CCIT Report #407
December 2002

Scalable Reliable Multi-rate Point to Multi-point Distribution of Bulk Data

Diego A. Crupnicoff and Yitzhak Birk
diego@mellanox.co.il birk@ee.technion.ac.il

Report CCIT #407, Center for Communications and Information Technologies, Electrical Engineering Department, Technion - Israel Institute of Technology, Haifa 32000, Israel

Abstract

In this work, we address the problem of reliable multicast distribution of bulk data to a large set of independent clients. The clients are not coordinated; i.e., they may start data reception at any point in time. Clients are heterogeneous in the sense that their maximum achievable data rates from the server vary widely, both among clients and with time. Moreover, the network may become congested and packets may be lost. Our goal is to permit each client to receive all the data as early as possible, while minimizing network resource utilization. The problem being addressed represents mass distribution of bulk data, such as software, software updates, content distribution, etc. This work demonstrates that any simple, erasure correcting codes computed over small blocks of data, can be used for this purpose in a near optimal way.

Our transmission scheme comprises a set of co-scheduled channels, whose transmission rates increase exponentially. Every channel carries all the data cyclically, along with redundant packets computed using an erasure correcting code. (Note that, due to the perpetual nature of the transmission, the use of redundancy does not cause the transmission of any "extra" data.) This code is applied to small, fixed-size groups of packets. The key element in this scheme is the partitioning into small groups, which permits the efficient use of standard codes, the smart scheduling of the transmission within each channel, and the interleaving of the channels. We show that a user that subscribes to any contiguous set of channels including the slowest one must only receive an amount of data equal to the original file size, regardless of starting time. This remains nearly true even if packets are lost or the user changes its subscription. At the same time, the network is used efficiently, in that the data rate carried over a given link is no greater than that required by the fastest downstream subscriber. The basic scheme is then extended in several ways to permit finer granularity of the user data rates, and is shown to remain near-optimal even if network conditions and subscriptions change dynamically. One of these extensions also applies to schemes that use an erasure correcting code that is computed over the entire file. Finally, the scheme is highly scalable because the only interaction with users is their subscription to channels.

Glossary and Notation

Chapter 2

q - Average packet loss rate

p - 1-q

G - File Size in packets

N^1- random variable which represents the number of times that a certain packet was sent until it was finally succesfuly received by a client

P - various probabilities

c - cycle index

N^G- random variable which represents the number of times (cycles) that G packets (the file) were sent until they were all succesfuly received by a client

i - packet index (0..G-1)

$d_{c,i}$ - number of times packet i has been sent up to cycle c. used in chapter 2 for the optimality proof of the simple packet schedule.

g - packet index (0..G-1) within a cycle

$P^G_{c,g}$- Probability that a client successfully received the G packets of a file after c complete cycles of G packets each and g more packets.

O - Optimal Packet Schedule

O_i - The ID of the packet sent at time slot i under packet schedule O

C - Candidate Packet Schedule

C_i - The ID of the packet sent at time slot i under packet schedule C

B - Better Packet Schedule (than C)

B_i - The ID of the packet sent at time slot i under packet schedule B

N^G_S - the random variable that represents the number of packets transmitted until succesfull reception of the *G* packets in the file under scheduling *S*. Where *S* is one of: *O* (the optimal packet schedule), *C* or *B*.

$I^{G,g}_{Si}$- Indicator. It is 1 if the packet in time slot i under schedule S is the one with ID g. It is 0 otherwise.

$F^{G,g}_{Si}$- Number of times the packet with ID g is sent at or before time slot i under packet schedule S.

i - packet IDs

j,k - indexes

m,n - time slot in packet schedule

U - random variable which represents the number of times that a certain packet was sent until it was finally succesfuly received by a client using an ideal reliable unicast protocol

U^G- random variable which represents the number of packet that were sent until a file of G packets was finally succesfuly received by a client using an ideal reliable unicast protocol

X^c - random variable which is the number of successful packet receptions out of *c* packets that were sent.

b - parameter for Gaussian distribution table lookup

P_L- Probability that the last trasmitted packet was lost.

P_S- Probability that the last trasmitted packet was successfully received.

α - Probability of transission from Loss to Success.

β - Probability of transission from Success to Loss.

Chapter 3

K - Number of data packets in FEC group.

N - Number of coded packets in FEC group.

G - Number of FEC groups in File.

g - Group index (0..G-1).

S - File Size in packets (S=GK).

q - Average packet loss rate

p - 1-q

X^c - random variable which is the number of successful packet receptions out of *c* packets that were sent.

V^c_m- Probability that X^c is equal to *m*.

P - various probabilities

c - cycles

$N^{G,K}$ - Random variable for the number of packets needed to receive a file of GK packets that is divided into G groups of K packets to which FEC is applied.

$P^{G,K}_{c,g}$- Probability that $N^{G,K}$ is equal to $cG+g$ which means success after c cycles of G packets plus g more packets.

O - Optimal Group Schedule

O_i - The ID of the group sent at time slot i under packet schedule O

C - Candidate Group Schedule

C_i - The ID of the group sent at time slot i under packet schedule C

B - Better Group Schedule (than C)

B_i - The ID of the group sent at time slot i under packet schedule B

$N^{G,K}_{S}$ - the random variable that represents the number of packets transmitted until succesfull reception of the *GK* packets in the file under scheduling *S*. Where *S* is one of: *O* (the optimal packet schedule), *C* or *B*.

$I^{G,g}_{Si}$- Indicator. It is 1 if the packet in time slot i under schedule S is the one from group g. It is 0 otherwise.

$F^{G,g}_{St}$- Number of times the packet from group g is sent at or before time slot t under packet schedule S.

i - group ID.

j,h - indexes.

m,n - time slot in group schedule.

PP - Desired success probability.

c* - estimated packets per packet

c** - estimated heuristically corrected packets per packet

Chapter 4

ipd - inter packet delay

j - multicast channel index

q - time slot number within multicast channel

R_j - rate of channel j.

B - Base rate.

l - subscription level

K - Number of data packets in FEC group.

N - Number of coded packets in FEC group.

p - packet index (0..N-1).

G - Number of FEC groups in File.

g - Group index (0..G-1).

J,W - $G=W2^J$ (where W is not an even number).

S - File Size in packets (S=GK).

s - slot number.

t - mini-slot number.

z - starting slot.

g^* - $g/(2^{J-l}W)$

r(g*) - non integer part of g*

M - $N=2^M$

ss - superslot number.

tt - mini-superslot number.

zz - starting superslot

gg - packet index within mini-superslot

Chapter 5

j.t - sampled channel ID.

q - time slot number within multicast channel

K - Number of data packets in FEC group.

N - Number of coded packets in FEC group.

p - packet index (0..N-1).

G - Number of FEC groups in File.

g - Group index (0..G-1).

S - File Size in packets (S=GK).

i,j - channel indexes

Chapter 6

K - Number of data packets in FEC group.

N - Number of coded packets in FEC group.

G - Number of FEC groups in File.

CHAPTER 1

Overview

Problem Statement

Given a large file located in a computer system herein after called a server and a very large number of client computers that desire to download the file and are connected to the server by means of a global interconnection network, we desire to make the file available to all receivers in minimum time and with the least consumption of network resources. We are interested in the case in which the receivers are arbitrarily dispersed within the network topology, have different and widely varying connection rates to the server, and may want to initiate the transfer at different points in time.

A trivial approach to the described problem would be to use some known point-to-point reliable data transfer protocol (such as FTP) between the sender and each one of the receivers. Every receiver could start the transfer at its desired time. The problem with this simple approach is the quick overloading of the sever connection with different flows that contain the exact same data. As the number of receivers grows, the results of such a solution would be devastating in terms of consumed network resources and would have unacceptable performance from a client's perspective. We wish to address this scalability problem and find a solution that is suitable for very large groups of receivers.

A proposed approach is to multicast the transferred data. In multicast communications, data is transferred from the source using a tree which spans all the receiver nodes. In this way, the replication of identical data over the same link is avoided thus drastically reducing the consumed network resources. The replication avoidance has the primary benefit of making the scheme scalable to a large number of receivers. This is

because of the nature of multicast transmissions in which link saturation (as a result of simultaneous transmissions) is eliminated.

In order to cope with the fact that individual receivers may want to initiate the file transfer at different times, the file could be repeatedly multicasted in a cyclic manner by the server. In other words, once finished with the first transfer, the server would start with it all over again and so forth for as long as there are receivers which have not yet finished to receive the data. Considering the bulk nature of the transferred data, in-order transfer is not necessary. Therefore every receiver could join the multicast transfer at any desired point in time and stay tuned for a whole cycle in order to receive the whole file.

An issue that would yet remain to be addressed is that of packet loss handling. Data packets are not guaranteed to arrive at its desired destinations in a best effort network such as the Internet. Individual receivers would experience different packet losses which would keep them from completing the reception even after having being tuned to one whole multicast cycle. By using packet identifiers, this situation could be regarded as that of an erasure channel. Erasure channels are characterized by the fact that errors happen in the form of packet losses which can be positively detected and identified by the receiving side.

Traditional reliable point to point data transfer schemes use feedback from receiver to sender for loss handling. Missed or corrupted packets are somehow reported to the server and retransmitted. In multicast communications reliability is a much more complicated issue. Several works have been done in the last few years on this regard which propose all kind of hierarchical and local recovery techniques. Every one of those schemes assume a feedback communication channel from receivers up the tree to the sender. As scalability is one of our major goals, we want to absolutely refrain from any from of feedback and hence these techniques will not be used. A scheme with no feedback requirements makes loss handling absolutely not influenced by the number of receivers and so completely scalable. Such a solution has the additional benefit of being also applicable to networks where feedback channels are virtually nonexistent such as wireless or satellite based.

Up to this point, we have not taken into account the heterogeneous characteristics of the receivers. Different connection rates, locations in the network with respect to the server and processing capabilities such as local storage access rate, result in different attainable effective rates for each of the clients. Moreover, other traffic in the network further complicates the situation by introducing dynamics into the problem. It is clear then that multicast transmission of the data at any single specific rate would not be a good solution for a vast number of heterogeneous receivers. Those with higher capabilities could have gotten the file faster while those with lower attainable rates would have caused excessive packet dropping either by themselves or by intermediate nodes.

Dropping as a result of uncontrolled packet flow is something we would very much desire to avoid in an environment like the Internet where congestion control can be a determinant factor of overall performance. We would not want our protocol to contaminate the network with packets that are bound to be dropped. The goal is then to devise a scheme that is friendly to other protocols that coexist in the network.

A first conceivable solution to the mentioned problem would be to simultaneously transfer the same file using different multicast channels at different rates. In this way, every receiver would subscribe to the channel corresponding to its highest attainable rate. The clear problem with such an approach is that network resources would not be optimally utilized. Let us consider two receivers topologically located at nearby places in the network but with different connection rates in their respective final hops. Because of the different rates, the two receivers would subscribe to different multicast channels resulting in extra data flowing through the common parts of their path.

An alternative is to use a receiver driven approach where the server sends the file using multiple channels and clients individually tune their reception rate through subscription to one or more of these channels simultaneously. This involves a sophisticated packet schedule at the server which is one of the main contributions of this work. The multicast group membership protocols, implemented at the distribution tree nodes, are responsible for the subscription and un-subscription mechanisms. Such dynamic membership mechanisms react to subscription changes so that data is no longer forwarded along the edges of the distribution tree if there are no subscribed clients downstream anymore. This property together with the self control applied by each receiver through its subscription policy is what effectively implements the congestion control.

The rest of this chapter provides an overview of this work and describes its organization.

Simple Cyclic Multicast Distribution of Bulk Data

The simplest way to handle losses in the erasure channel model with no feedback described above would be for the receiver to stay tuned for additional cycles in order to get the still missing packets. In CHAPTER 2 on page 17, we provide a mathematical analysis for some models of such a basic scheme. We analyze performance measures for receiver delays as well as consumed network resources. We back our statistical models with simulations and develop an approximation for the performance measures that provides insight into the factors that determine the results of this simple approach. We compare the results of our simple approach to those of an ideal selective retransmission point to point protocol. We conclude that the file size has a big impact in the results of the proposed scheme as compared to those achieved by feedback based point-to-point protocols where file size has no effect.

Multicast with FEC for Bulk Data Distribution

Since our goal is to provide a scheme where the results from a receiver's perspective are comparable to those that would have been achieved through a point-to-point reliable protocol, in CHAPTER 3 on page 43 we further evolve the simple cyclic scheme to make it less affected by the size of the file. This involves the use of known forward error correction (FEC) techniques. In a networking framework, a *(K,N)* forward error correction scheme works as follows: *N* coded packets are generated from *K* data packets at the server. The *N*

coded packets are then transmitted through the network. The correct reception of any *K* out of the *N* transmitted coded packets along with their identity, allows a receiver to reconstruct the *K* original packets.

FEC has been applied to many communication systems. A widely used mode, not related to our case, is one in which FEC is introduced to effectively improve the loss rate characteristics of a transmission media. In such a scheme, *N* units of information are sent instead of *K* for every *K* units that have to be transmitted. The amount of redundancy (*N-K*) becomes a significant factor for those applications as it determines the overhead paid for the use of FEC. Our application of FEC is substantially different. We use the error correcting codes combined with a multicast cyclic transmission schedule to enable each client to take advantage of any packet received in order to make progress in the reception of the file. The server keeps cycling over the *N* packets anyway so to satisfy additional clients. For that reason, not only that a large *N* does not imply a larger overhead, the larger the *N* the better the attainable results for our application.

Assuming a file of size *S* [packets], a straightforward application of FEC to the bulk file distribution problem would ideally encode the whole file (*K=S*) so to generate a very large number *N* of coded packets. The coded packets could then be sent cyclically using multicast. Each receiver could start reception at any point in time and would need to successfully receive any *S* packets out of those that were transmitted during its reception time in order to be able to reconstruct the original file[1]. From a client perspective, this methodology is ideal. The achieved results are the same as those of an equivalent selective-retransmission-based point-to-point reliable protocol. The reason for this is quite obvious. Every single packet successfully received counts towards completion of the file reception. This is equivalent to the case in which by explicitly requesting retransmission of those packets that were lost a unicast reliable protocol achieves its results. The use of FEC enables the multicast receiver to take advantage of any received packet where in the previous simple cyclic scheme in order to recover a packet that was lost the receiver had to wait for that specific packet to be transmitted again. This last fact is what made the simple cyclic scheme so affected by the size of the file.

The trivial application of FEC as presented above is the ideal solution to the bulk file distribution problem. Network resources are optimally utilized through the use of multicast while at the same time each receiver experiences the best possible performance results. However, practical implementations of publicly available forward error correction coding and decoding schemes have shown to be of non-linearly increasing time complexity with the growth of *K*. We want to provide a protocol that works with very large files and therefore we are limited to relatively moderate values of *K* as compared to those that would result from setting it to the total number of packets on the file as suggested above[2]. We therefore choose a computationaly efficient *K* and divide the file into *G* groups of *K* packets each (where *G=S/K*). Forward error correction is then applied to each one of the *G* groups to achieve *N* coded packets per group. The resulting coded packets are

1. provided *N* is large enough (in comparison to *K*) so that each receiver manages to successfully receive *K* valid packets before the server cycles over the same coded packets again

then transmitted using multicast and a group interleaving packet schedule that we prove to be optimal from every receiver's standpoint for any given set of parameters *K* and *N*.

We include a mathematical analysis of the performance of this protocol along with simulation results that back our statistical model. We compare the results to those of the previous simple cyclic scheme and show the effect of the FEC group size in the performance of the proposed scheme. We repeat then the comparison against the ideal selective retransmission protocol and show the benefits achieved through the use of FEC groups. We show that the results for a wide relevant range of setup parameters are comparable to those of the ideal point-to-point reliable scheme thus achieving our desired goal.

We also develop a bound for the mathematical expression that models the behavior of the presented protocol. We further simplify this bound to achieve an approximation that behaves very closely to the calculated expression and provides meaningful insights about the impact of the different parameters on the performance results.

Multi-rate Distribution to Heterogeneous Clients

In order to take full advantage of the benefits of multicast transmission in the presented heterogeneous clients scenario, we need a scheme where slower receivers take advantage of part of the data that goes to the faster ones but never add to the load of the links. In CHAPTER 4 on page 83, we show that in such a case, the network utilization is optimal in the sense that for every link in the transmission tree there is at least one client downstream that is receiving all the data that flows through the link. We call this situation **optimal network utilization** since the load on each link is never greater than what it would have been had **only** the fastest receiver downstream be downloading the file using a point-to-point reliable transport protocol.

In order to achieve the described optimal situation, we have pursued the approach of aggregated (or cumulative) channels. The server multicasts the data using different channels at different rates and receivers subscribe to one or more channels subject to the following restriction: All receivers subscribe to channel 0 in which data is sent at a base rate. Receivers with higher reception capabilities subscribe to additional channels in their numbered order. In other words, in order for a receiver to subscribe to channel j it has to subscribe to channels 0..j-1. We prove that this subscription policy results in optimal network utilization as defined above.

We then consider the combined packet stream received by each receiver under any subscription rate and propose a packet schedule mechanism and channel rate assignment scheme that results in optimal attainable

2. Some new FEC codes have been introduced lately that allow to use much larger values of K than those codes we refer to in this work. These codes are proprietary a fact that makes them far less attractive.In this work, we develop a schedule mechanism that combined with publicly available well known FEC codes, achieves results comparable to those that would be achieved with the new proprietary codes.

performance from the receiver standpoint. The main challenge is for the packet schedule to maintain the group interleaving property, that was proved optimal in a previous section, for all receivers simultaneously regardless of their subscription rate and their starting point in time. We have thus solved the multi-rate problem while achieving the same results[3] presented previously when the receiver rate heterogeneity was not taken into consideration.

Rate Resolution Enhancements for Layered Multicast

The multi-channel packet schedule scheme described above is based on cumulative subscription to channels whose rates are each double the rate of the previous one. The packets on channel 0 and 1 are sent at a base rate. From there on, every channel has double the rate of the previous one. We call this rate assignment: **exponential**. Clearly, with such a rate assignment and the cumulative subscription policy described above, each client is then allowed to select a rate equal to an integer power of 2 times the base rate. This coarse resolution in the rate selection may be too restraining for some practical cases. In particular, widely deployed congestion control schemes require a higher resolution in the rate selection in order to react as expected to network congestion.

