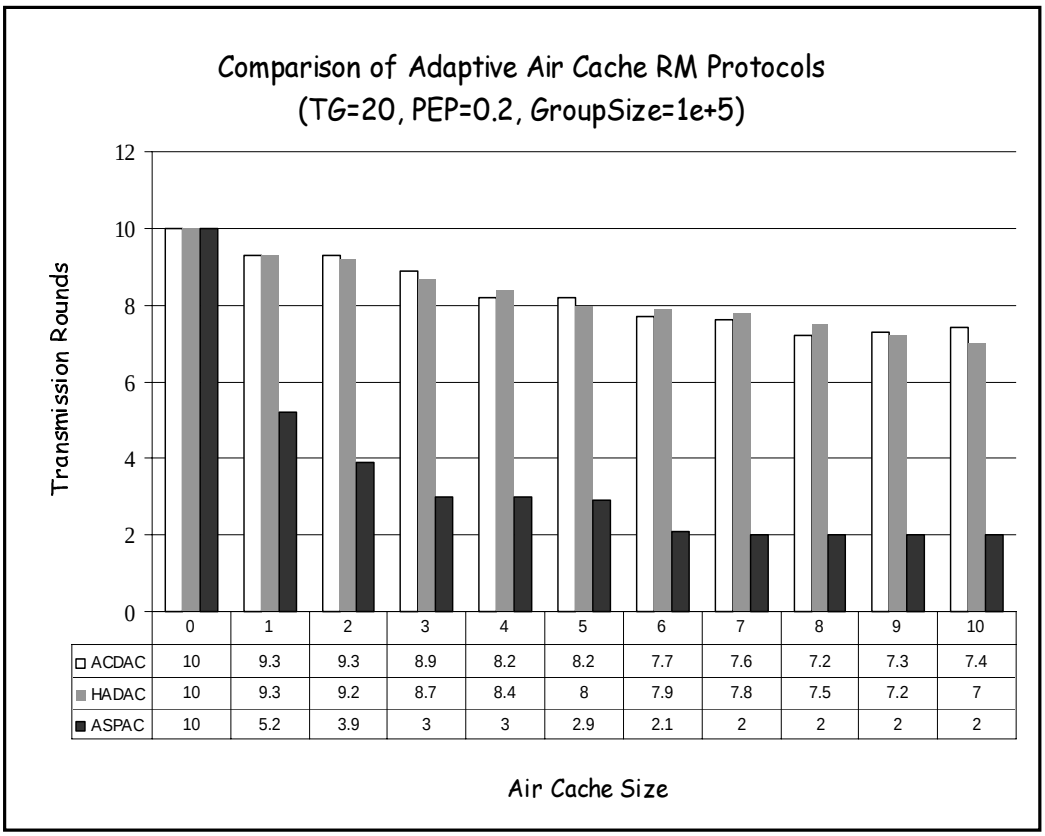
5.6.1 Comparison of the Adaptive Air Cache RM Protocols

The two graphs that follow demonstrate some comparison results that correspond to the performance differences that the adaptive Air Cache reliable multicasting protocols (ACDAC, HADAC, ASPAC) present. The following sets of results have been collected by simulating the adaptive Air Cache reliable multicasting protocols under the same assumptions for the network environment and the transmitted data. The first graph (figure 5.15) corresponds to the scenario where the multicast group has 10^4 members, and the second graph (figure 5.16) corresponds to the scenario where the multicast group has 10^5 members. The simulation results that are presented to the following two graphs have been collected by assuming that the packet error probability (PEP) is 0.2 and the TG is 20 data packets.