In CHAPTER 5 on page 131 we further evolve our packet schedule scheme to provide more flexibility in the choice of the desired rate by each of the receivers. We explore two approaches to accomplish this goal. In the first one we achieve higher resolution rates by generating slower channels out of the schedule mechanism achieved before. We show that by sampling at the server our previously optimal channels, to achieve a new set of channels all of the same rate, we get a schedule scheme that behaves very closely to the optimal case and at the same time provides complete freedom in the rate selection.

The main drawback of the channel sampling technique is the increase in the number of multicast channels used. Our second approach to rate resolution enhancement uses a different technique that keeps the number of multicast channels to the same amount as in the original cumulative exponential scheme. For this, we relax the restriction of aggregated channels and evaluate a selective (non-cumulative) subscription approach. In other words, every receiver is free to select among the channels the desired combination that best matches its attainable reception rate. We test this selective approach using our proposed exponential channel packet schedule and show near-optimal results. As we prove, under a non-cumulative subscription scheme like this, clients may pick any combination of the transmitted channels and that may result in a non-optimal utilization of network resources. We show that for our suggested non-cumulative scheme the network over utilization is bounded by a small number and analyze the implications of this fact for practical cases.

3. optimal for any given FEC configuration (K,N) and comparable to what can be achieved with an ideal point-to-point reliable protocol.

We further evolve our exponential non-cumulative scheme into a combined approach where intermediate nodes in the distribution tree (such as routers) consolidate the channel requirements from downstream agents in order to achieve optimal network utilization. This is done at the expense of not satisfying in some cases the exact requirement of one or more of the downstream agents. We show a consolidation heuristic that guarantees to each client at least the same rate than the one that would have been achieved under the exponential cumulative scheme. Our heuristics are based on an attempt to satisfy the slower receivers as close as possible to their exact requirement. This is motivated by the observation that these receivers are the ones that will mostly benefit (in absolute terms) from an increase in their reception rate.

Unlike the previous simulations where results were analyzed from a single receiver standpoint, for this case we build a simulation infrastructure that enables us to mimic the effect of multiple receivers in a randomly generated multicast tree. We apply then our proposed heuristics to very large sets of receivers and show the improvement achieved over the exponential cumulative scheme.

Network Dynamics

We evaluate in CHAPTER 6 on page 167 the impact of network dynamics in the performance results attained by our proposed mechanisms. We simulate a scenario where the subscription rate for a client is changed during its reception time and measure the effect of this rate change in its perceived performance results. We show that the impact of this is very low and therefore claim that our proposed methods are applicable to the practical case where network dynamics affect the rate selection.

Summary

Our work is summarized in CHAPTER 7 on page 173

CHAPTER 2

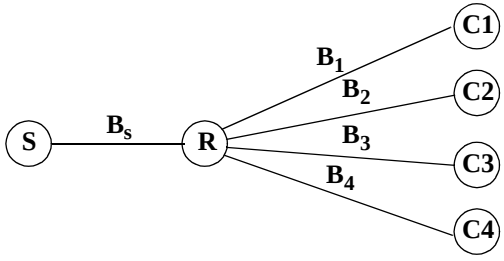
Simple Cyclic Multicast Distribution of Bulk Data

2.1 - Introduction

Distribution of Bulk Data

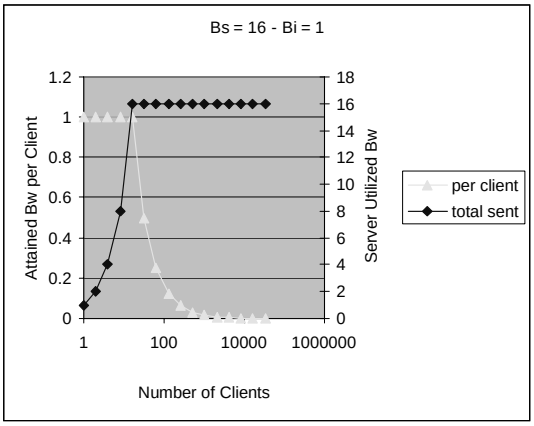
Let us consider a server connected to a large internetwork using a line of bandwidth B_s and several clients connected to the same network using lines of capacity B_i where B_s is greater than B_i for all clients. Let us assume the capacities of the mentioned links are the only bandwidth limiting factors in the communication between the server and the clients. Let us further assume that the probabilities for missing or corrupted packets are independent and equal to *q* for every packet transmitted.

FIGURE 2.1 - A Server and Several Clients



When a single client C_n downloads a file from the server (using for example the Unicast File Transfer Protocol), its perceived results are mandated by its effective transfer rate which is $B_n*(1-q)$. This is a reasonable result but is based on the assumption that there is always available bandwidth on the server's connection to the router. A serious problem arises when several clients attempt to download from the same server at the same time. The Unicast FTP approach does not scale very well. Very soon the server connection collapses and becomes the bandwidth limiting factor. Assuming fair distribution of bandwidth among all clients, the following figure shows the impact on the effective data rate for identical types of Unicast FTP customers as the number of transfers increases.

FIGURE 2.2 - Multiple clients downloading from the same server



If the files being transferred to the distinct clients are all different, there is not much that can be done as there is a real need for bandwidth augmentation on the server connection. However, there are many cases when all (or most of) the clients are trying to download the very same file. A good example of such a case is when a software company releases a new version of a very popular software or an update patch. In such cases people tend to connect and try to download those newly available files as soon as they become available causing the collapsing of the server connection. This phenomena is sometimes called "The midnight madness problem" because it usually happens at the late hours of the night when those releases are effectively made available to the public. Other cases with similar kind of problems are file distribution services for data such as weather, stocks, news and server mirrors updates.

Using Multicast for Bulk Data Distribution

Multicast or point-to-multi-point communications are an attractive solution for the problem presented so far. In Multicast communications, data is transferred using a multicast routing tree that reaches all clients inter-

ested in the transmission. In this way, a single portion of the data flowing through a link in the multicast tree satisfies at the same time all receivers downstream that link in the tree.

Multicast communication has been and still is a subject of vast research. Several problems exist when using this kind of data transmission including among others, multicast group management [1] [2], multicast routing [3] [4] [5], reliable multicast protocols [7] [11] [8] [9] [10] [11] [13], congestion control for multicast [12] [14] [15] [16] [17], session naming and management and multicast security issues.

Some of this problems will be addressed in this work in the framework of bulk data distribution.

2.2 - Simple Multicast for Bulk Data Distribution

File Multicasting

As seen, in order to overcome the limitations of the server connection and to take advantage of the fact that several (possibly non synchronized) copies of the same information are flowing on the same link, a natural solution is to use Multicast. Clients will use special multicast file transfer programs and join scheduled sessions of the file transfer. This sessions could be attended by every client wishing to download the file at the specified time (or at any time after the previous multicast has begun). Users would have to wait maybe a few minutes for the starting of the next session but the advantage would be great as the number of users listening to the same multicast session would be the reducing factor in the usage of the outgoing communication link from the server. The timing and consequently the number of concurrent multicast session would be determined by the server communication capabilities.

Cyclic Multicast

The above method of scheduling multiple (possibly overlapping in time) multicast transfers of the same file is really not necessary for our problem. That solution is well suited for media streaming where the ordering of the received information is substantial. In our case, since every receiver needs the whole file anyway, we could use a single cyclic multicast. In other words, the server would send the file and start all over again as long as there is at least one client receiving the data. The clients could join at any stage and listen to the multicast transfer for a whole cycle which can begin at any time. No waiting for the start of the next session would be needed and in addition one single instance of the multicast would have to be transmitted consuming much less bandwidth at the server connection than before. With this approach we are back at our ideal transfer rate and we can support an unlimited amount of clients without any impact on the server's connection.

Packet Loss

A problem that still needs to be solved, for this solution to work, is that of packet loss handling. In a packet switching network packets are lost mainly for two reasons. The first is related to the properties of the transmission media and is often regarded as bit error rate. Most modern transmission technologies achieve very small bit error rates (in the order of 10^-10 or smaller) and therefore most of the packet losses experienced in practical cases are related to the second cause: congestion control. Widely deployed network fabric devices react to congestion by dropping packets. This is in turn interpreted by the endpoints as an implicit congestion notification which causes them to slow down their injection rate thus alleviating the congestion problems. This dropping behavior is the main reason for packet loss in cases relevant for our problem.

When using a unicast approach, an end to end reliable transport protocol is implemented between the server and each client. Transmission errors are handled by the standard selective retransmission mechanisms. A missing or corrupted packet is resent by the server upon request from the client that detected its loss. With the multicast approach, a reliable scheme is much harder to implement. Reliable Multicast has been a subject of research and no single widespread solution exists yet for this task. We will further elaborate on this important issue in the following chapters but we can say at this point that one of the main issues in Reliable Multicast is that of scaling of the solution to huge groups of receivers.

Handling Packet Loss in Cyclic Multicast

In our particular case, we could take advantage of the cyclic nature of our multicast. A viable trivial approach for error handling with no overhead whatsoever from the server point of view, would be for the client to stay tuned to the multicast and wait for the next cycle in order to receive missed or corrupted packets. This method is completely scalable in the number of users as it makes no use whatsoever of a feedback channel.

In a somewhat different context, the cyclic distribution of data is addressed in [19]. That work focuses on the multiplexing of multiple streams in the same distribution channel and does not analyze the effect of packet losses.

We analyze the impact of packet losses in the performance of this simple algorithm. We assume that losses are caused by either packets that never reached the destination or corrupted ones which are discarded by lower protocol layers and appear to the application as missed packets as well. We further assume in this analysis that losses are non correlated in time. We address the time correlated model afterwards.

Analysis for Cyclic Multicast

We aim to calculate the time a receiver has to stay tuned to the cyclic multicast in order to receive the G distinct packets that form the whole file. We will measure this time in packet units. Let q be the probability that a packet does not reach our particular receiver. And clearly p=1-q is the probability of successful reception of that packet.

Let N^1 be a random variable which represents the number of times that a certain packet was sent until it was finally successfully received by our particular client. Clearly, the probability that a certain packet was received at or before cycle c is given by:

: **(EQ 1)**

$$P\left(N^1 \le c\right) = 1 - P\left(N^1 > c\right) = 1 - q^c$$

This is independent for every packet so if we denote with N^G the random variable that represents the number of cycles needed so that the G packets are successfully received, then:

(EQ 2)

$$P\left(N^G \le c\right) = \left(1 - q^c\right)^G$$

Given a file composed of G packets, the average number of cycles for receiving the whole file is then given by:

: **(EQ 3)**

$$E\left(N^G\right) = \sum_{c=1}^{\infty} c \cdot P\left(N^G = c\right) = \sum_{c=1}^{\infty} c \cdot \left[P\left(N^G > c-1\right) - P\left(N^G > c\right)\right] =$$

$$= \sum_{c=1}^{\infty} P\left(N^G > c-1\right) = \sum_{c=0}^{\infty} P\left(N^G > c\right) = \sum_{c=0}^{\infty} \left(1 - P\left(N^G \le c\right)\right)$$

(EQ 4)

$$E\left(N^{G}\right)=\sum_{c=0}^{\infty}\left(1-\left(1-q^{c}\right)^{G}\right)=\sum_{c=0}^{\infty}\left(1-\sum_{i=0}^{G}\left(\binom{G}{i}(-1)^{i}\left(q^{c}\right)^{i}\right)\right)=$$

$$=\sum_{c=0}^{\infty}\left(1-1-\sum_{i=1}^{G}\left(\binom{G}{i}(-1)^{i}\left(q^{c}\right)^{i}\right)\right)=\sum_{c=0}^{\infty}\left(-\sum_{i=1}^{G}\left(\binom{G}{i}(-1)^{i}\left(q^{c}\right)^{i}\right)\right)=$$

$$=-\sum_{i=1}^{G}\left(\binom{G}{i}(-1)^{i}\sum_{c=0}^{\infty}\left(\left(q^{i}\right)^{c}\right)\right)=-\sum_{i=1}^{G}\left(\binom{G}{i}(-1)^{i}\frac{1}{1-q^{i}}\right)$$

Optimality of the Simple Cyclic Multicast

The scheme proposed above involves the server continuously looping over the file packets. We prove that in the absence of any form of error correction codes, when only the original data packets are being transmitted, this cyclic looping is optimal in terms of the average number of packets that need to be transmitted for a receiver to successfully complete the reception of the whole file.

For brevity, the proof is given in “Optimality of the Simple Cyclic Schedule for the Whole Cycle Model” in Appendix A (page 183).

The proof shows that, for the packet schedule to be optimal, the server has to send every packet once before sending a packet for the second time and so on. The packet order within the cycle could be any. However, in order to satisfy the further constraint of clients starting reception at any point time we need to restrict the server to use the same packet order in all cycles. With this no matter at which point in time a client starts, every *G* transmitted consecutive packets are all different.

A Better Model for the Cyclic Multicast

The result attained so far is in fact worse than what will be experienced in a practical case. The reason for this is that the model above assumes a client will keep receiving during the whole cycle even after successfully receiving its last missing packet[1]. This difference can be significant especially when files are large. A more real prediction follows. We calculate the probability that the transfer is successfully completed after receiving packet *g* in cycle *c* assuming our same packet scheduling scheme is used.

1. A common mistake is to assume the difference between these two approaches to be half a cycle. This would have been true if before the last cycle there is only one packet missing which is not necessarily the case.

We define our random variable N^G to be now the number of packets (as opposite to the number of cycles) transmitted before successful reception of the whole file. We calculate then the probability of success after receiving exactly $cG+g+1$ packets as:

(EQ 5)

$$P\left(N^G \le cG+g+1\right) \equiv P_{c,g}^G = \underbrace{p \cdot q^c}_{\substack{\text{packet } g \text{ is} \\ \text{succesfull} \\ \text{after } c+1 \text{ trials}}} \cdot \underbrace{\left(1-q^c\right)^{G-(g+1)}}_{\substack{P(N_{G-(g+1)} \le c) \text{ the probability} \\ \text{that } G-(g+1) \text{ (the packets after} \\ \text{packet } g \text{ in every cycle) already} \\ \text{succeeded in the previous } c \text{ cycles}}} \cdot \overbrace{\left(1-q^{c+1}\right)^g}^{\substack{P(N_g \le c+1) \text{ the probability} \\ \text{that } g \text{ (the packets before} \\ \text{packet } g \text{ in every cycle) already} \\ \text{succeeded in the previous } c \text{ cycles} \\ \text{or the current one}}} \qquad \begin{array}{l} c = 0 \ldots \infty \\ g = 0 \ldots G-1 \end{array}$$

The average number of packets that will be sent from the moment a client joins the multicast transfer until he successfully receives all the packets is then given by:

(EQ 6)

$$E\left(N^G\right) = \sum_{c=0}^{\infty} \sum_{g=0}^{G-1} (c \cdot G + g + 1) \cdot P_{c,g}^G$$

Proof of Optimality for the Evolved Model

We prove that for the evolved model, where a client completes its reception whenever it receives the last packet that was missing, the simple cyclic packet schedule is still the one that results in minimum average reception time from the perspective of any client and regardless of its starting time.

The proof is a special case (K=1) of a more general one presented in “Proof of Optimality for the Group Interleaving Schedule” on page 52. We also present in “Optimality of the Simple Cyclic Schedule for the Partial Cycle Model” in Appendix A (page 185), a simplified version of the proof, that suits the special case in question.

Charts for Cyclic Multicast

In the following charts, we can see the calculated average normalized reception time for different error probabilities q and packets per file G. The averages were normalized by dividing them by G in order to make them comparable.

Figure 2.3 on page 24 and Figure 2.4 on page 24 compare the results of using the whole cycle model versus the partial cycle model as a function of the packet loss rate and file size respectively. As predicted, the partial cycle model achieves better results (closer to reality anyway) and the difference is not constant.

FIGURE 2.3 - Cyclic Multicast - Whole cycle model vs. Partial cycle model

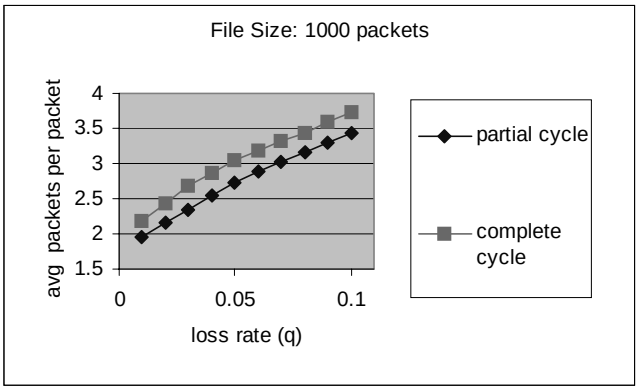


FIGURE 2.4 - Cyclic Multicast - Whole cycle model vs. Partial cycle model

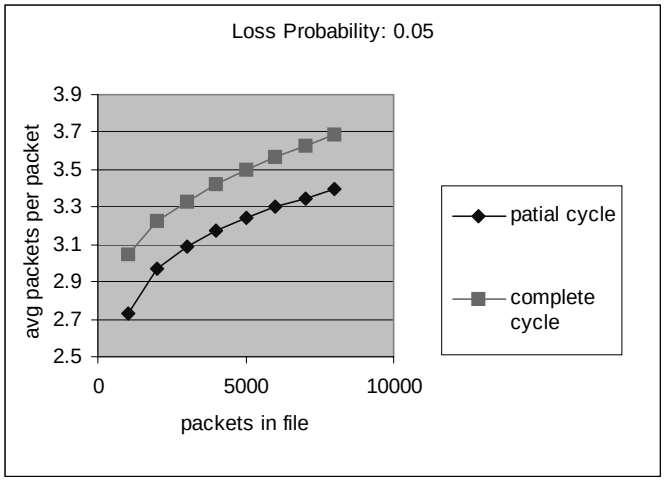


Figure 2.5 on page 25 shows the results using the partial cycle model as a function of the file size for different packet loss rates.

FIGURE 2.5 - Simple Cyclic Multicast - Avg as a function of File Size

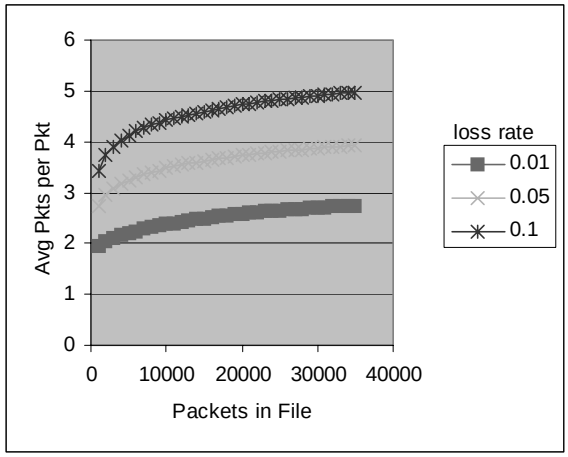
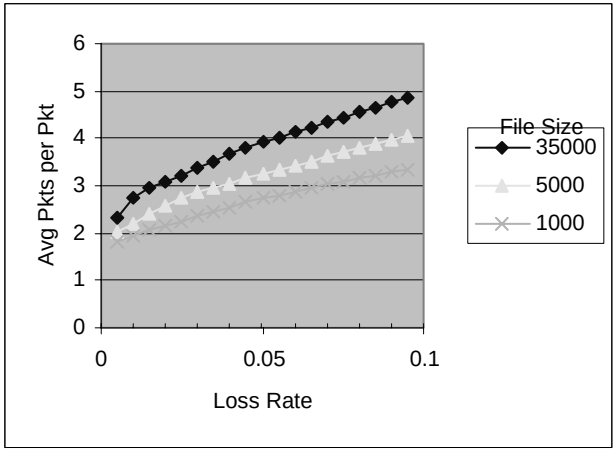


Figure 2.6 on page 25 shows the average number of cycles needed to receive the whole file using the partial cycle model as a function of the packet loss rate.

FIGURE 2.6 - Simple Cyclic Multicast - Avg as a function of Packet Loss



We can clearly see the price paid for errors. Even very low error probabilities render a very high price to pay. The problem is with the whole cycle that has to be awaited in order to recover a missed packet. Measuring the price paid in absolute time (not normalized) would have further emphasized the problem with large files.

A Simple Approximation for the Simple Cyclic Multicast

A similar approach for evaluating the performance of this simple cyclic scheme is to look at the amount of cycles needed to guarantee the complete file reception with a probability larger than P (for P close to 1). Looking at this performance measure is in some cases even more relevant than looking at the average.

As far as the scheduling is concerned, we have shown optimality of the average by proving optimality for each component in the average weighted sum. Therefore our proof applies to this case as well.

We have seen that:

(EQ 7)

$$P\left(N^G \le c\right) = \left(1-q^c\right)^G = \left(e^{\ln\left(1-q^c\right)}\right)^G \underbrace{\approx}_{q^c \approx 0} \left(e^{-q^c}\right)^G = e^{-q^c G}$$

Therefore:

(EQ 8)

$$-\ln\left(P\left(N^G \le c\right)\right) = -q^c G$$

We want to find *c* for *P* close to 1 so using:

(EQ 9)

$$-\ln\left(P\right) \approx \left(1-P\right)$$

We get:

(EQ 10)

$$\frac{1-P}{G} = q^c \rightarrow c = \frac{\ln\left(\frac{1-P}{G}\right)}{\ln q} = \frac{\ln(1-P)-\ln(G)}{\ln(q)} = \frac{\ln(1-P)}{\ln(q)} + \frac{\ln(G)}{-\ln(q)}$$

The expression above clearly matches our simulated and calculated results as shown in the following charts.

FIGURE 2.7 - Cyclic Multicast - Approximation as a function of File Size

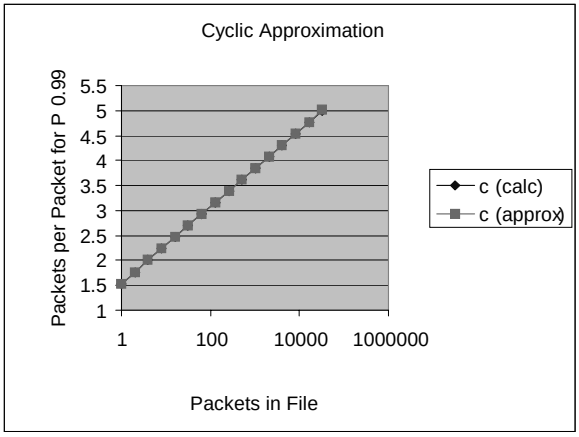
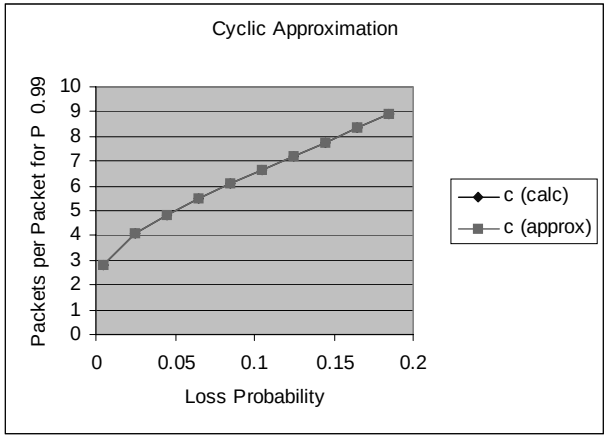


FIGURE 2.8 - Cyclic Multicast -Approximation as a function of Loss Rate



Analysis for ARQ Transport Protocols

We have calculated so far the performance implications from a client perspective when receiving a file using the suggested simple cyclic multicast scheme. In order to compare these results, we calculate now the equivalent measurements for a unicast transmission with same error parameter and file size.

Packet loss in unicast reliable communications is commonly handled using ARQ schemes. Every Receiver communicates its packet losses to the sender by means of a feedback mechanism and this in turn selectively retransmits those packets that were lost. For bandwidth optimization reasons there is usually more than one packet outstanding (sent but not yet acknowledged) in a unicast transmission but for the purposes of packet loss handling the protocol is equivalent to the following simplified description: The sender keeps sending each packet until it is successfully received at the destination.

With this model in mind, every packet successfully received counts toward successful completion. This is what makes the big difference when compared to the cyclic multicast analyzed above. Because of the same reason, unicast is to this extent not influenced by time correlation among losses. Clearly the only thing that matters in this simplified analysis is the error rate.

We start by calculating the probability of not receiving a single packet after *c* attempts:

(EQ 11)

$$P(U > c) = q^c$$

The average number of times a certain packet has to be sent in order to be successfully received is then:

(EQ 12)

$$E(U) = \sum_{c=1}^{\infty} c \cdot P(U = c) = \sum_{c=1}^{\infty} c \cdot [P(U > c-1) - P(U > c)] =$$

$$= \sum_{c=1}^{\infty} P(U > c-1) = \sum_{c=0}^{\infty} P(U > c) = \sum_{c=0}^{\infty} q^c = 1/p$$

Since for every subsequent packet its transmission will start when the previous was successfully received then the average total time (in packet units) for the transmission of *G* packets is given by:

(EQ 13)

$$E(U^G) = G \cdot E(U) = G/p$$

Comparison Charts Unicast vs. Simple Cyclic Multicast

Figure 2.9 on page 29 and Figure 2.10 on page 30 compare the results for simple cyclic multicast and unicast as a function of file size and loss probability respectively.

FIGURE 2.9 - Cyclic vs. Unicast - Avg Cycles as a function of file size.

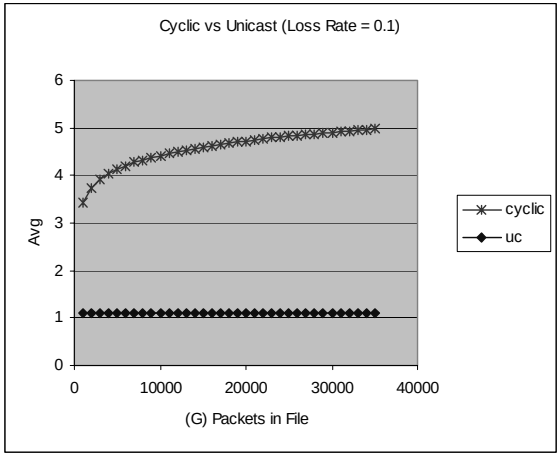
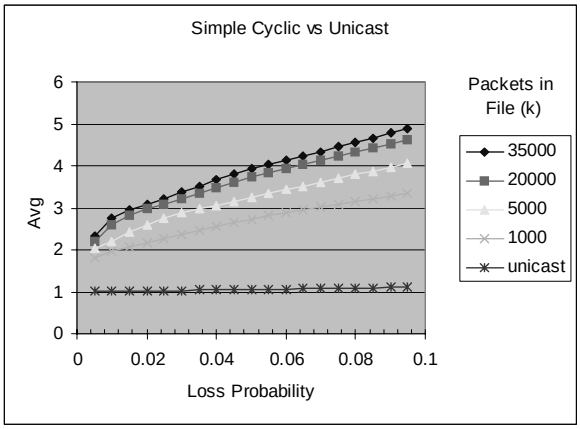


Figure 2.9 on page 29 shows the results as a function of the file size. As expected, with point-point reliable communications the file size has no effect on the normalized performance measure perceived by the receivers. The simple cyclic multicast shows the log behavior that we have seen in Figure , “A Simple Approximation for the Simple Cyclic Multicast,” on page 26.

Figure 2.10 on page 30 shows the same performance measure but this time as a function of the average loss rate. The loss rate impact on unicast is clearly low. The simple cyclic multicast is much more affected by the loss rate especially when the file is very large. This result is not surprising, lost packets have a huge impact on the no-feedback simple multicast scheme since the receiver needs to wait for a whole file transmission cycle in order to have the chance of recovering a lost packet.

FIGURE 2.10 - Cyclic vs. Unicast - Avg Cycles as a function of Loss Probability



In a future chapter we address this limitations of simple cyclic multicast. We will show reliable multicast schemes that obtain results comparable to those of unicast through the use of error correction codes.

An Approximation for Unicast

We already stated that averages can sometimes be misleading. As an alternative we estimated how many packets will a receiver see before we can say that it has successfully received the complete file with a probability higher than *P* (for *P* close to 1). We follow the same approach here for unicast.

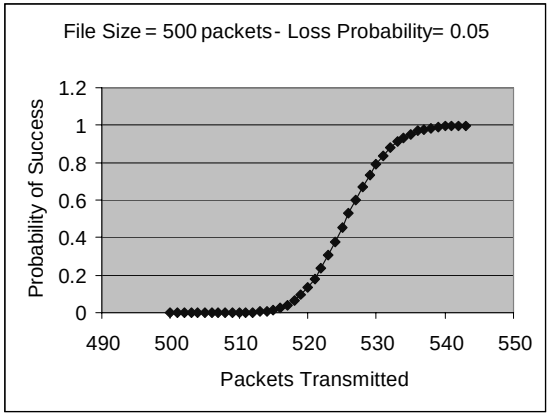
For this we need the probability of success at or before *c* packets which is given by:

(EQ 14)

$$P\left(U^G \le c\right) = \sum_{i=G}^{c} \binom{c}{i} (1-q)^i q^{c-i}$$

Figure 2.11 on page 31 shows a calculated plot of this function.

FIGURE 2.11 - Probability of Success after c packets



Let us define a random variable X which is the number of successful packet receptions out of c packets that were sent. The density function for X is then:

(EQ 15)

$$f_{X^c}(i) = P(X^c = i) = \binom{c}{i}(1-q)^i q^{c-i}$$

and its distribution is given by:

(EQ 16)

$$F_{X^c}(j) = \sum_{i=0}^{j} f_{X^c}(i) = \sum_{i=0}^{j} \binom{c}{i}(1-q)^i q^{c-i}$$

As expected,

(EQ 17)

$$F_{X^c}(c) = \sum_{i=0}^{c} f_{X^c}(i) = \sum_{i=0}^{c} \binom{c}{i}(1-q)^i q^{c-i} = ((1-q)+q)^c = 1$$

From here:

(EQ 18)

$$P\left(U^G \le c\right) = F_{X^c}(c) - F_{X^c}(G-1) = 1 - F_{X^c}(G-1)$$

Since for our practical cases (very large *G*):

(EQ 19)

$$c \cdot q \cdot (1-q) > G \cdot q \cdot (1-q) >> 1$$

then we can apply the DeMoivre-Laplace Theorem to obtain:

(EQ 20)

$$P\left(U^G \le c\right) = 1 - F_{X^c}(G-1) = 1 - GaussDist\left(\frac{G-1-(1-q)\cdot c}{\sqrt{q\cdot(1-q)\cdot c}}\right)$$

Let us define *b* to be the value that which for a given desired success probability *P*, makes:

(EQ 21)

$$P = 1 - GaussDist(-b)$$

Then, given a desired *P* we need to find the *c* for which:

(EQ 22)

$$\frac{G-1-(1-q)\cdot c}{\sqrt{q\cdot(1-q)\cdot c}} = -b$$

Solving for c we get:

(EQ 23)

$$c = \frac{2b^2(1-q)q + 4(1-q)(G-1) + 2b\sqrt{b^2(1-q)^2q^2 + 4(1-q)^2q(G-1)}}{4(1-q)^2}$$

which for small error rates can be approximated as:

(EQ 24)

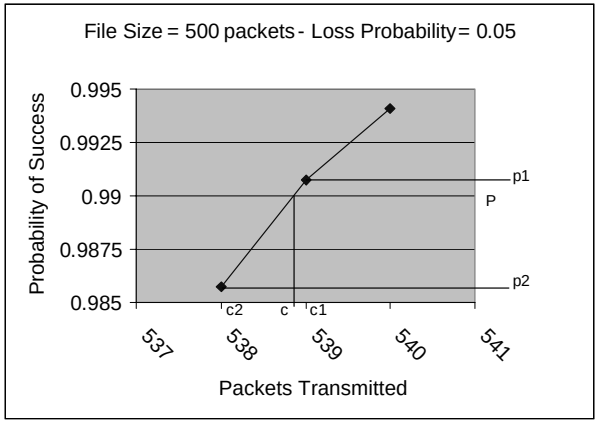
$$c \approx \frac{(G-1)+b\sqrt{q(G-1)}}{(1-q)}$$

As expected, the approximation tends to the avg when G tends to infinity.

In order to verify the accuracy of this approximation we plot in the following figures the calculated and approximated versions of *c* for *P*=0.99. The corresponding *b* for *P*=0.99 is roughly 2.35

In order to attain the calculated version of *c* for the following figures we linearly interpolate the curve *P(c)* in the vicinity of the desired *P* as shown in Figure 2.12 on page 33.

FIGURE 2.12 - Unicast Approximation - Linear Interpolation for Calculated c



c is therefore calculated as:

(EQ 25)

$$c = \frac{P - p2}{p1 - p2} + c2$$

Figure 2.13 on page 34 and Figure 2.14 on page 34 show the results of the approximation verification.

In Figure 2.13 on page 34 we can see the point at which the conditions for using the DeMoivre-Laplace theorem start to apply. Clearly the approximation is very good for file sizes longer than 100.

FIGURE 2.13 - Unicast Approximation- Cycles as function of file size.

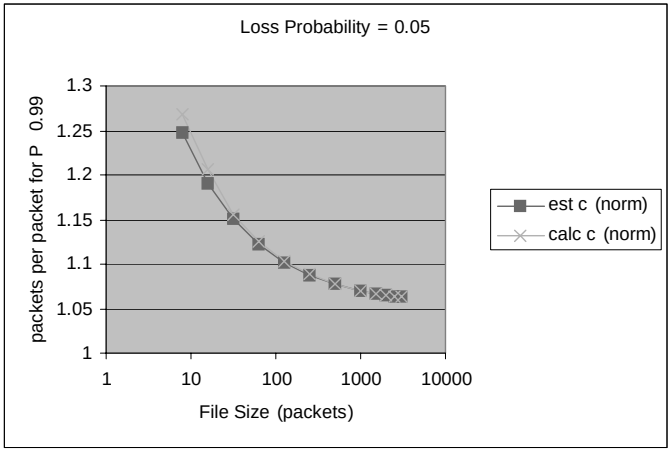


FIGURE 2.14 - Unicast Approximation - Cycles as function of loss probability.

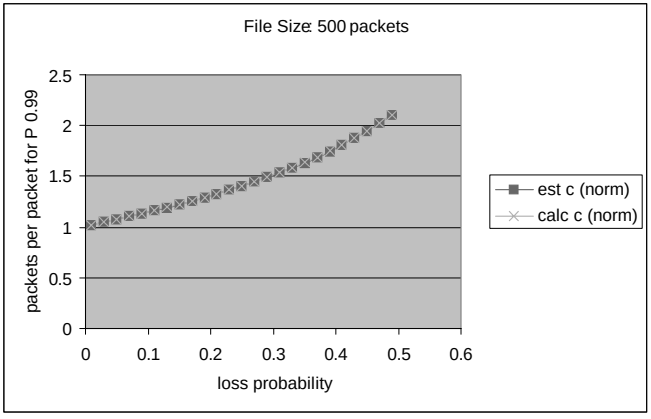
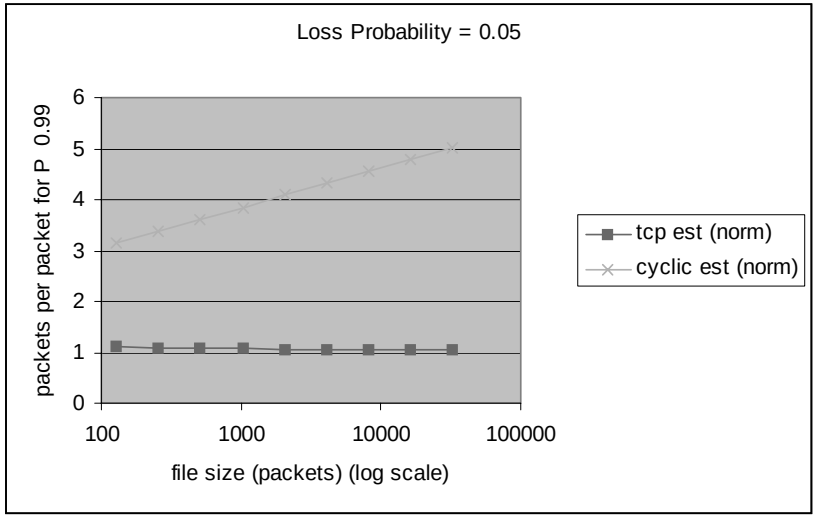


Figure 2.15, “Cyclic vs. Unicast - Cycles as function of File Size.,” on page 35 (note the log scale) compares the results for the simple cyclic multicast versus a unicast protocol. As we already concluded, the performance of the simple multicast from a receiver’s perspective is considerably lower than that of the analyzed ideal unicast. On the other hand, the simple cyclic multicast scheme allows the transmission to a huge number of non-synchronized receivers simultaneously with optimal utilization of network resources. This is of

paramount relevance since the performance presented from a receiver's perspective may not be ideal but it is definitely acceptable especially when for resource availability reasons the unicast alternative is simply not viable. Notwithstanding, in the following chapters we pursue the development of a solution that will bring the multicast results from a client's perspective closer to those of the unicast case.