	0	1	2	3	4	5	6	7	8	9	10
ACDAC	8.24	7.88	7.74	7.37	7.14	6.91	6.55	6.24	6.18	6.14	5.98
HADAC	8.24	7.88	7.67	7.35	7.04	6.74	6.7	6.36	6.28	6.16	5.93
ASPAC	8.24	4.52	3.3	3.02	2.86	2.1	2.02	2	2	2	2

Figure 5.15: Comparison of Adaptive Air Cache RM Protocols (TG=20,PEP=0.2,GS=10^4)



	0	1	2	3	4	5	6	7	8	9	10
ACDAC	10	9.3	9.3	8.9	8.2	8.2	7.7	7.6	7.2	7.3	7.4
HADAC	10	9.3	9.2	8.7	8.4	8	7.9	7.8	7.5	7.2	7
ASPAC	10	5.2	3.9	3	3	2.9	2.1	2	2	2	2

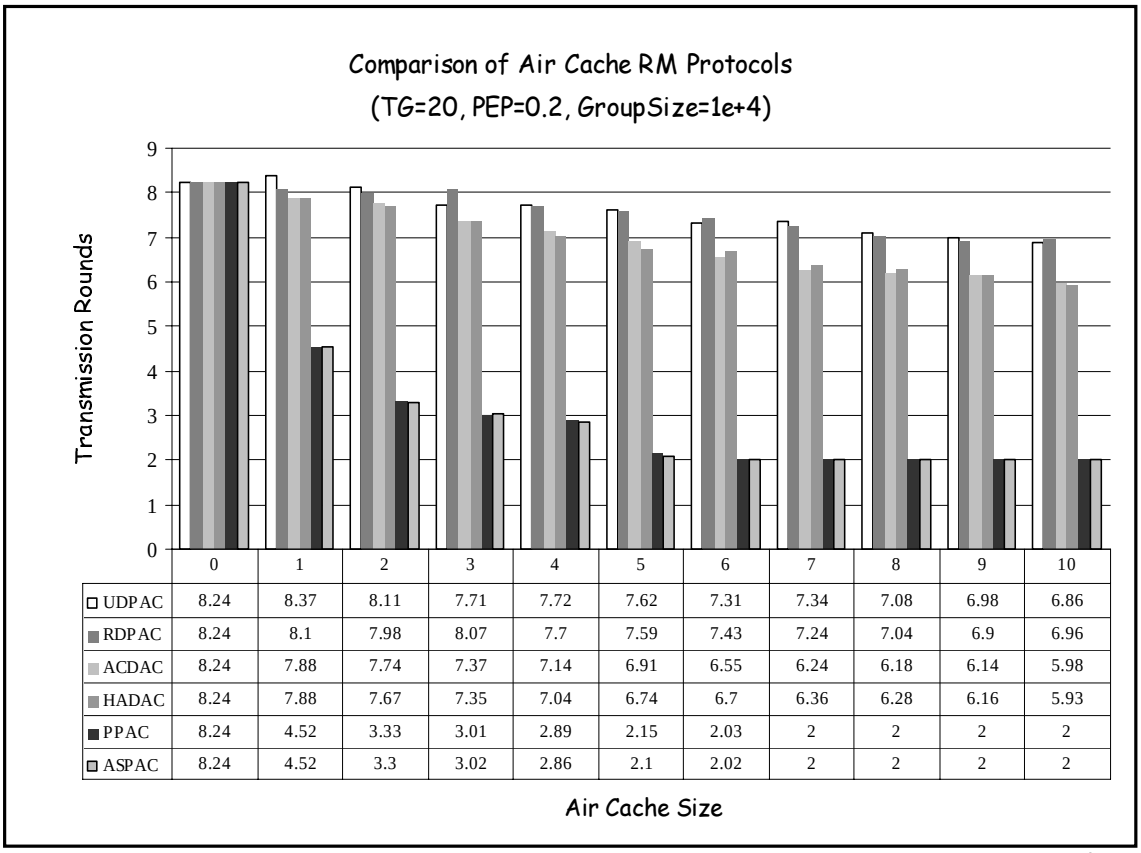
Figure 5.16: Comparison of Adaptive Air Cache RM Protocols (TG=20,PEP=0.2,GS=10^5)

The above two graphs are the proof to the observations that we did for the performance behavior of each one of the adaptive Air Cache reliable multicasting protocols throughout this chapter. Briefly those observations are:

- ASPAC has very promising performance characteristics compared to the ACDAC and HADAC protocols. The latter observation is based on the fact that ASPAC is a FEC-based protocol, since the Air Cache is filled with parity packets, in contrast to the ARQ-based protocols (HADAC and ACDAC), where the Air Cache is filled with data packets. The effectiveness of the combination of the air caching with FEC for the error correction of the corrupted or erroneous data packets at the receivers is obvious from the difference in performance of ASPAC compared to ACDAC and HADAC.
- By analyzing concurrently the results of figure 5.15 and the results of figure 5.16, we get some insight on the scalability characteristics of those protocols. By increasing the multicast group size from 10^4 members to 10^5 members, the performance of ASPAC remains almost unchanged compared to the increase of required Transmission Rounds (e.g., the increase is almost a single transmission round) in the case of ACDAC and HADAC. ASPAC is very scalable and this is due to the combination of FEC with air caching.

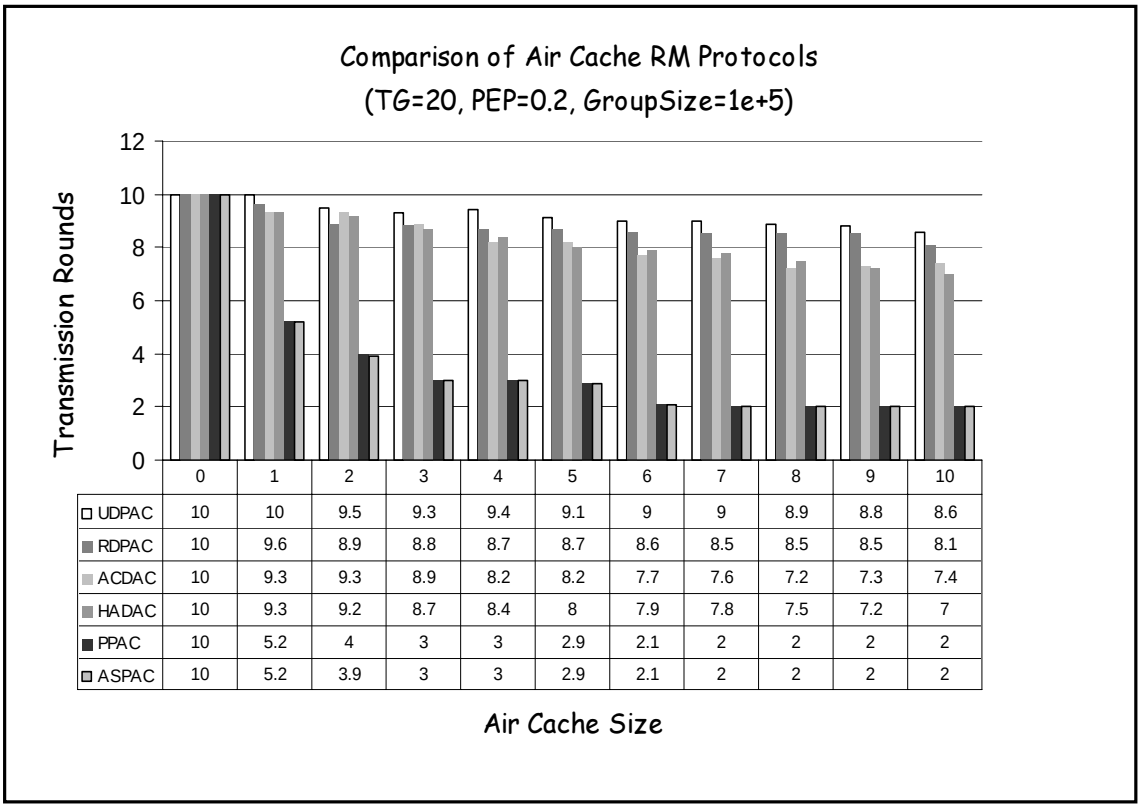
5.6.2 Adaptive vs. Non-Adaptive Air Cache RM Protocols