FIGURE 2.15 - Cyclic vs. Unicast - Cycles as function of File Size.



2.3 - Time-Correlated Losses

Packet Losses in the Internet

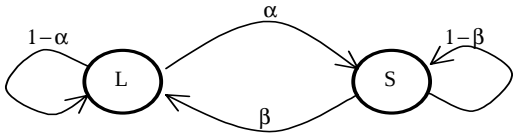
Throughout our analysis, we have assumed losses to be non correlated. We regarded the loss probability to be constant and independent from packet to packet. However, in the internet for example, most packet routers respond to congestion by discarding packets. In this kind of routers, when packet queue lengths get over some threshold, packets are dropped as a congestion relief mechanism. A variety of policies exists for packet dropping selection. A widely deployed class of heuristics results in the packet queue tail being dropped. This kind of behavior clearly results in correlation among packet losses[2].

The Receiver Two-state Model

In order to model this bursty loss scenario we will use a receiver two-state approach. We assume the receiver to be in either one of two states (S for success and L for loss). A receiver in the S state has successfully received the last packet. A receiver is in the L state when the last packet that should have arrived has been lost. Basically, this model allows the loss probability for the next packet to be different depending on whether the last packet was lost or not.

The state diagram for the two-state model with its transition probabilities is depicted in Figure 2.16 on page 36. By setting the adequate transition probabilities we can model with this scheme the desired error burst scenario.

FIGURE 2.16 - Receiver two-state model



Our parameters are the total loss probability and the average burst length. We now calculate the corresponding transition probabilities for the two-state model that suit our selected parameters.

Let *N* be a random variable denoting the number of consecutive losses in a burst. Our desired average burst is then *E(N)*. The total loss probability is equal to the probability of being in state *L* which will be denoted P_L.

Clearly:

2. Correlation in its pure meaning may be also periodic (which may be very bad for our algorithm if the period happens to be G) however that is not a real scenario. In real life loss correlation appears in the form of bursts.

(EQ 26)

$$P(N=n)=(1-\alpha)^{n-1}\cdot\alpha$$

The average number of losses in a burst is then:

(EQ 27)

$$E(N)=\sum_{n=1}^{\infty}n(1-\alpha)^{n-1}\cdot\alpha=\alpha\cdot\sum_{n=1}^{\infty}n(1-\alpha)^{n-1}=\alpha\cdot\frac{1}{\alpha^2}=\frac{1}{\alpha}$$

so:

(EQ 28)

$$\alpha=\frac{1}{E(N)}$$

The state transition equation is:

(EQ 29)

$$P_S=\alpha P_L+(1-\beta)P_S$$

Where:

(EQ 30)

$$P_L+P_S=1\Rightarrow P_L=1-P_S$$

Replacing P_L:

(EQ 31)

$$P_S=\alpha(1-P_S)+(1-\beta)P_S$$

We get:

(EQ 32)

$$\beta = \frac{\alpha(1-P_S)}{P_S} = \frac{(1-P_S)}{E(N)\cdot P_S} = \frac{P_L}{E(N)\cdot(1-P_L)}$$

Parameters for Simulations Using the Two-state Model

All our calculations so far assumed uncorrelated losses. In other words the loss probability for the next packet was the same independently of whether the last packet was lost or not. This case can be simulated using the two-state model by setting:

(EQ 33)

$$1-\alpha = \beta$$

In such a case:

(EQ 34)

$$\frac{E(N)-1}{E(N)} = \frac{P_L}{E(N)\cdot(1-P_L)} \Rightarrow (1-P_L)(E(N)-1) = P_L$$

(EQ 35)

$$E(N) = \frac{1}{(1-P_L)}$$

which looks familiar as expected. It is the result for the uncorrelated case where the average loss burst is:

(EQ 36)

$$\frac{1}{(1-q)}$$

When simulating the correlated case, we will choose:

(EQ 37)

$$E(N) > 1/(1-P_L)$$

in order to model the behavior of the internet routers mentioned above.

The case where:

(EQ 38)

$$E(N) < 1/(1-P_L)$$

is less interesting since it represents the case where the loss probability is higher when the last packet has been successfully received than when it has been lost.

Simulation Results

We simulated the loss correlated model and obtained the results shown in the following charts.

For this loss correlated model we use a simulator (as opposed to the calculations we used above for the non-correlated case). Figure 2.17 on page 40 shows the simulator results as compared to the calculated ones when operating the simulator with parameters that result in non-correlated model as described above. We clearly see that the results match perfectly.

FIGURE 2.17 - Verifying the two-state model simulator

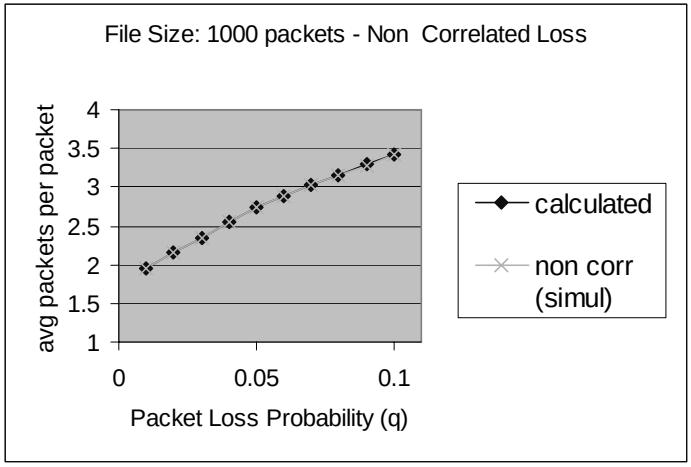


Figure 2.18 on page 40 and Figure 2.19 on page 41 show the impact of correlation on the results as a function of loss probability and file size respectively.

FIGURE 2.18 - Time Correlated model - Function of loss probability

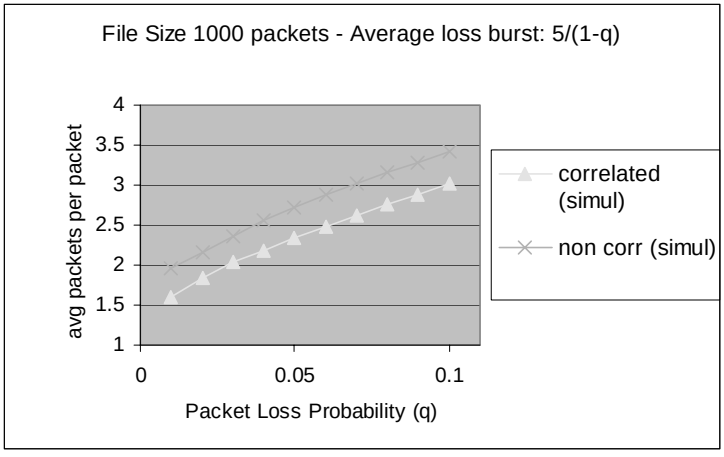
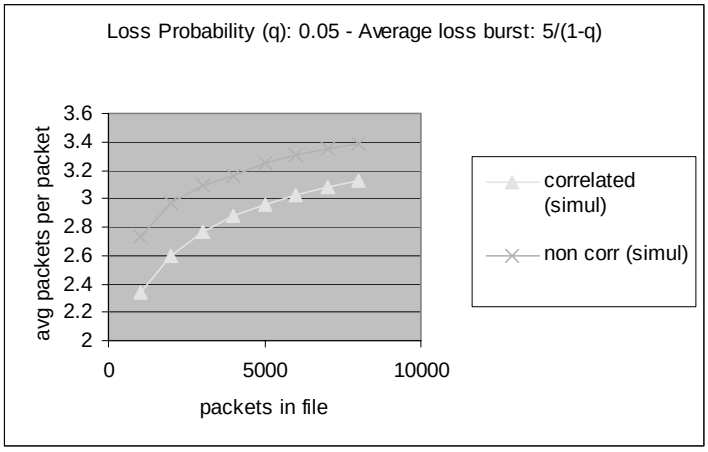


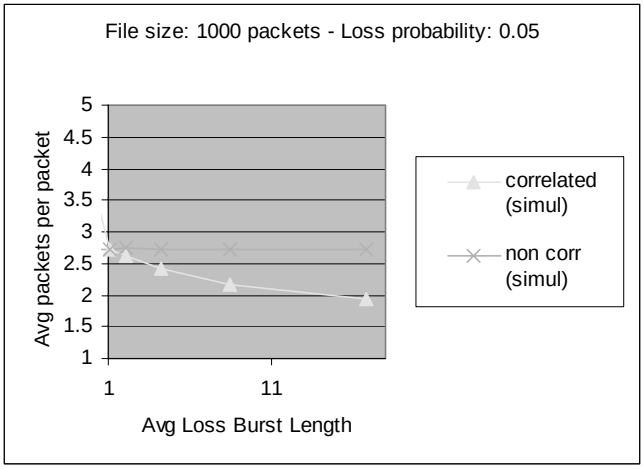
FIGURE 2.19 - Time Correlated model - Function of file size



We see that loss correlation improves the expected results. The reasons are related to the very nature of the cyclic scheduling. When losses are correlated, for any given average packet loss probability, the probability of a loss in the same packet in subsequent cycles is drastically reduced (since the loss rate was already "consumed" in packets adjacent to the first time of the one that was lost). A single retransmission then helps to receive several packet that were originally lost which brings the successful end of transmission closer.

Figure 2.20 on page 42 further emphasizes this property. In this figure we see the average cycles as a function of the loss burst length. We can see that the longer the average burst the greater the improvement. In fact the result approaches asymptotically the result for unicast when the burst length is increased. This can be explained intuitively by considering the extreme case where all packet losses happened in the same burst. In such a case, the complete file will be successfully received after reception of exactly the amount of packets in the burst. But since this amount of needed extra packets is equal to the loss probability multiplied by the total amount of packets sent we get the same results as for the unicast case.

FIGURE 2.20 - Time Correlated model - Function of loss burst length



Other Models for Loss Correlation

More sophisticated models for loss correlation could be devised using more than two states. We could further generalize our model of error burst by defining a state for every number of consecutive errors and also one for every number of consecutive successes. We could use a model with a non finite number of states with transition probabilities defined as a function of the state. We could also have a finite number of states with some states representing more than a certain number of errors or successes.

Conclusions

We have shown that for correlated cases which model the actual behavior of some practical scenarios the results are better than those when using a non-correlated model. The uncorrelated case will therefore be regarded as lower bound because positive correlation can only improve. We will keep using the non-correlated models for their simplicity whenever necessary with the knowledge they provide a worst case estimation.

CHAPTER 3

Multicast with FEC for Bulk Data Distribution

3.1 - Reliable Multicast Transport Protocols

Reliable data communication is generally characterized by the capability of the network protocol to provide application clients with a transmission framework where packets are received by their destination peers with neither losses nor duplications and in the same order as they were sent.

In a multicast scenario like the one we are evaluating, the problems of ordering and of duplicate packets can be relatively easily solved as they have no direct implication and no need of interaction with the network and the sender in particular. There are no scaling issues in this regard since every receiver of the multicast group can handle this task independently. A simple approach would be to include some sequence number in the packets and have a higher protocol layer reorder and eliminate duplicates accordingly.

On the other hand, the issue of packet losses is much harder to overcome. At first thought, some kind of interaction with the sender is required to request the retransmission of the missed packets. The following sections further elaborate on the different approaches in the search for solutions to this problem.

ARQ (Automatic Retransmission Request)

In a previous chapter we described a simplistic point to point ARQ based protocol for which a server keeps sending a packet to a client until the packet has been successfully received and then moves to the next packet. The server is explicitly notified by the client of packet successes and misses. Most of the point-to-

point reliable protocols use more evolved methods that take into account other considerations, however for the sake of comparison, this loss handling model resembles the real case in a quite faithful manner.

In a multicast framework sending explicit acknowledges is very problematic. When the number of receivers increases the single server has to deal with an augmenting number of response packets. Moreover it has to keep context information for each of its receivers to manage retransmissions. This scheme is therefore not applicable to large receiver sets.

NAK based schemes are somewhat better than ACKs for multicast. In a NAK based reliable multicast protocol, only receivers that have failed to receive a packet explicitly request a retransmission. Those that have succeeded do not send any feedback packets. This reduces the number of responses that the server has to deal with when the loss probability is low. However when the number of receivers increases beyond some point, the probability that a packet has been missed by one of the receivers increases and the server starts to get more and more responses. Moreover, loss is correlated among receivers in multicast communication. When one receiver misses a packet, it is very likely that most of its topologically close neighbors have missed the packet as well. In such a case, a single loss would result in a lot of negative feedback packets that will have to be processed by the server wasting precious processing time.

These problems are often referred to as the feedback implosion problem because of the extra load that huge feedback causes to the server. Hierarchical methods have been suggested to mitigate the feedback implosion problems, however there is yet no massively deployed solution that solves all the problems.

An interesting issue related to selective retransmissions in multicast is whether the server sends the repeated packet to the whole multicast group or just to those that explicitly requested it on a separate point-to-point channel. While the former scheme negatively impacts those that have already received the packet the latter uses much more bandwidth.

Forward Error Correction

As we have mentioned above, feedback based schemes have serious scalability problems when applied to multicast with very large receiver sets. We need an alternative approach that makes no use of a reverse channel. Apart from being scalable, a scheme like that could be used in highly asymmetric infrastructures (such as wireless networks) where little or no upload channel is available.

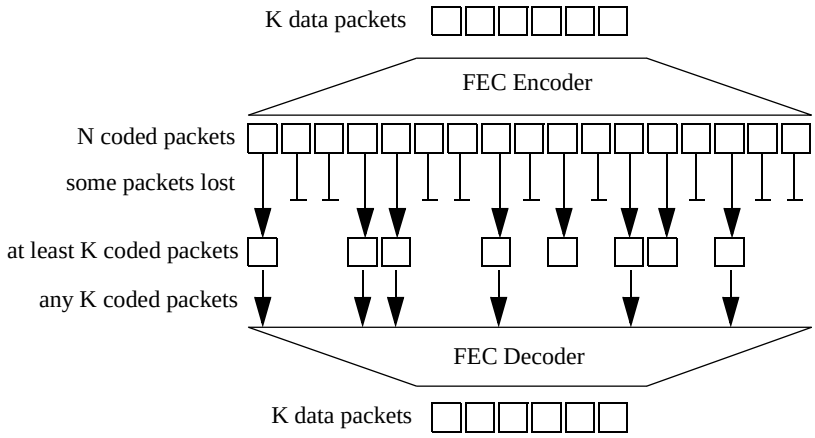
The basic idea behind all no-feedback approaches is to have the server do retransmissions in an open-loop kind of way. In a previous chapter we analyzed the simplest of such schemes. There, our server repeatedly transmitted all the packets of the file in a cyclic manner while each of the individual clients kept receiving until the whole file was successfully completed. We evaluated the performance of that simple mechanism to find out that the number of packets in the file had quite a big impact on the time that takes for each receiver to complete. This is an undesired property especially when comparing to an ideal selective retransmission scheme whose performance is not affected by the file size.

A more evolved method which as we will show deals with the file size effect, can be achieved through the use of Forward Error Correction. Forward error correction (FEC) is based on the presumption that some losses will occur. The sender anticipates the effect of losses by sending some kind of redundant information along with the packets. Receivers use this redundancy in order to reconstruct lost packets. The idea clearly resembles the concept of parity. Under FEC schemes, a specific packet that is successfully received, can be used to recover any one out of a group of lost packets thus effectively reducing the waiting time that a client experiences when a particular packet was lost.

The application of FEC to reliable multicast was investigated in [18] [20] [21] [22] [23] [24] and [25]. Among these, [18] [20] and [23], deal with multicast distribution of bulk data.

Lower layers of the protocol stack (link layer CRC, network layer fragment timeout/checksum, etc.) provide error detection and drop erroneous packets. By using a sequence number, the client can identity the received packets and detect those that were lost. Under these conditions the network can be regarded as an erasure channel. In this computer communication framework, we use the error correction capabilities of FEC to reconstruct the missing (or “erased”) packets and it is for this that FEC codes are sometimes called erasure resilient codes.

FIGURE 3.1 - Forward Error Correction Schematic Description



The general idea behind an erasure coding scheme with parameters (K,N) is depicted in Figure 3.1 on page 45. The sender divides the data stream into K packets. The K data packets are passed through an encoder that generates N>K packets. The N packets are then sent to the receivers. The encoding process is

built so that upon reception of any K+ε packets along with their identities, a receiver can reconstruct the K original packets that formed the original data stream. When ε is zero the encoding is called Maximal Distance Separable (MDS) and has the obviously desirable property of maximum efficiency.

The building of FEC encoders and decoders is beyond the scope of our work. We will use widely available FEC codes such as those described in [27] to base our analysis. A basic introduction to FEC principles of operations can be found in [27] and [28]. More comprehensive material is available in [26].

Remark: FEC has been applied in many communication systems. A widely used mode, is one in which FEC is introduced to effectively improve the loss rate characteristics of a transmission media. In such a scheme, *N* units of information are sent instead of *K* for every *K* units that have to be transmitted. The amount of redundancy (*N*-*K*) becomes a significant factor for those applications as it determines the overhead paid for the use of FEC. Our application of FEC is substantially different. We use the error correcting codes combined with a multicast cyclic transmission schedule to enable each client to take advantage of any packet received in order to make progress in the reception of the file. The server keeps cycling over the *N* packets anyway so to satisfy additional clients. For that reason, not only that a large *N* does not imply a larger overhead, the larger the *N* the better the attainable results for our application. Furthermore, if *N* is very large and *K* is the size of the file (since every receiver will keep receiving packets until successfully receiving any *K* packets) the case is clearly equivalent to what the best selective retransmission approach (i.e. reliable unicast) can provide and therefore optimal in terms of receiver perceived performance.

From a server perspective, the reason that makes FEC particularly effective in a multicast scenario is the fact that a single transmitted packet is useful to substitute a lost packet at multiple receivers simultaneously even when it comes to replace a different missed packet at each client. The advantages of FEC are exacerbated when the amount of clients grows to a very large number. The probability that each packet is missed by at least one receiver grows with the number of receivers and the fact that a single FEC packet can overcome multiple packet losses at different receivers makes FEC especially attractive.

The absolute benefit of FEC from a sender's perspective is somewhat reduced when the shared portion of the multicast tree increases. The reason for that is quite obvious. The more shared links in the tree, the higher the possibility that a single packet loss in some intermediate link will simultaneously affect a larger population of receivers. When such a thing happens, the retransmission of that particular packet would have solved the problem to every single one of them resulting in no apparent benefit in the usage of FEC. Notwithstanding, the use of FEC is never detrimental and for the above example to be applicable we would have needed some kind of feedback mechanism to report the identity of the lost packet (a thing we already concluded to be not desired).

3.2 - Using Erasure Codes with Multicast for Bulk Data Distribution

FEC Groups

We desire to transmit a very large file from a server to a very large number of receivers using multicast and FEC. Ideally, we would have desired to encode the entire file using FEC and start transmitting the encoded packets. Under such a scheme our clients would experience the same exact performance as an equivalent unicast client using feedback based packet retransmission. This follows from the fact that when the whole file is encoded using FEC as a block, every successfully received packet counts toward completion.

However, the encoding and decoding complexity of publicly available forward error correction codes grows non-linearly when the amount of packet increases. Consequently, for computability reasons, forward error correction is generally applied to data blocks of conservative size. Our goal is to find a solution applicable to files of huge size and therefore we will resort to file partitioning. Big files will be divided into blocks for which FEC codes are computable. [27] presents a software implementation of codes that can run in personal computers with very acceptable performance. Extrapolation of the results in [27] enable us to conclude that with nowadays computers, FEC codes are more than easily feasible for block sizes that range around the tens of packets.

It is worth noting though that the coding and decoding stages are not a critical part of our application, as they are not required in real time. Coding may be done before transmission starts; and decoding can be done independently to the packet reception process. Nevertheless we desire our mechanism to not be affected by code complexity since otherwise the decoding stage would take a relevant part in the performance assessment.

In this chapter, we deal with the packet schedule needed to attain optimal results from a receiver's perspective when the file is partitioned and encoded with FEC as suggested above. Alternatively, a very efficient class of erasure codes has been proposed in [28]. These codes are shown to be adequate for very large data blocks at the expense of some loss in the code efficiency ($\varepsilon>0$). In other words, they allow large blocks to be coded with no partitioning at the cost of some more than K code packets needed to reconstruct K data packets. It has been shown in [28] that in many cases the amount of extra packets needed is small as compared to the benefits attained. However, our solutions apply to files of huge sizes for which even these codes would need some file splitting and moreover, this novel FEC codes are proprietary and therefore it is very interesting to find alternative solutions that provide comparable results by using codes that are available in the public domain.

In this chapter we devise solutions that apply to cases where traditional, publicly available, maximal distance separable, error-correcting codes are used and therefore the block size is limited to a few tens of packets. We split our file into blocks of suitable size and propose a packet schedule scheme that achieves results

comparable to those of equivalent selective retransmission techniques from a receiver's perspective. Recall this was the main motivation for further developing the simple cyclic multicast scheme presented in the previous chapter. We desire to obtain a multicast mechanism (so to maintain the optimal network resources utilization), that is perceived from a client's perspective as one with comparable performance to that of reliable unicast.

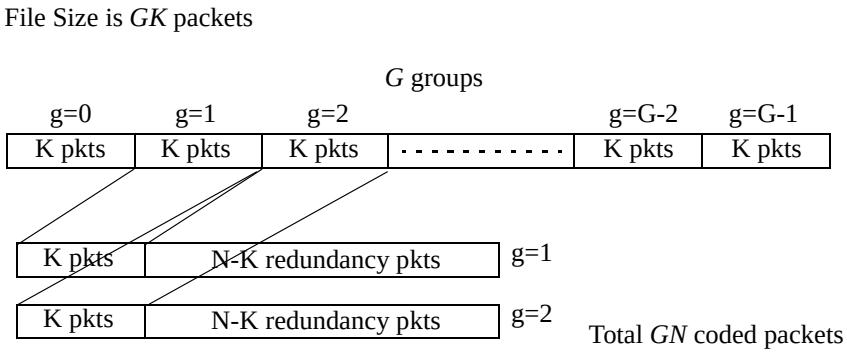
Let us then denote with *K* the number of packets that for computability reasons can be encoded using FEC. We call each of these blocks a group. We divide our file into G groups of *K* packets each and denote each of the groups with:

(EQ 1)

$$g = 0,1,\cdots G-1$$

Clearly *GK* is the size of the file.

FIGURE 3.2 - Dividing a File into groups of K packets for FEC



In order to successfully complete the reception of the file, each client needs any *K* different packets out of the *N* in every group. All packets are sent by the server using a specific schedule mechanism. Each client keeps receiving packets until it has successfully accumulated enough packets to complete the file reception.

Cyclic Multicast with FEC

The server has now a total of *GN* different packets obtained by encoding the *GK* packets of the file. We suggest the following packet schedule denoted **group interleaving** for the server to transmit the file.

Definition 3.1 - Group Interleaving Packet Schedule
The packet to be sent at time slot *i* belongs to the group given by

(EQ 2)

$$i \bmod G$$

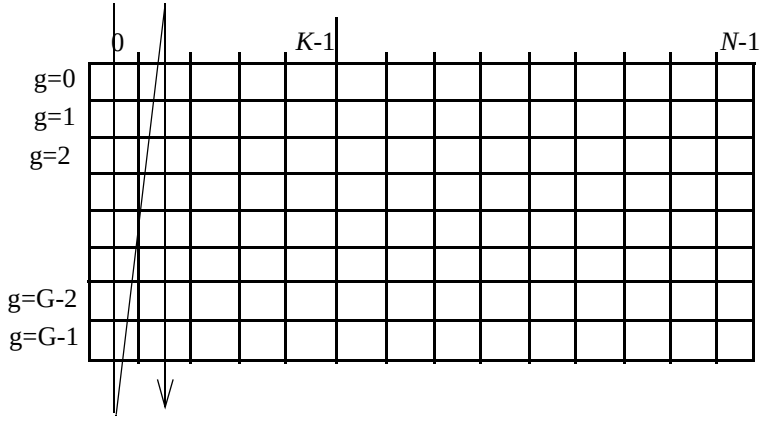
and the coded packet index within the selected group (to be sent at time slot *i*) is given by:

(EQ 3)

$$\left\lfloor i/G \right\rfloor \bmod N$$

In this context, we define a **cycle** to be the process of sending *G* packets, (using the **group interleaving** schedule) one per FEC group. Figure 3.3 on page 49 depicts the **group interleaving** schedule and shows 2 **cycles**. The transmission by the server is continuous so when the *GN* packets are exhausted the process starts over again.

FIGURE 3.3 - Group Interleaving Packet Schedule



We evaluate below the performance of this FEC based mechanism in terms of the average amount of packets that a single client has to receive in order to get the whole file when the **group interleaving** schedule is used

to transmit the packets. We show that the use of FEC results in a significant improvement in the performance perceived by the receivers as compared to the simple cyclic scheme of the previous chapter. In particular, we expect the results with *G* groups of *K* packets to be much better than those of the simple cyclic case since the number of packets in a **cycle** is reduced by a factor *K*. This reduction means that a receiver that is missing a packet has to wait much less time to have the chance of receiving it than before where the whole file was retransmitted in a cyclic manner.

We prove that given a FEC mechanism with specific parameters *K* and *N*, the best results from a receivers perspective are attained when the file is transmitted using the **group interleaving** schedule presented above.

From the server's perspective the benefits come from the fact that with FEC a single packet may be used by different clients to replace different packets that each of them has missed.

The **group interleaving** schedule was also presented in [20]. However we have found no reference in the literature that provides mathematical analysis of its performance neither proofs of its optimality from a receiver's perspective.

Performance Analysis of Multicast with FEC

Let us denote by X^c a random variable that represents the exact amount of packets received successfully by a client out of c sent packets; and with V_m^c, the probability that X^c is equal to m. If the loss probability for every packets is independent and equal to q, and p is 1-q, then:

(EQ 4)

$$V_m^c \equiv P\left(X^c = m\right) = \begin{cases} p^m q^{c-m} \dfrac{c!}{m! \cdot (c-m)!} & 0 \le m \le c \\ 0 & otherwise \end{cases}$$

Let us derive then the probability that the file was successfully received by one client after the packets from c complete cycles plus g+1 packets from the current cycle have been sent. For this we define $N^{G,K}$ to be a random variable representing the number of packets that were sent for a client to successfully receive the file. The probability that $N^{G,K}$ is equal to $cG+g+1$ packets is given by:

(EQ 5)

$$P_{c,g}^{G,K} \equiv P\left(N^{G,K} = cG+g+1\right) = p \cdot V_{k-1}^c \cdot \left(\sum_{j=k}^{c} V_j^c\right)^{G-(g+1)} \cdot \left(\sum_{j=k}^{c+1} V_j^{c+1}\right)^g \qquad c = 0..\infty \quad g = 0..G-1$$

and the average for $N^{G,K}$ is then:

(EQ 6)

$$E\left(N^{G,K}\right)=\sum_{c=0}^{\infty}\sum_{g=0}^{G-1}\left[(G\cdot c+g+1)\cdot P_{c,g}^{G,K}\right]$$

This analysis is made assuming we have an unlimited supply of coded packets per group (*N*). In practice, we will see that this assumption is adequate. FEC schemes with *N=8K* are pretty common. We will see that the results for P($N^{G,K}$>8GK) are negligible for virtually all relevant cases and thus the assumption is justified.

Special Case: *K*=1. As we shall see, larger values of *K* result in lower average transfer times but increase the complexity of the encoding/decoding. In particular, when *K*=1, *G* is equal to the file size in packets. In such a case, since there is a single packet per group, the group interleaving mechanism degenerates into a mere cyclic transmission of the file packets. Then:

(EQ 7)

$$P_{c,g}^{G,1}=p\cdot V_0^c\cdot\left(\sum_{j=1}^{c}V_j^c\right)^{G-(g+1)}\cdot\left(\sum_{j=1}^{c+1}V_j^{c+1}\right)^g=p\cdot q^c\cdot\left(\sum_{j=1}^{c}V_j^c\right)^{G-(g+1)}\cdot\left(\sum_{j=1}^{c+1}V_j^{c+1}\right)^g$$
$$c=0,1,\cdots\infty\quad g=0,1,\cdots G-1$$

which is the same result we achieved in the previous chapter for the simple cyclic multicast with no FEC as expected.

Special Case: *G*=1. Another extreme case is when we have a single FEC group for the whole file. From a per-receiver performance standpoint, the proposed algorithm under this condition has to obtain the same results as that of a selective retransmission point to point mechanism such as TCP. This is because under this scheme, since all packets belong to the same (and only group) each packet successfully received counts towards completion.

Since *G*=1, *K* is equal to the whole file size. Then we use:

(EQ 8)

$$P_{c,g}^{1,K}=p\cdot V_{K-1}^c\cdot\left(\sum_{j=K}^{c}V_j^c\right)^{1-(g+1)}\cdot\left(\sum_{j=K}^{c+1}V_j^{c+1}\right)^g\qquad c=0..\infty\quad g=0..1-1$$

but since *G*=1, *g* can only be 0 which results in:

(EQ 9)

$$P_{c,g}^{1,K} = p \cdot V_{K-1}^{c} = \begin{cases} p \cdot p^{K-1} q^{c-K+1} \dfrac{c!}{(K-1)! \cdot (c-K+1)!} & c \geq K-1 \\ 0 & otherwise \end{cases}$$

The average for (Eq. 9 on page 52) is *K/p* which is the same result as expected from the unicast analysis in the previous chapter. For the detailed mathematical derivation see “Average for G=1” in Appendix A (page 192).

Proof of Optimality for the Group Interleaving Schedule

In this section we prove that for any given *K*, the **group interleaving** schedule suggested above is optimal from a receiver’s perspective. We are assuming for the proof that *N* is infinite. As we see below, being *N* one order of magnitude greater than *K* for all relevant cases, this assumption is perfectly justified. Notwithstanding, the proof can be extended, using the same principle, to the finite *N* case.

Theorem 3.1 - Group Interleaving Optimality

Let us denote with *O* the **group interleaving** schedule defined in “Group Interleaving Packet Schedule” on page 49 and with O_i the group to which the packet sent in time slot *i* belongs under such a schedule.

(EQ 10)

$$O_i = i \bmod G \qquad i = 0,1,2,\cdots \infty$$

Given a file of size *S* packets and an implementable FEC code with parameter *K*. Assuming *G=S/K*, the schedule *O* results in the smallest average amount of packets needed to be transmitted for a client to receive the whole file.

Proof:

We prove by showing that for any packet schedule (denoted *C*) other than *O* (the one we claim to be optimal), there exists another packet schedule (denoted *B*) that gives better results. We construct this better schedule and shown that the construction method converges to *O*.

We denote with:

(EQ 11)

$$N_S^{G,K}$$

the random variable that represents the number of packets transmitted until successful reception of the *GK* packets in the file under schedule *S*. Where *S* is one of *O* (the optimal packet schedule), *C* or *B*.

We define *F* (below) to be number of packets of a specific group *g* that have been sent out of the total first *t* packets sent under schedule *S*.

(EQ 12)

$$I_{S_i}^{G,g} = \begin{Bmatrix} 1 & S_i = g \\ 0 & S_i \neq g \end{Bmatrix} \; g \in \{0,1,\cdots G-1\} \, i \in \{0,1,\cdots \infty\}$$

(EQ 13)

$$F_{S,t}^{G,g} = \sum_{i=0}^{t-1} I_{S_i}^{G,g} \;\; g \in \{0,1,\cdots G-1\} \, t \in \{0,1,\cdots \infty\}$$

We will show that for any schedule scheme *C* (candidate schedule) with:

(EQ 14)

$$C_i \in \{0,1,2,\cdots G-1\} \quad i = 0,1,2,\cdots \infty$$

different from *O*, there exists another schedule *B* (better schedule) with:

(EQ 15)

$$B_i \in \{0,1,2,\cdots G-1\} \quad i = 0,1,2,\cdots \infty$$

such that *B* results in a smaller average than *C*.:

(EQ 16)

$$\forall C \neq O \; \exists B \; / \; E\left(N_B^{G,K}\right) < E\left(N_C^{G,K}\right)$$

In other words, we will prove first that for any schedule (except the one we claim to be optimal) there exists another schedule that achieves better results.

Let us divide the schedule scheme *C* into cycles of *G* packets each. Since *C* is different from *O* there is at least one cycle of *G* packets for which in *C* there is at least more than one packet of the same group[1]. Let us pick the first such cycle.

(EQ 17)

$$C \neq O \Rightarrow \exists c,i,j,h \,/\, C_{G\cdot c+j} = C_{G\cdot c+h} = i \quad c \in \{0,1,2,3,\cdots\infty\} \;\; i,j,h \in \{0,1,2,\cdots G-1\}$$

Case 1: ($c<K$) The first cycle in which there is more than one instance of the same packet is before sending *GK* packets. Let us look now at the first point in the proposed schedule *C* at which for the first time *K* packets have been sent from all *G* groups. Let us denote the total number of packets sent until this point with *m*.

(EQ 18)

$$m = \min_{i \in \{0,1,2,\cdots\infty\}} / \forall g \in \{0,1,2,\cdots G-1\} \exists t \le i \,/\, F_{C,t}^{G,g} = K$$

Note that if *C* is such that *m* does not exist (i->infinite) then our suggested *O* is of course better than *C* since $E(N_C)$ is infinite. So *B* will chosen to be equal to *O* in such a case.

If *m* exists, our suggested improved schedule *B* will send the exact same packets as *C* after the first *m* packets. For the first *m* our suggested improvement will reorder the same packets so that the first *GK* packets are *K* out of each of the *G* groups (in any order) and the remaining *m-GK* are randomly ordered.

(EQ 19)

$$B_i = \begin{cases} |i|_G & 0 \le i < GK \\ \text{any order from the packets in } C \text{ not used in the first } GK. & GK \le i \le m \\ C_i & m < i \end{cases}$$

The probability of success at or before packet *i* is received for *i* greater than or equal to *m* is the same for both schemes. However for *C* the probability of success at or before packet *i* for *i* smaller than *m* is zero where for the suggested improved scheme it is greater than zero for all *i* greater than or equal than *GK*-1.

1. The order of the packets within each cycle is not relevant at this stage. We show later that in order for different clients to start reception at different points in time then the order of packets within all cycles has to be the same.

(EQ 20)

$$\begin{aligned} P\left(N_B^{G,K} \le i\right) &= P\left(N_C^{G,K} \le i\right) = 0 && 0 \le i < GK - 1 \\ P\left(N_B^{G,K} \le i\right) &> P\left(N_C^{G,K} \le i\right) = 0 && GK - 1 \le i < m \\ P\left(N_B^{G,K} \le i\right) &= P\left(N_C^{G,K} \le i\right) && m \le i \end{aligned}$$

Case 2: (c>=K) The first cycle in which there is more than one packet of the same group is not within the first K cycles. Let us pick within cycle c the first time that a group is transmitted for the second time and denote its position in the packet stream with n.

(EQ 21)

$$n = \min_{i \in \{cG, cG+1, cG+2, \cdots cG+G-1\}} / \exists j \in \{c \cdot G, c \cdot G+1, c \cdot G+2, \cdots i-1\} / C_i = C_j$$

Let us go further down the packet stream and denote with m the first packet position after n for which a group that has not yet been sent on cycle c is transmitted in C.