In this paragraph we present a couple of comparison graphs that correspond to the performance behavior in terms of the required Transmission Rounds metric, of both the adaptive and non-adaptive Air Cache reliable multicasting protocols



	0	1	2	3	4	5	6	7	8	9	10
UDPAC	8.24	8.37	8.11	7.71	7.72	7.62	7.31	7.34	7.08	6.98	6.86
RDPAC	8.24	8.1	7.98	8.07	7.7	7.59	7.43	7.24	7.04	6.9	6.96
ACDAC	8.24	7.88	7.74	7.37	7.14	6.91	6.55	6.24	6.18	6.14	5.98
HADAC	8.24	7.88	7.67	7.35	7.04	6.74	6.7	6.36	6.28	6.16	5.93
PPAC	8.24	4.52	3.33	3.01	2.89	2.15	2.03	2	2	2	2
ASPAC	8.24	4.52	3.3	3.02	2.86	2.1	2.02	2	2	2	2

Figure 5.17: Comparison of the Air Cache RM Protocols (TG=20,PEP=0.2,GS=10^4)



	0	1	2	3	4	5	6	7	8	9	10
UDPAC	10	10	9.5	9.3	9.4	9.1	9	9	8.9	8.8	8.6
RDPAC	10	9.6	8.9	8.8	8.7	8.7	8.6	8.5	8.5	8.5	8.1
ACDAC	10	9.3	9.3	8.9	8.2	8.2	7.7	7.6	7.2	7.3	7.4
HADAC	10	9.3	9.2	8.7	8.4	8	7.9	7.8	7.5	7.2	7
PPAC	10	5.2	4	3	3	2.9	2.1	2	2	2	2
ASPAC	10	5.2	3.9	3	3	2.9	2.1	2	2	2	2

Figure 5.18: Comparison of the Air Cache RM Protocols (TG=20,PEP=0.2,GS=10^4)

The performance comparison is done in terms of required Transmission Rounds for the reliable transmission of a group of 20 data packets (e.g., TG=20) for various Air Cache sizes, varying from 0 packets (e.g., RM protocols without Air Caching) to 10 packets. A first observation is that the protocols that are based on FEC and Air Caching perform much better than the protocols that use ARQ and Air Caching, which is a result that was expected. Also, the performance of the protocols, which rely on air caching, is better compared to the case where there is no application of the air caching technique (e.g., no Air Cache). On the other extreme, the performance improvement is saturated as we increase the Air Cache size after a point. Observing and analyzing the results from the point of view of comparing the adaptive protocols and their non-adaptive counterparts we can drawn some very important conclusions:

- Even though the size of Air Cache is adapted in ASPAC, which results in using overall less bandwidth per transmission round (see figures 5.17 and 5.18), there is no degradation in performance compared to PPAC. The latter protocol uses constant Air Cache size, equal to the maximum Air Cache size that ASPAC is allowed to use per transmission round, throughout the reliable transmission.
- ACDAC presents better performance characteristics compared to its non-adaptive counterparts (e.g., UDPAC, RDPAC), because of the content adaptation of the Air Cache, which contains the most requested packets in each transmission round.
- HADAC compared to its non-adaptive counterparts (UDPAC, RDPAC) presents better performance characteristics because of the content adaptation of the Air Cache.

- HADAC compared to ACDAC presents very similar performance characteristics, because both of them adapt the content of the Air Cache. On the other hand, HADAC is more advantageous because of the combined content and size adaptation. Even though HADAC has similar performance characteristics with ACDAC, uses overall less bandwidth per transmission round.

Chapter 6: Conclusions and Future Work

6.1 Overview of the Chapter

In this chapter, we collect some of the general conclusions that we reached to, when we were evaluating the proposed reliable multicasting protocols. The detailed conclusions for each proposed protocol have been already presented in chapters 3, 4 and 5. The goal of this general recollection of observations on the performance and functionality of the protocols, will help us to recall some of the general characteristics that those protocols present based on the class (e.g., adaptive or non-adaptive) that they belong to and based on the techniques (e.g., FEC, ARQ and Air Caching) that their operation is based to. Finally, in the last section of this chapter we will give some directions for future work based on the protocols and techniques that we presented here.

6.2 Conclusions

In this paragraph we will be presenting some general conclusions that we draw from the evaluation of the proposed protocols.

- Protocols based only on ARQ and not in FEC do not present very promising performance characteristics but are lightweight protocols. The latter characteristic makes those protocols favorable, since the ARQ-based protocols are not very demanding in terms of the memory and processing requirements that pose to the participating network devices. So, those protocols can be accommodated from

devices that do not have powerful processors and large amount of buffering space, such as the handheld devices are. Also those lightweight characteristics reflect on the battery power that those protocols utilize. For example, low power processors use less battery power, so the lifetime of the device becomes longer.

- The protocols, which based on FEC and also use Air Caching present very promising performance characteristics. The performance gain that we get by mixing those two techniques is sufficient to hide the extra bandwidth usage, which is due to air caching. We proved the latter by defining the Normalized Gain metric. The FEC-based protocols are more demanding (e.g., compared to the protocols that are based on ARQ) in terms of the processing and buffering characteristics that the participating devices are required to satisfy in order to accommodate those reliable multicasting protocols, due to the use of parity packets and the online encoding/decoding that takes place in FEC
- The functionality of the non-adaptive protocols is not relying on the feedback from the receivers, so this results in using the Air Cache not as efficient as they could do. We use constant Air Cache size and the contents of the Air Cache in UDPAC and RDPAC are chosen at random. So, it is unavoidable that we will not get the best possible performance from those protocols and the reserved bandwidth for air caching will not be utilized efficiently, or even we will reserve redundant bandwidth, which is ineffective in improving further the performance of the corresponding protocol (PPAC). For these reasons we introduced the class of adaptive reliable multicasting protocols, where the size and content of the Air Cache are adapted based on the feedback from the receivers. By adapting the Air Cache size per transmission round

we hope that we will get the same performance characteristics compared to the protocols where we assumed constant Air Cache size but by using less bandwidth in average, dedicated to Air Cache. By adapting the content of the Air Cache we expect to get much better performance characteristics compared to the corresponding protocols that do not adapt the content of the Air Cache per transmission round. By studying the non-adaptive and adaptive classes of reliable multicasting protocols we concluded that the size and content adaptation gives us the expected results in terms of bandwidth savings and performance improvement, respectively.

- Apart from the performance of the protocols in terms of the *required Transmission Rounds* for the completition of the reliable transmission of the TG, we investigated the protocols in terms of their robustness. We can classify the proposed protocols in three different categories.

  Class I

  In this class belong the protocols UDPAC and RDPAC, which present robustness characteristics similar to those protocols that rely solely on ARQ for the error recovery of corrupted or erroneous packets. Actually, the performance of the protocols that are based on ARQ degrades exponentially as we move to higher packet error probabilities. The UDPAC and RDPAC protocols present lower degradation rate in performance than the latter protocols due to the application of air caching, even though the degradation rate of the performance of those protocols remains exponential.

Class II

The second class is consisted from the ACDAC and HADAC protocols. Those protocols are the adaptive version of the UDPAC and RDPAC protocols, whose functionality is not based on FEC. Because of the adaptation characteristics that the ACDAC and HADAC protocols present, the degradation rate of their performance is much lower (e.g., compared to their non-adaptive counterparts) as we move to higher packet error probabilities (PEPs). On the other hand, there is still space for improvement in terms of the robustness characteristics that a reliable multicasting protocol can achieve. That is because ACDAC and HADAC are not based on FEC, which is proven to be a very effective technique for the error recovery of erroneous or corrupted packets at the receiving ends.

Class III

In this class belong the protocols that present very promising robustness characteristics. The main characteristic of those protocols compared to the above-mentioned ones, is the use of FEC as an error recovery method. PPAC and ASPAC belong to this class of protocols, that their performance degrades extremely slow as we move to higher packet error probabilities (PEPs). This is due to the use of parity packets, which equally can compensate for the loss or corruption of any of the data packets of the TG.