(EQ 22)

$$m = \min_{i \in \{n+1, n+2, \cdots, \infty\}} / \forall j \in \{c \cdot G, c \cdot G+1, c \cdot G+2, \cdots i-1\} / C_i \neq C_j$$

Note that if C is such that position m does not exist (i.e. keeps sending packets from the same group forever after position n) then our suggested optimal scheme is trivially better.

The improved schedule B will look exactly like C with packets in positions m-1 and m interchanged.

(EQ 23)

$$B_i = \begin{cases} C_i & i \neq m-1, i \neq m \\ C_m & i = m-1 \\ C_{m-1} & i = m \end{cases}$$

Since the packets sent so far are the same, the probability of success at or before packet i is received for i other than m-1 is the same for both schemes.

(EQ 24)

$$P\left(N_B^{G,K} \le i\right) = P\left(N_C^{G,K} \le i\right) \quad i \neq m-1$$

However as proved by Lemma A.3 - on page 194, the probability of success at or before packet *m*-1 is greater for *B* than for *C*.

We have shown then for both case 1 and case 2 that:

(EQ 25)

$$P\left(N_B^{G,K} \le i\right) \ge P\left(N_C^{G,K} \le i\right) \quad i = 0,1,\cdots \infty$$

and also that there is at least one *i* for which:

(EQ 26)

$$\exists i \,/\, P\left(N_B^{G,K} \le i\right) > P\left(N_C^{G,K} \le i\right)$$

From this and from:

(EQ 27)

$$E\left(N_S^{G,K}\right) = \sum_{i=0}^{\infty}\left(1 - P\left(N_S^{G,K} \le i\right)\right)$$

We get:

(EQ 28)

$$E\left(N_B^{G,K}\right) < E\left(N_C^{G,K}\right)$$

We have shown so far that for every schedule *C* other than *O* there is at least one schedule *B* which results in a better Avg. Since there are infinite schedules we yet need to show that our schedule improvement scheme converges to *O* when applied to the resulting improved schedule on and on. This result follows from the construction of *B*.

Clearly when the original *C* is such that it is covered by case 1 the resulting *B* is such that if regarded as a new *C* it will be covered by case 2. Therefore we only need to prove that the *B* proposed for case 2 converges to *O*.

From the construction of *B* we can see that if applying the same construction to *B* the resulting schedule would have resulted in *m* being one less than the one before (until *m* becomes *n*+1, this can happen after the first iteration) and then *n* would have advanced at least one position. After some more iterations *n* would cross a cycle boundary thus incrementing *c* and leaving the previous cycle equal to *O* in that range.

Finally, the packet schedule within each cycle follows from the fact that we desire each client to start receiving at any point in time with no impact to its perceived results. Therefore we need all cycles to have the same packet order to achieve group interleaving regardless of the starting point within the cycle.

qed

FEC with Group Interleaving Results

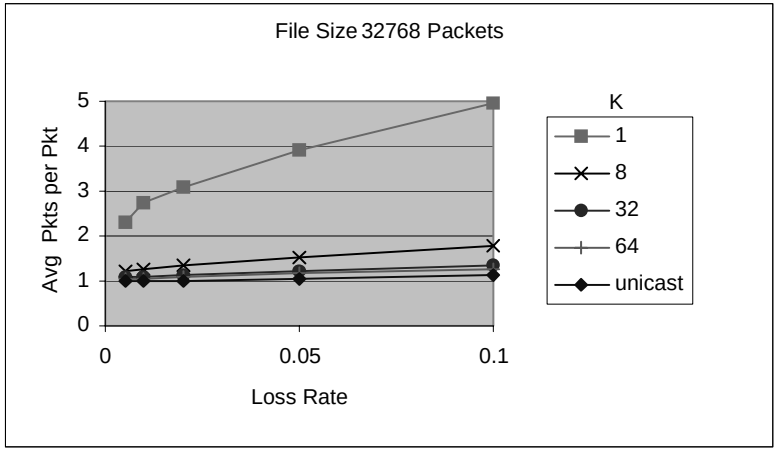
In the following figures we show the effect of FEC and group interleaving on the average amount of packets that a receiver will have to wait for complete reception of the file. We normalize the average by dividing it by the total number of packets in the file in order to make results comparable. We call this performance measure **avg packets per packet**.

(EQ 29)

$$\text{avg packets per packet} \equiv \frac{E\left(N^{G,K}\right)}{GK}$$

Figure 3.4 on page 57 shows the avg packets per packet as a function of the loss rate. We can clearly appreciate in this figure the benefit of FEC. The curve that shows the results for K=1 is the one that corresponds to the results of the previous chapter where no FEC was applied. As we see in the graph, increasing K has a big positive impact on the results. The lowest curve which corresponds to the ideal unicast case is our goal. We can see that by increasing K, the results of the **group interleaving** schedule approach our goal.

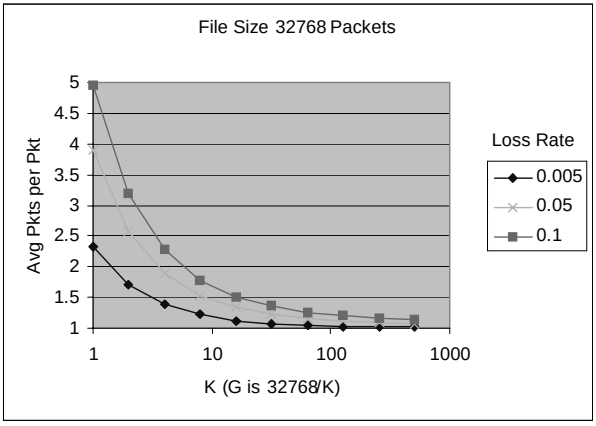
FIGURE 3.4 - Group Interleaving (Avg as a function of loss rate)



An important question is that of which value of *K* should be selected. From the results presented in these figures, the greater the K the better. However, as we have already said, for implementation reasons *K* has to be kept relatively small. We can also appreciate that the benefit increase decreases as we approach the ideal case. We use the results shown in the following charts in order to make the right choice.

In Figure 3.5 on page 58, we vary *K* (and calculate G as the total number of packets in the file divided by the chosen *K* so that the file size is kept constant). We show the results for some different loss rates in the same chart for the sake of comparison. As expected when *K* is 1 the results are the same as in the previous chapter. We clearly see that results improve drastically when *K* grows. Finally as *K* gets closer to the file size (which turns G into smaller and smaller numbers) the result approaches that of a unicast transmission. As expected the unicast result is the same as that for the case where *K* is equal to the file size (and therefore *G* is 1).

FIGURE 3.5 - Group Interleaving (Average as a function of K)



It can also be appreciated from this figure that the additional benefit of increasing *K* decreases as *K* grows. In particular, by picking *K* around 32 we see that a great portion of the FEC benefit has already been achieved. We further emphasize this point in Figure 3.6 on page 59 where we show the percentage of benefit obtained for every value of *K*. We define 100% benefit to be that of using *K* equal to the file size (unicast equivalent results) and 0% to be the one for the simple cyclic multicast case of the previous chapter.

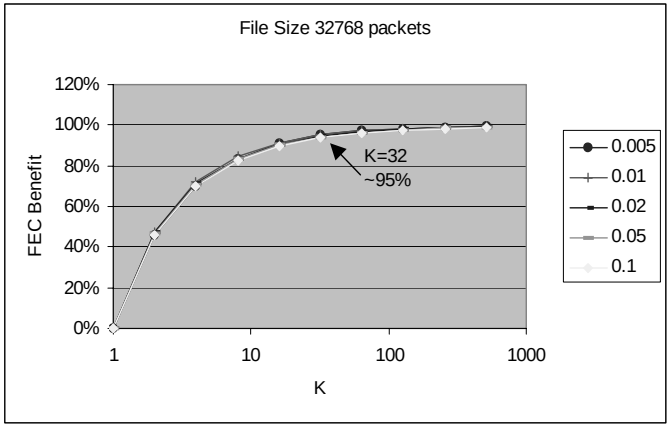
The FECBenefit is then given by:

(EQ 30)

$$FECBenefit(K) = \frac{E\left(N^{S,1}\right) - E\left(N^{S/K,K}\right)}{E\left(N^{S,1}\right) - E\left(N^{1,S}\right)}$$

where *S* is the file size in packets.

FIGURE 3.6 - Group Interleaving (FEC Benefit as a function of K)



As we see in Figure 3.6 on page 59 the dependence of the benefit of using FEC on the value of *K* is not very much affected by the loss rate. This is not to be confused with the expected average amount of packets needed which are indeed sensitive to the loss rate as we see in Figure 3.4 on page 57. A salient consequence of what we appreciate in Figure 3.6 on page 59 is that we can pick a value of *K* such as 32 (which is perfectly implementable as far as computability of the FEC code is concerned) and that same value is suitable for a wide range of loss rates. For that value of *K* we see that well over 90% of the benefits of FEC are achieved.

Finally, once we pick *K* to be 32, we show in Figure 3.7 on page 60 the effect both file size and avg loss rate on the expected packets-per-packet measure. The results are clearly much better than those presented in the previous chapter for the simple cyclic multicast with no FEC. We obtain results smaller than 2 even for very large files and big loss rates. Even in situations where the typical loss rate is around 10% (still a large one indeed) the results for large files[2] (32MB) is around 1.2 packets-per-packet which is very satisfactory. The logarithmic nature of the impact of file size in the results (as seen for the previous simple cyclic) is preserved here as well (note the log scale of the file size axis), However, as we shall see in a future section, its effect is mitigated since it is divided by *K* which being 32 reduces its influence in a significant manner.

2. G=1024, K=32, Packet Size=1KB

FIGURE 3.7 - Group Interleaving Results (K=32)

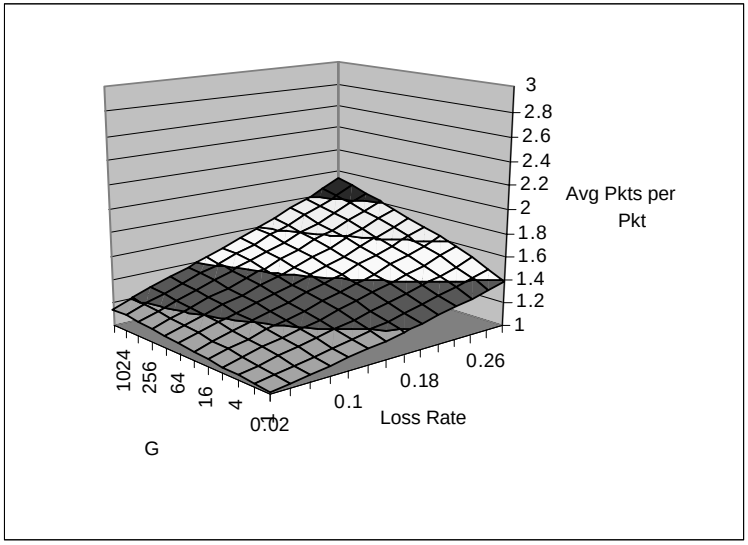
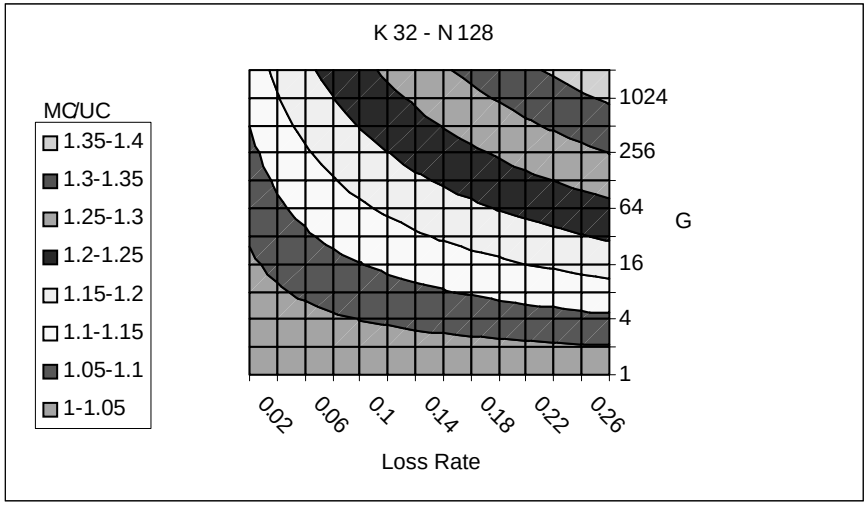


FIGURE 3.8 - Group Interleaving (Compared to Unicast - K=32)

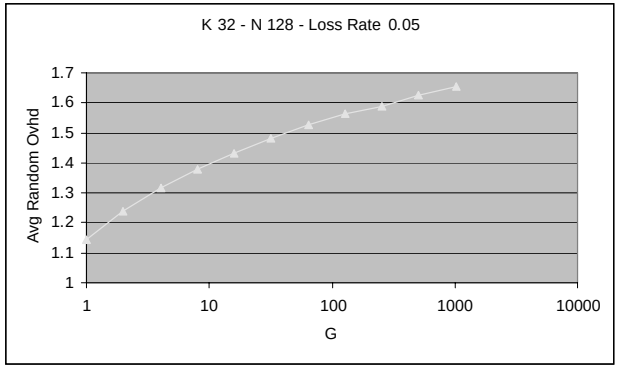


In Figure 3.8 on page 60 we compare the results attained with FEC (*K*=32) to those of the ideal selective retransmission protocol (reliable unicast). We plot the ratio between the results shown in Figure 3.7 on page 60 and those of the equivalent unicast case, as a function of file size and avg loss rate. As it can be seen, the results are now comparable and satisfactory. As an example, let us consider the case where the average loss rate is 6% and the file is 32MB long (G=1024). Figure 3.8 on page 60 shows that the extra amount of time perceived by a client of cyclic multicast with FEC as compared to what would have been the case if he was privately[3] downloading the file using reliable unicast, is merely about 20%[4] higher. We claim these results to be very satisfactory. Specially when considering the fact that the same transmission is now being made available to any number of simultaneous non-synchronized receivers with optimal utilization of network resources[5].

Comparing Group Interleaving to Random Schedule

In order to emphasize the necessity for a careful packet schedule scheme such as the one we described above (and proved to be optimal), we compare the results of the **Group Interleaving** schedule to those of a Random Packet Schedule. Under the Random Schedule, in every time slot, the group to which the packet that is going to be sent belongs, and the coded packet within that group, are randomly selected with a uniform distribution.

FIGURE 3.9 - Group Interleaving vs. Random (Function of File Size)



3. This is a theoretical comparison since as we have shown at the beginning of this work, reliable unicast to all receivers is not an option. Actual results if attempted would be much worse than anything because of the heavy congestion at the server's connection to the network.
4. In a real situation it is expected that this value will be even lower because of the burst type correlation of packet losses.
5. We formally define the concept of optimal network utilization in the next chapter.

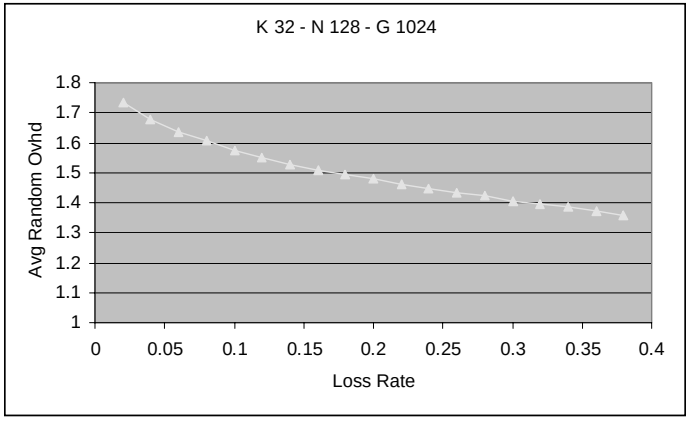
Figure 3.9 on page 61 compares the results of our optimal schedule to the ones that would be achieved if packets were scheduled in a random manner. We plot the Random Overhead which is the number of packets needed to receive the file under random schedule divided by the number of packets needed when the proposed **group interleaving** schedule is applied. We can clearly appreciate the benefits of the **group interleaving** schedule which become even bigger as the file size grows. For files of reasonable relevant size, the overhead of a random schedule are on the order of 60%.

Note that when the number of groups (G) is 1 the Random Overhead is still greater than 1. This is because even though there is no difference in the group of the scheduled packets (all packets belong to the same single group) there is still a probability that the exact same packet is scheduled more than once.

Figure 3.10 on page 62 shows the Random Overhead as a function of the average loss rate. As expected, we see the overhead decrease as the loss rate grows. This is because when the loss rate is bigger and lots of packets are missed random selection hits more frequently a needed packet than when the loss rate was small. We appreciate though that for the relevant range of loss rates the overhead of the Random schedule is very large.

With this we conclude that the group interleaving schedule is not just optimal but also necessary to achieve results comparable to those of an equivalent selective retransmission protocol. This result will be of special importance in the next chapter when multi-rate versions of the schedule are developed in order to maintain the optimal properties of the group interleaving.

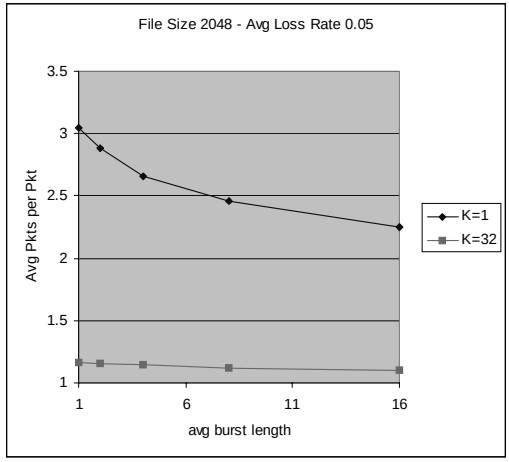
FIGURE 3.10 - Group Interleaving vs. Random (Function of Loss Rate)



Time Correlated Losses

Figure 3.11 on page 63 shows the effect of burst losses in the performance results for two different values of *K*. For *K*=1 which models the case where FEC is not applied, we can see that loss burstiness has a great impact in the final results (as we have seen in the previous chapter). For *K*=32 the effect of bursts is much smaller. Results are improved when the average burst length increases but the relative impact of this effect is small as compared to the no FEC case. This is expected. When the FEC group size grows, the identity of lost packets becomes less important as there are more packets in the stream that can be used to recover it. In particular when the FEC is applied to the whole file, the statistical properties of the loss burstiness have no effect whatsoever in the results. Such a scheme (very much like selective retransmission) is only impacted by the overall number of losses.