- In the performance evaluation of the protocols we also, studied the scalability characteristics of the protocols, since this is a very important characteristic for the success of the reliable multicasting protocols, due to the popularity of the satellite

services and the growth rate of the population that uses those services. In the same fashion as before, we divide the proposed protocols in three classes based on their scalability characteristics.

Class I

In this class belong the protocols that their scalability characteristics does not differ much from the characteristics of those protocols that are based on ARQ for error recovery. The latter protocols are characterized from the exponential degradation of their performance as we move to larger multicast group sizes. In this class belong the UDPAC and RDPAC protocols, which are based on the combination of ARQ with air caching for the error recovery of the erroneous or corrupted data packets. The degradation rate of the performance of those protocols is a little bit better than the degradation rate of the performance of protocols that are non Air Cache ARQ-based, even though the rate is still exponential.

Class II

This class of protocols, in terms of their scalability characteristics, involves the protocols that present better behavior than the reliable multicasting protocols of the previous class (e.g., UDPAC and RDPAC). The reliable multicasting protocols that belong in this class are the ACDAC and HADAC, which are the adaptive counterparts of the UDPAC and RDPAC protocols. The adaptation characteristics of the ACDAC and HADAC protocols are the reason for their better scalability characteristics. Even though the latter protocols are more scalable, still the degradation rate of their performance is not very promising, as the multicast group

gets larger. That is due to the fact that those protocols do not use FEC for the error recovery of corrupted or erroneous data packets. So, there is still space for improvement in terms of the scalability characteristics that a reliable multicasting protocol can achieve.

Class III

In this class belong the protocols that present very promising scalability characteristics. Those protocols are the PPAC and ASPAC, and their success is based on the use of FEC technique combined with air caching for the error recovery of corrupted or erroneous data packets. The degradation rate of their performance is very slow, as the multicast group gets larger. Those protocols have been proven very scalable since they perform very well even when the multicast group size is of the order of hundred of thousands members.

6.3 Future Work

We conclude this chapter by making some suggestions for future work based on the study that we presented in this thesis. The directions for future research are based mostly on the further evaluation of the protocols that we proposed here. Some of these future directions follow:

- Develop a mathematical model for the above protocols, or in general for reliable multicasting protocols that combine classical error recovery methods like FEC and ARQ with newly introduced techniques like Air Caching. In such a way we can

optimize more the functionality of the protocols, we can check the correctness and accuracy of the simulation results and we can study the scalability and robustness characteristics of these protocols under known confidence bounds.

- We suggest the use of performance metrics like end-to-end latency in terms of actual time (i.e., *ms, secs*) and not in terms of the *required Transmission Rounds* metric, which we have used throughout this work. In a such a way, we can evaluate the performance characteristics of the protocols more accurately since the time will reflect also important factors like the propagation delay, the delay introduced be waiting for feedback, etc. So, more detailed and finer performance metrics are required for more extensive study of these protocols. In this thesis we just proved the performance potential that some of the protocols have, and by analyzing the results that we collected from the simulation of these protocols, it is obvious that they deserve to be studied further.
- One more suggestion would be to use a well-known simulator like OPNET or NS-2 for the simulation of these protocols. These simulators are more detailed and can give more accurate results, since they include details that we could not include in our customized simulator. By doing that we can get a more complete picture for the performance behavior of the protocols.
- A final suggestion would be to investigate further the memory and processing requirements of the proposed protocols and what are their effects on the correct and efficient functionality of each one of the proposed reliable multicasting protocols. Also, through this study we can find out, in a more accurate way, the requirements of each one of the proposed protocols and furthermore what devices can accommodate

them based on nowadays technology. Through this study will have also the opportunity to demonstrate the usefulness of the protocols like UDPAC, RDPAC, ACDAC and HADAC, which we have already claimed that are lightweight and can be accommodated from devices, which are not powerful, in contrast to the demanding FEC-based protocols, like PPAC and ASPAC, which require powerful devices in order to achieve their potential performance.

BIBLIOGRAPHY

[1] M. Allman et al., “Ongoing TCP Research Related to Satellites.” draft-ieft-tcpsat-res-issues-09.txt.

[2] K. Stathatos, N. Roussopoulos, J.S. Baras, “Adaptive Data Broadcasting Using Air-Cache” 1st International Workshop on Satellite-based Information Services, Rye, New York, Nov. 1996, pp. 30-37

[3] R. Braudes, S. Zabele. “ Requirements for Multicast Protocols” RFC 1458, May 1993.

[4] C. Diot, W. Dabbous, J. Crowcroft. “Multipoint communication: A survey of protocols, functions and mechanisms”. IEEE Journal on Selected Areas in Communications, vol.15, no.3, p. 277-90. April 1997.

[5] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang. “A reliable multicast framework for light-weight sessions and application level framing”. Comput. Commun. Rev. (USA), Computer Communication Review, vol.25, no.4, p. 342-56, September 1995

[6] D. Gossink, J. Macker. “Reliable Multicast and Integrated Parity Retransmission with Channel Estimation Considerations”. IEEE Globecom, 1998.

[7] Konstantinos Stathatos, Nick Roussopoulos, John S. Baras, “Disseminating Updates to Mobile Clients”, CSHCN TR 98-16

[8] Eddie C. Shek, Son Dao, Yongguang Zhang, Darrel J. Van Buer: Dynamic Multicast Information Dissemination in Hybrid Satellite-Wireless Networks. MobiDE 1999: pp. 30-35

[9] Nonnenmacher, J., Biersack, E. W.: "Reliable Multicast: Where to use FEC", Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96), Sophia Antipolis, France, October 1996, pp. 134-148

[10] S. Kasera, J. Kurose, D. Towsley. “A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast”. Proceedings IEEE INFOCOM 1998, San Francisco, CA, USA March 1998.

[11] P. Rodriguez, E. W Biersack, K. W. Ross "Improving the WWW: Caching or Multicast?" 1998 Web Cache Workshop. http://wwwcache.ja.net/events/workshop/papers.html

[12] J. Macker. “Reliable multicast transport and integrated erasure-based forward error correction”. MILCOM 97 Proceedings. vol. 2, p. 973-7, November 1997.

[13] Rodriguez and E.W. Biersack. “Bringing the Web to the Network Edge: Large Caches and Satellite Distribution”. In Proceedings of WOSBIS '98, ACM/IEEE MobiCom Workshop on Satellite-based Information Services, October 1998.

[14] A.J. McAuley, “Reliable broadband communication using a burst erasure correcting code”. Computer Communication Review, vol.20, no.4, p. 297-306, Philadelphia, PA. September 1990 pp. 287-306.

[15] K. Miller. *Multicast Networking and Applications.* Reading, Massachusetts, Addison Wesley, 1999

[16] J. Nonnenmacher and E. W. Biersack, "Optimal multicast feedback," in IEEE Infocom, (San Francisco, California), p. 964, March/April 1998

[17] J. Nonnenmacher, E. Biersack, and D. Towsley. “Parity-based loss recovery for reliable multicast transmission”. IEEE/ACM Transactions on Networking, vol.6, no.4, p. 349-61, August 1998.

[18] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, G. Carle. “How bad is reliable multicast without local recovery?”. Proceedings. IEEE INFOCOM '98, vol. 3, p. 972-9, 1998.

[19] J. Nonnenmacher and E. W. Biersack. “Scalable Feedback for Large Groups”. IEEE Transactions on Networking, vol. 7, no.3, p. 375-386. June 1999.