FIGURE 3.11 - Group Interleaving - Error correlation - Different K



3.3 - A Bound for Multicast Bulk Data Distribution with FEC

Our goal in this section is to derive an expression for the minimum number of packets that need to be transmitted to a client so that the whole file is successfully received with a probability of *P*. We normalize this number by dividing it by the number of packets in the file so to make the results comparable among different cases.

So far we used computer generated calculations that gave us some insights on the performance of the group interleaving algorithm as compared to an ideal point to point reliable transport. In this section we wish to bound the success probability and obtain an expression from which the number of packets that need to be transmitted for the success probability to be higher than *P* can be calculated analytically.

It is important for us to achieve a simple analytical expression so that we can get further insights about the dominant factors in the performance of the proposed algorithm. We start by looking at a single FEC group and then extend the results to the multi-group case.

Bounding the Probability of Success for a Single Group

We want to bound the probability of successful reception of *K* packets of a single group at the point in time or before *c* packets of that group have been transmitted which is given by:

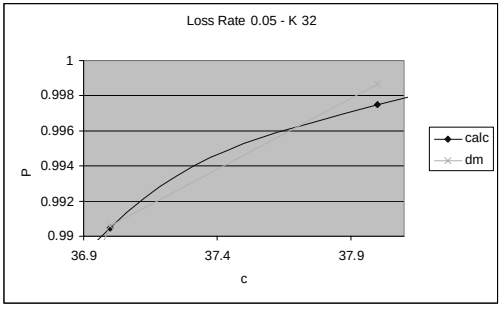
(EQ 31)

$$P\left(N^{1,K} \le c\right) = \sum_{i=K}^{c} \binom{c}{i} (1-q)^i q^{c-i}$$

As follows from this expression, when analyzing the probability of success for a single group (assuming the number of coded packets is infinite) the error handling model is identical to that of unicast transmission of a *K*-packet file.

In a previous chapter we used the DeMoivre-Laplace theorem to approximate the probability of success at or before *c* packets for a unicast case. This analysis somewhat resembles what we desire to show in this section. Figure 3.12 on page 64 shows the DeMoivre-Laplace approximation for a relevant set of transmission parameters.

FIGURE 3.12 - DeMoivre-Laplace Approximation Drawbacks



As we can see from the chart, the DeMoivre-Laplace approximation is not suitable for our needs in this case. Unfortunately the main condition for the DeMoivre-Laplace theorem to make a close approximation does not hold. The theorem requires $cq(1-q)$ to be a large value. In our case, where small values of q are highly relevant, and K (which is in the same order of c) can be a low value such as 32, 16 or even 8, this condition does not hold. A more important reason for the unsuitability is the fact that the DeMoivre-Laplace theorem makes an approximation and not a bound as we see in the chart above. Even small differences could have a large impact on our calculation since we wish to estimate c for probability figures very close to 1 (as we want to calculate the probability of simultaneous success for G groups, for a very large G).

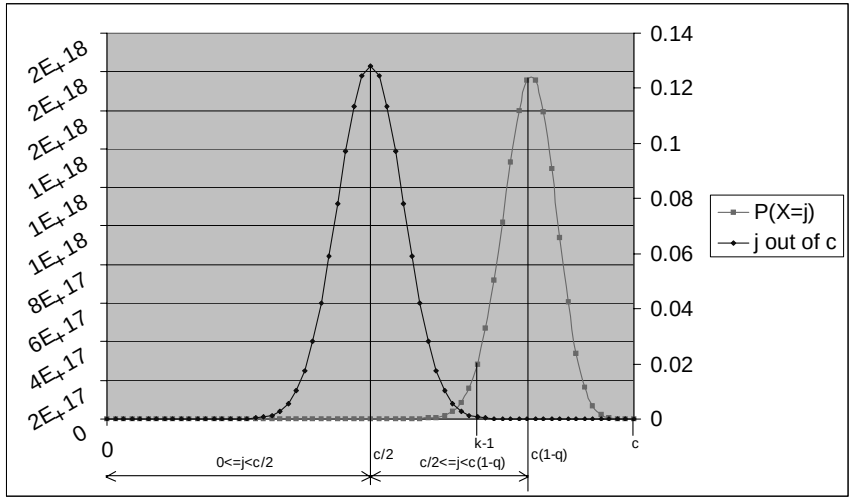
A further disadvantage of the DeMoivre-Laplace theorem is the fact that it is based on lookups to a Gaussian distribution table. This lookup is clearly non-invertible and therefore not suitable to put in the form of an analytical expression.

Let us then define a random variable X^c which is the number of successful packet receptions out of c packets that were sent. Clearly:

(EQ 32)

$$P\left(N^{1,K} \le c\right) = 1 - P\left(X^c < K\right)$$

FIGURE 3.13 - Domain for the bound



And the density function for X^c which is plotted in Figure 3.13 on page 65 is given by:

(EQ 33)

$$f_{X^c}(j) = P(X^c = j) = \binom{c}{j}(1-q)^j q^{c-j}$$

where q is the loss probability per packet and p is 1-q.

A known bound is the one given by the Tchevycheff inequality.

(EQ 34)

$$P\left(\left|X^c - E\left(X^c\right)\right| \geq \varepsilon\right) \leq \frac{\sigma^2}{\varepsilon^2}$$

For the binomial distribution:

(EQ 35)

$$E\left(X^c\right) = cp$$

and the variance:

(EQ 36)

$$\sigma^2 = cpq$$

We aim to calculate:

(EQ 37)

$$P\left(X^c < K\right) = P\left(\left|X^c - E\left(X^c\right)\right| \geq \varepsilon\right) \rightarrow \varepsilon = cp - K$$

but in order for the result to be relevant we need to make sure that c is big enough so that:

(EQ 38)

$$\varepsilon = cp - K > c - cp$$

which is true for:

(EQ 39)

$$c > \frac{K}{p - q}$$

In that range, the inequality gives us:

(EQ 40)

$$P\left(N^{1,K} \le c\right) = 1 - P\left(X^{c} < K\right) \ge 1 - \frac{cpq}{\left(cp - K\right)^2}$$

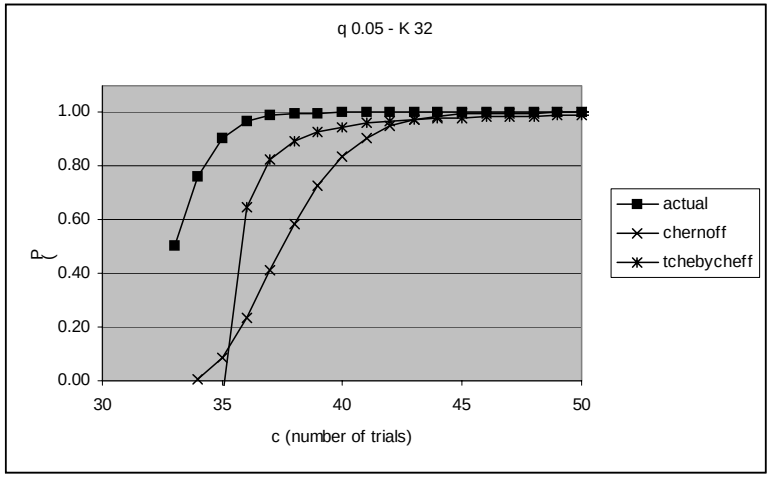
Another applicable known bound is the Chernoff [32] bound which for our case looks as follows:

(EQ 41)

$$P\left(N^{1,K} \le c\right) = 1 - P\left(X^{c} < K\right) \ge 1 - e^{\frac{-2\left(cp-K\right)^2}{c}}$$

We compare the results of the Tchebycheff and Chernoff bounds against the actual values in Figure 3.14 on page 67. As can be seen, these bounds are very loose and therefore not suitable for a meaningful analysis.

FIGURE 3.14 - Tchebycheff and Chernoff bounds



We proceed then to develop our own bound. In our approach we take advantage of the observation that for the relevant range of *j*, the distribution under analysis decreases faster than a geometric sequence with the decrease of *j*.

Let us define:

(EQ 42)

$$\rho_j \equiv \frac{P\left(X^c = j-1\right)}{P\left(X^c = j\right)}$$

then:

(EQ 43)

$$\rho_j = \frac{\binom{c}{j-1} p^{j-1} q^{c-j+1}}{\binom{c}{j-1} p^j q^{c-j}} = \frac{\frac{c!}{(j-1)!(c-j+1)!}\left(\frac{p}{q}\right)^{j-1} q^c}{\frac{c!}{(j)!(c-j)!}\left(\frac{p}{q}\right)^{j} q^c} = \frac{j}{(c-j+1)}\frac{q}{p}$$

We are interested in a range for which the ratio should be smaller than one (*P* decreases as *j* decreases), therefore:

(EQ 44)

$$\left(\rho_j < 1\right) \rightarrow \left(jq < cp - jp + p\right) \rightarrow \left(jp + jq = j < cp + p\right)$$

which matches what we see in Figure 3.13 on page 65.

The ratio is clearly an increasing function of *j* for $j<cp+p$. And then for $c>K/p$ we get:

(EQ 45)

$$\max_{0 \le j < K} \rho_j = \rho_{K-1} = \frac{(K-1)q}{(c-K+2)p}$$

We use now the ratio to bound the density for X^c as follows:

(EQ 46)

$$P\left(\underset{0\le j<K}{X^c = j}\right) = \rho_{j+1}P\left(X^c = j+1\right) < \rho_{K-1}P\left(X^c = j+1\right) <$$
$$< \rho_{K-1}\left(\rho_{K-1}P\left(X^c = j+2\right)\right) < \cdots < \left(\rho_{K-1}\right)^{K-1-j}P\left(X^c = K-1\right)$$

Then:

(EQ 47)

$$P\left(\underset{c>K/p}{X^c < K}\right) = \sum_{j=0}^{K-1} P\left(X^c = j\right) < \sum_{j=0}^{K-1}\left[\left(\rho_{K-1}\right)^{K-1-j}P\left(X^c = K-1\right)\right] =$$
$$= P\left(X^c = K-1\right)\sum_{j=0}^{K-1}\left(\rho_{K-1}\right)^j$$

We tried a few different approaches here namely:

1. Take advantage of the fact that the corresponding infinite geometric series converges and then use:

(EQ 48)

$$P\left(\underset{c>K/p}{X^c < K}\right) < P\left(X^c = K-1\right)\sum_{j=0}^{\infty}\left(\rho_{K-1}\right)^j = \frac{P\left(X^c = K-1\right)}{1-\rho_{K-1}}$$

2. Use the finite geometric series:

(EQ 49)

$$P\left(\underset{c>K/p}{X^c < K}\right) < P\left(X^c = K-1\right)\sum_{j=0}^{\infty}\left(\rho_{K-1}\right)^j = \frac{1-\left(\rho_{K-1}\right)^K}{1-\rho_{K-1}}P\left(X^c = K-1\right)$$

3. Further split the range into smaller portions (i.e. j<c/2 when q<p) to reduce the difference between the bound and the calculated value.

(EQ 50)

$$P\left(\underset{c>K/p}{X^c < K}\right) < P\left(X^c = K-1\right)\left[\sum_{j=0}^{c/2}\left(\rho_{c/2}\right)^j + \sum_{j=c/2+1}^{K-1}\left(\rho_{K-1}\right)^j\right]$$

We tried all approaches and found out the simplest one provides a bound that is very close to the calculated value for a wide range of the parameters. We leave the improvement of the bound using the mentioned techniques as an option when closest approximations are desired.

Summarizing, our proposed bound is given by:

(EQ 51)

$$\underset{c>K/p}{P\left(X^c<K\right)}<\frac{P\left(X^c=K-1\right)}{1-\rho_{K-1}}=\frac{(c-K+2)p}{cp-K+1+p}\binom{c}{K-1}p^{K-1}q^{c-K+1}$$

and therefore:

(EQ 52)

$$\underset{c>K/p}{P\left(N^{1,K}\le c\right)}=1-P\left(X^c<K\right)>1-\frac{(c-K+2)p}{cp-K+1+p}\binom{c}{K-1}p^{K-1}q^{c-K+1}$$

We show the results in Figure 3.15 on page 70 and Figure 3.16 on page 71 for loss rates of 0.05 and 0.25 respectively. We compare the attained bound with the computer generated exact calculation.

FIGURE 3.15 - Bound for P(N^{1,K}<=c) - Loss Rate 0.05

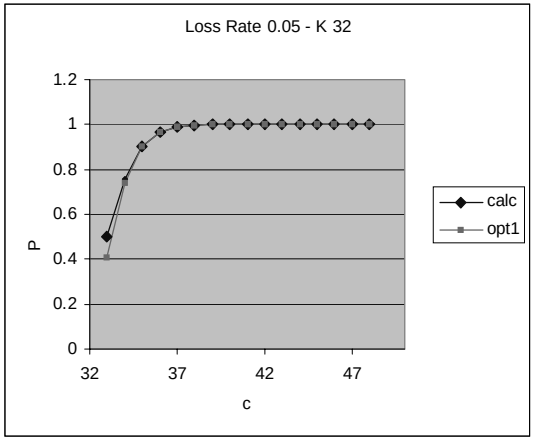
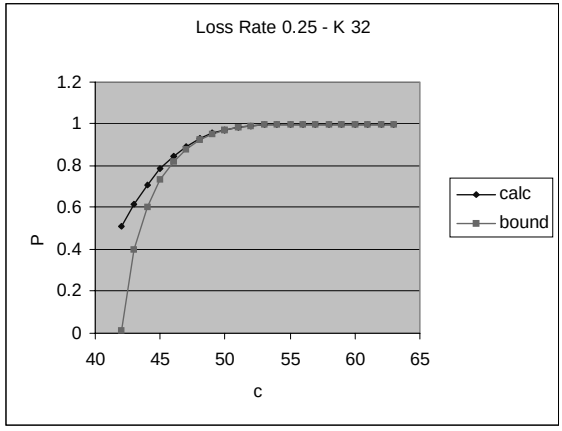
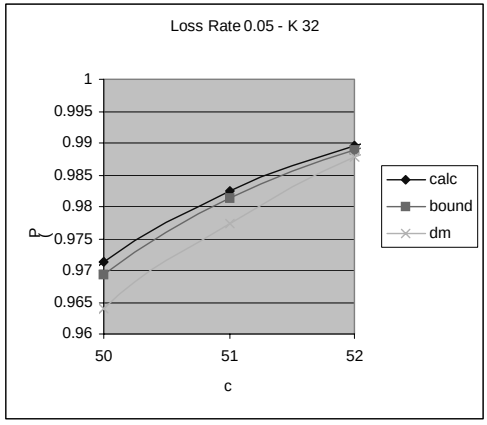


FIGURE 3.16 - Bound for $P(N^{1,K}<=c)$ - Loss Rate 0.25



The results are clearly satisfactory. The bound is very close in all areas of interest (when P is sufficiently close to 1) even for relatively large values of *q* such as 0.25 as we see in Figure 3.16 on page 71.

FIGURE 3.17 - Bound for $P(N^{1,K}<=c)$ - Comparison with DeMoivre-Laplace



To further emphasize the properties of the suggested bound we show in Figure 3.17 on page 71 a zoomed in section of one of the previous charts. In this figure we can see the calculated version compared to our bounds and to the DeMoivre-Laplace approximation.

Isolating the Needed Number of Cycles from the Bound

There is still more to be done in order to attain our goal of getting an expression from which we can isolate *c* as a function of all the other parameters (*K*, *q* and the desired success probability). The bound we presented so far is highly valuable for estimating *P* as we have shown above. However it is hard to isolate *c* from it. For that purpose we further evolve it as follows.

The idea is to simplify the bound so that *c* appears in a form that can be easily isolated. We start with the first factor that has *c* in the expression above.

We have:

(EQ 53)

$$\left(c > \frac{K}{p}\right) \rightarrow \frac{(c-K+2)p}{cp-K+1+p} < \frac{\left(\frac{K}{p}-K+2\right)p}{\frac{K}{p}p-K+1+p} = \frac{Kq+2p}{1+p}$$

So:

(EQ 54)

$$P\left(N^{1,K}_{c>K/p} \le c\right) > 1 - \frac{Kq+2p}{1+p}\binom{c}{K-1}p^{K-1}q^{c-K+1}$$

This is still not enough to make the isolation of *c* practical. The combinatorial expression above is hard to invert. In order to further simplify it we use Newton's binomial as follows:

(EQ 55)

$$2^c = (1+1)^c = \sum_{i=0}^{c}\left[\binom{c}{i}\cdot 1^i \cdot 1^{c-i}\right] =$$

$$= \sum_{i=0}^{K-2}\binom{c}{i} + \binom{c}{K-1} + \sum_{i=K}^{c}\binom{c}{i} > \binom{c}{K-1}$$

and then substitute in the bound to obtain:

(EQ 56)

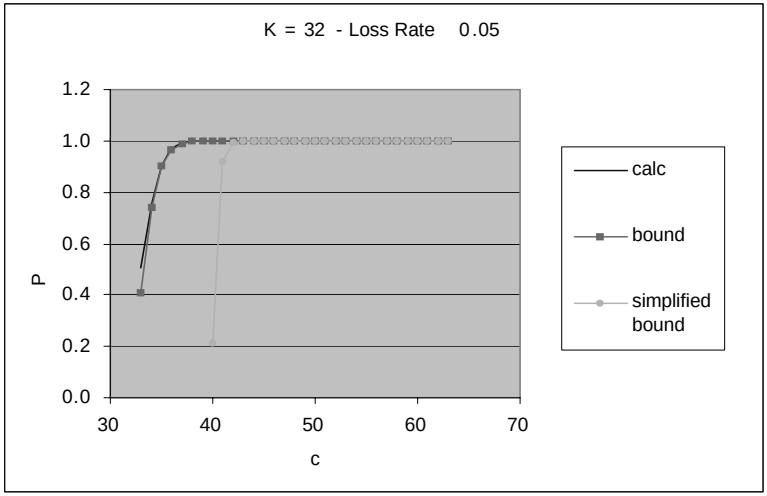
$$P\left(\underset{c>K/p}{N^{1,K}} \le c\right) > 1 - \frac{Kq+2p}{1+p}\left(\frac{p}{q}\right)^{K-1} 2^c q^c$$

which is an expression from which *c* can be easily isolated as desired.