[20] Rubenstein, D., Kurose, J., and Towsley, D. Real-Time Reliable Multicast Using Proactive Forward Error Correction. In Proceedings of NOSSDAV 1998 (Cambridge, England, July 1998), ACM

[21] Timur Friedman and Don Towsley. Multicast session membership size estimation. In Proceedings of IEEE INFOCOM, New York, NY, pages 965-972, March 1999

[22] S. Paul, K.K. Sabnani, J.C. Lin, and S. Bhattacharya, “Reliable multicast transport protocol (RMTP)”. IEEE Journal on Selected Areas in Communications, vol.15, no.3, p. 407-21, April 1997.

[23] Papadopoulos C., Parulkar G., Varghese. “An error control scheme for large-scale multicast applications”. Proceedings. IEEE INFOCOM '98, vol 3, p. 1188-96, 1998.

[24] L. Rizzo. “Effective erasure codes for reliable computer communication protocols”. Computer Communication Review, vol.27, no.2, p. 24-36, April 1997.

[25] D Rubenstein, S. Kasera, D. Towsley, and Jim Kurose. “Improving Reliable Multicast Using Active Parity Encoding Services (APES)”. Proceedings IEEE INFOCOM 1999 vol. 3, p. 1248-1255, March 1999.

[26] D. Towsley and J. Kurose. “A comparison of sender-initiated and receiver-initiated reliable multicast protocols”. IEEE Journal on Selected Areas in Communications, vol.15, no.3, p. 398-406, April 1997.

[27] J. Nonnenmacher and E.W. Biersack, "Asynchronous Multicast Push: AMP." In Proc. of International Conference on Computer Communications, Cannes, France, November 1997

[28] Grossglauser M., "Optimal Deterministic Timeouts for Reliable Scalable Multicast," IEEE Journal on Selected Areas in Communication, vol. 15, pp. 422--433, April 1997

[29] Almeroth, K.C.; Ammar, M.H.; Zongming Fei; "Scalable delivery of Web pages using cyclic best-effort multicast" Proceedings IEEE INFOCOM'98 Conference on Computer Communications. April 1998. p. 1214-21 vol.3

[30] K. Almeroth, Y. Zhang. “Using Satellite Links as Delivery Paths in the Multicast Backbone (MBone)”. Proceedings WOSBIS 1998. p. 47-53. October, 1998.

[31] M. Hofmann, “A generic concept for large-scale multicast”, Proceedings 1996 International Zurich Seminar on Digital Communications, IZS 96. p. 95-106. February 1996.

[32] C. Huitema. “The case for packet level FEC”, Proceedings IFIP 5 th International Workshop on Protocols for High Speed Networks (PfHSN’96), Sophia Antipolis, France, pp. 110-120, October. 1996.

[33] M. Jung, J. Nonnenmacher, E. Biersack. “Reliable Multicast via Satellite: Uni-directionalversus Bi-directional Communication”. Proceedings of KiVS 1999.

[34] S. Lin and D.J. Costello. *Error Correcting Coding: Fundamentals and Application*s. Englewood Cliffs, NJ: Prentice Hall, 1983

[35] J. Macker, Klinker, M. Corson. “Reliable multicast data delivery for military networking”, MILCOM 96 Proceedings. vol. 2, p. 399-403, October 1996.

[36] K. Miller, K. Robertson, A. Tweedly, M. White. “StarBurst Multicast File Transfer Protocol (MFTP) Specification”. Internet Draft, Work in Progress, draft-miller-mftp-spec03.txt, April 1998.

[37] J. Nonnenmacher and E. W. Biersack. “The impact of routing on multicast error recovery,” Computer Communication Review, vol.21, no.10, p. 867-79, July 1998.

[38] J. Nonnenmacher and E. W. Biersack. “Performance modelling of reliable multicast transmission”, Proceedings IEEE INFOCOM '97, vol. 2, p. 471-9, April 1997

[39] M. Yamamoto, J. Kurose , D. Towsley and H. Ikeda. “A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols”. Proceedings IEEE INFOCOM '97, p. 480-8, vol.2, April 1997.

[40] S. M. Payne, J. S. Baras, “Reliable Multicasting via Satellite: Delay Considerations”, Proc. of ATIRP, 4th Annual Symposium, March 21-23,2000.