We show in Figure 3.18 on page 73 the invertible bound as compared to the previous one. Clearly the results are worse than before. The biggest impact is that of the use of 2^c which highly simplifies the calculation at the expense of worsening the bound. The important bound feature is still preserved and as we see below when we isolate *c* and compare its estimation with the calculated values the results are very good.

We clearly see two sections in Figure 3.18 on page 73. The first part of the chart where the simpler bound has its worse behavior is related to the fact that we are multiplying a yet relatively high residual probability by a number much larger than what should have been. On the other hand for higher values of *c*, even though 2^c is yet higher, the probability by which it is multiplied is very small making its effect negligible. In particular, as we will see below, since we need simultaneous success for a large number of groups (*G*) we will be interested in cases where *P* is very close to 1 a range at which the invertible bound is very good.

FIGURE 3.18 - Invertible Bound for $P(N^{1,K}<=c)$ - Loss Rate 0.05



Isolation of *c* for the single group case is from here straightforward. Given a desired success probability *PP,*

(EQ 57)

$$(1-PP)=\frac{Kq+2p}{1+p}\left(\frac{p}{q}\right)^{K-1}2^c q^c \rightarrow$$

$$\rightarrow (2q)^c=(1-PP)\left(\frac{1+p}{Kq+2p}\right)\left(\frac{q}{p}\right)^{K-1}$$

The number of cycles *c* that a client needs to receive the file with probability *PP* is then at least:

(EQ 58)

$$c=\frac{\ln(1-PP)+\ln\left(\frac{1+p}{Kq+2p}\right)+(K-1)\ln\left(\frac{q}{p}\right)}{\ln(2q)}$$

This result is an intermediate step. In our scenario we have *G* groups to take care of simultaneously. We further develop our bound for *c* in the multiple group scenario in next section.

Accounting for Multiple Simultaneous FEC Groups

When considering the *G* simultaneously transmitted FEC groups with *K* packets each that resulted from dividing a file of *S* packets for FEC computability our success probability function at or before *c* packet cycles becomes[6]:

(EQ 59)

$$P(N^{G,K}\le c)=\left[P(N^{1,K}\le c)\right]^G$$

So if *PP* is the desired success probability then *c* needs to be such that:

6. we use a slightly simpler model here where N^{GK} represents the number of cycles as opposed to the number of packets. A cycle is the time that takes to send G packets which according to our schedule scheme belong each to one different group. This is similar to the simplest approach we took in the previous chapter however in this case the impact in the result is somewhat smaller since a whole cycle is much smaller than the file.

(EQ 60)

$$P\left(X^{c} < K\right) = 1 - \sqrt[G]{PP}$$

Using the formula we achieved in the previous section we get:

(EQ 61)

$$c = \frac{\ln\left(1 - \sqrt[G]{PP}\right) + \ln\left(\frac{1+p}{Kq+2p}\right) + (K-1)\ln\left(\frac{q}{p}\right)}{\ln(2q)}$$

which is the bound we were looking for.

A Practical Approximation - Isolating the Effect of G

There is yet an improvement that can be done to the expression for *c* above. The effects of *G* and *PP* can be made clearer if we manage to separate them. Since *PP* is close to 1 we can approximate:

(EQ 62)

$$\sqrt[G]{PP} = e^{\frac{\ln PP}{G}} \approx 1 + \frac{\ln PP}{G}$$

and:

(EQ 63)

$$\ln PP \approx -(1 - PP)$$

So we are looking for *c* so that:

(EQ 64)

$$P\left(X^{c} < K\right) = 1 - \sqrt[G]{PP} \approx \frac{1}{G}(1 - PP)$$

Using the formula we achieved in the previous section we get:

(EQ 65)

$$c = \frac{-\ln(G) + \ln(1-PP) + \ln\left(\frac{1+p}{Kq+2p}\right) + (K-1)\ln\left(\frac{q}{p}\right)}{\ln(2q)}$$

which is the estimation[7] for *c* we were looking for.

Let us now normalize *c* by dividing it by *K* in order to obtain the packets-per-packet measure.

(EQ 66)

$$c^* = \frac{-\ln(G) + \ln(1-PP) + \ln\left(\frac{1+p}{Kq+2p}\right) + (K-1)\ln\left(\frac{q}{p}\right)}{K\ln(2q)}$$

Remark: A salient observation of this expression is that the effect of the file size in the packet-per-packet measure is divided by the FEC parameter *K*. Recall that in the previous chapter, when the "simple cyclic multicast with no FEC"[8] was analyzed, we concluded that the logarithm of the file size was a determinant factor in its performance. We see now that through the use of FEC, this effect gets "diluted" by the division by K.

The following figures verify our estimated *c** against the one calculated using the exact probability functions. We do that as a function of all the parameters in order to understand the limitations of the proposed estimation. The results are highly satisfactory. The expected differences (mainly from the use of 2^c) do not distort the general effect of the parameters in the estimated *c** and therefore a lot of interesting features can be analyzed.

7. We call this an estimation (as opposed to a bound) because of the approximation we took for PP^1/G. As we see below the behavior of the estimation is still on the safe side as we intended it to be while developing the bound.
8. equivalent to K=1

FIGURE 3.19 - c* as a function of K - Loss Rate 0.05

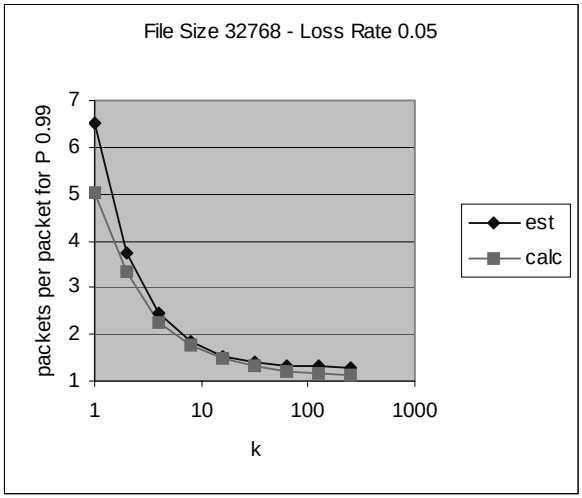


Figure 3.19 on page 77 shows *c** as a function of *K* for the same file of size 32768 packets. As *K* grows *G* decreases so to keep *GK* equal to 32768.

The figures show that the estimation is pretty satisfactory. The qualitative features are completely preserved. Even though at this stage the plotted expression is an estimation[9], we can clearly observe that it is always on the safe side of the calculated value.

We can see that for very small values of *K*, the difference between the estimation and the calculated value is not negligible. We try to asses this worst case difference by analyzing our estimation in the extreme case where *K*=1 (simple cyclic multicast with no FEC). This is the case we analyzed in the previous chapter. Substituting *K*=1 in our estimation we get:

(EQ 67)

$$c = \frac{-\ln(G) + \ln(1-S) + \ln\left(\frac{1+p}{q+2p}\right)}{\ln(2q)}$$

9. because of the estimation for PP^1/G

The third term is quite negligible since the argument of the logarithm is very close to 1. If we eliminate it we obtain:

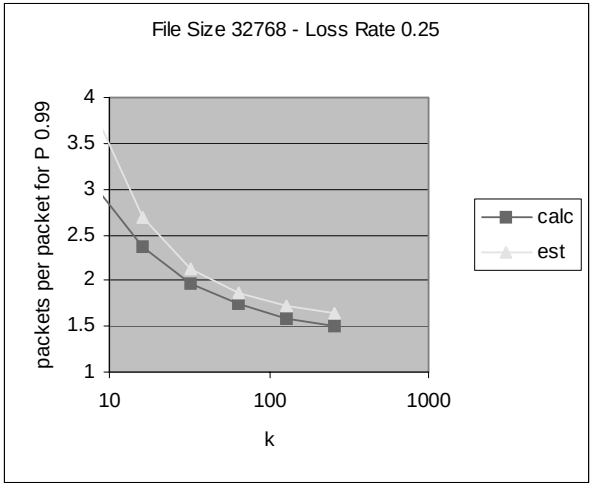
(EQ 68)

$$c = \frac{\ln(1-S)}{\ln(2q)} - \frac{\ln(G)}{\ln(2q)}$$

which is close in its form to the results we obtained for the estimations of the simple case in the previous chapter. The qualitative impact of the parameters is exactly the same. The difference in the argument for the logarithm in the denominator is the source for the difference and it comes from the compromises we had to take to be able to isolate *c* in a simple manner.

Figure 3.20 on page 78 repeats the same experiment for a higher loss rate. 0.25 was picked because it is pretty much at the edge of the relevant loss rate range for a wide variety of applications. The results here are very much like those of the previous example.

FIGURE 3.20 - c* as a function of K



A careless reader may wrongly conclude from Figure 3.20 on page 78 that the estimation converges to the calculated value for larger *K*. This is not true. The calculated value converges to the unicast result when *K* is equal to the file size (and *G* is consequently 1) as expected (we have proved this convergence in a previous section). The estimation converges to a somewhat higher value given by:

(EQ 69)

$$c = \frac{\ln\left(\frac{q}{p}\right)}{\ln(2q)}$$

This is again mandated by the compromises we made in order to achieve a simple invertible expression.

Figure 3.21 on page 79 shows the behavior of our estimation as a function of the loss rate. The file size is kept constant at 32768 packets as well as K which is held at 32.

FIGURE 3.21 - c* as a function of Loss Rate

An interesting observation is that for any particular value of *K*, the difference between the calculated c and the estimated one (plotted on the same chart) as a function of the loss rate is fairly constant for a wide and relevant range of loss probabilities. We will see below that this observation holds when the estimation is tested as a function of the file size as well. We therefore apply a heuristic correction factor to our estimation as follows:

(EQ 70)

$$c^{**} = \frac{-\ln(G) + \ln(1-PP) + \ln\left(\frac{1+p}{Kq+2p}\right) + (K-1)\ln\left(\frac{q}{p}\right)}{K \cdot \ln(2q)} - \varepsilon(K)$$

where:

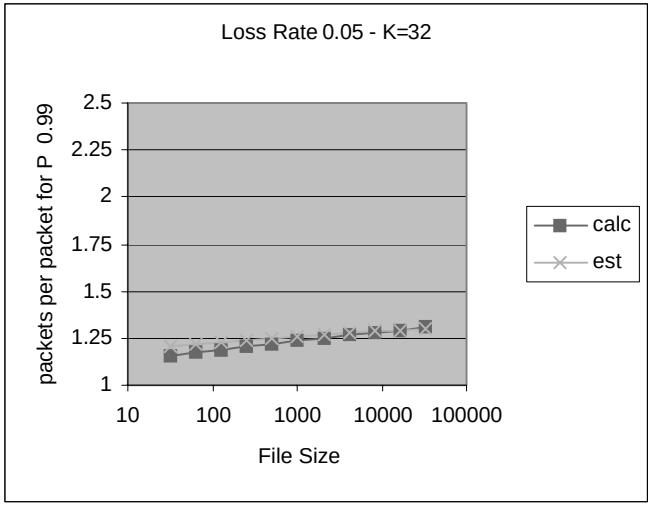
(EQ 71)

$$\varepsilon(K = 32) = 0.1$$

as we can see from Figure 3.21 on page 79.

We verify the corrected estimation (c^{**}) as a function of the file size in Figure 3.22 on page 80. K is fixed at 32 so as the file size increases the number of groups does so accordingly. We show the estimation for a loss rate of 0.05.

FIGURE 3.22 - c as a function of the file size (Loss Rate 0.05)**

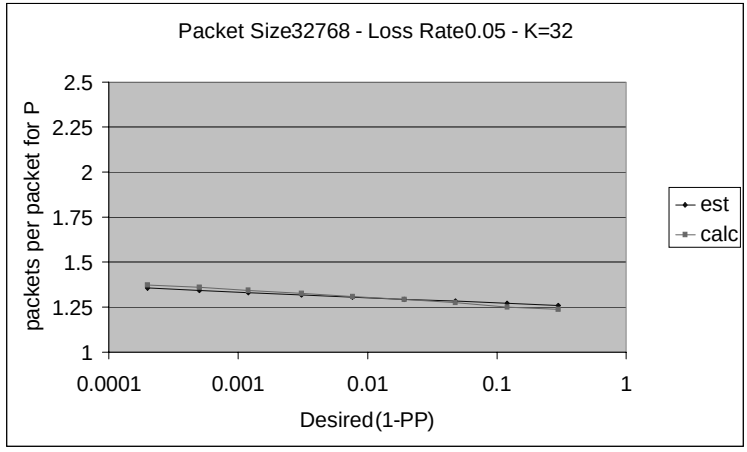


As can be seen, the corrected estimation is very close to the calculated value. This is especially true for file sizes in the relevant interesting range.

Remark: Summarizing, we have shown an estimation formula that behaves very close to what the calculated value does for a wide range of loss rates and virtually any file size. Since the file size and the average loss rate are effectively the actual changing parameters, we claim that for any specific implementation[10] we provide a simple expression from which performance results from a receiver perspective can be estimated with fairly good precision.

We finally evaluate our estimation as a function of the desired success probability *PP* and show the results in Figure 3.23 on page 81. Again we see the estimation is very close to the calculated value. This result enhances the usability of our estimation for the cases where the desired success probability may vary among different cases.

FIGURE 3.23 - c as a function of the desired success prob (Loss Rate 0.05)**



10. implies a specific value of *K*

CHAPTER 4

Multi-rate Distribution to Heterogeneous Clients

4.1 - Congestion Control in Multicast

Introduction

Multicast communication over a packet switched network has its main challenges deeply related to its very nature: One to Many Transmission. A large set of arbitrarily dispersed and heterogeneous receivers renders the issues of packet loss handling and congestion control into a major challenge as traditional feedback-based solutions applied to point-to-point communications get immediately disqualified.

Transmission errors and packet losses which are handled in traditional point to point communication in a variety of ways result in a much more difficult problem to overcome in point to multipoint transmissions. Feedback gets limited by scalability considerations and selective retransmission techniques get prohibitive as their number would increase unbounded with the growth in the number of receivers.

Flow and congestion control, which are a determinant factor in the performance of very large networks, impose the second limiting factor. The dispersion in location of receivers and the variety in their connections to the network make traditional control schemes unusable. Feedback based methods collapse for scalability reasons and, more important, no single data rate would be adequate for most point to multipoint applications with a large number of receivers.

We have shown in previous chapters the use of cyclic transmissions methods combined with forward error correction techniques in order to provide scalable packet loss handling, yet refraining from any kind of feedback channel requirements. We analyzed the performance from the receivers point of view and showed a scheme which was optimal in terms of expected delay. In this chapter we attack the problem of congestion avoidance with the same scalability goal in mind.

The most widely deployed method for congestion control in large networks is the one used by TCP. Routers in the TCP/IP internet drop packets as a result of congestion. The TCP injection endpoints react to packet drops according to specified rules. Basically, packet injection rate is reduced when packet drops are detected, and is gradually increased when transmissions succeed. This policy has as its main objective the relief of network congestion.

The research areas of congestion detection, packet dropping policies and endpoint reaction to congestion have been and still are of major interest to the scientific community and the networking industry. A variety of schemes have been proposed and tested with different degrees of success. Virtually all of these approaches are based on feedback and are therefore unsuitable to our problem.

Even in the presence of some kind of feedback channel, the congestion control issue for multicast is still problematic. Packet injection rate affects all receivers simultaneously while these may be located at different areas of the network that experience congestion in a different manner. In a simplistic approach, since at any point in time different bandwidth is available to different clients, the transmission rate for the multicast session could be made suitable for the weaker receivers. If all clients were connected to the server using links that limited their bandwidth to some similar value then this simplistic solution may have been acceptable to some extent. However, from a realistic standpoint, when some users have larger available bandwidths such as a 128Kbps ISDN or higher speed lines like cable modems (2Mbps), T1 or E1 (1.5-2Mbps), the suggested technique is not practicable. Imagine some thousands customers receiving the file at 14Kbps because one single listener to the multicast has an old modem.

An alternative approach to congestion control in multicast transmissions where routers along the way filter the packet stream so that corresponding sections of the tree get their adequate rate is discussed in [16]. However, the main limitation of these kind of approaches is related to the difficulty in the deployment of such a solution in a huge global networks. New protocols for which it is sufficient to update the interested endpoints with no change required to the network infrastructure are much more likely to succeed in an environment like the internet.

Receiver-Driven Flow Control

As mentioned above, the key to true scalability for the multicast approach proposed in previous chapters is that it provides a solution to the packet loss problem with no requirements for a feedback channel. In this section we develop a solution for the congestion control problem with the same motivation in mind. For this

purpose, transmission rate control has to be receiver driven. Each receiver has to be able to independently regulate its own reception rate according to its perceived state of the network congestion on the path from the server to itself.

In order to implement such a solution, multicast group membership mechanisms can be used. The simplest receiver driven mechanism could involve a server simultaneously sending the same file at different rates using different multicast addresses. Each receiver could join the one multicast address that matches its reception rate. The clear problem with such an approach is that network resources would be far from being optimally utilized. Let us consider two receivers topologically located at nearby places in the network but with different connection rates in their respective final hops. Because of the different rates, the two receivers would subscribe to different multicast channels resulting in extra data flowing through the common parts of their path. We further address this issue and formalize some related results in “The Optimal Network Utilization Paradigm” on page 87.

The receiver driven approach can be further evolved into one in which clients subscribe to one or more than one channel simultaneously according to their rate capabilities. We will show that by using multiple channels with appropriate subscription policies, the network utilization can be made optimal.

As far as network dynamics are concerned, a receiver driven approach can be adequate as well. In response to packet drops, receivers would drop their subscription to one or more of the multicast channels they are currently receiving. When coordinated among the clients, this packet-loss triggered group-unsubscription effectively reduces the network traffic on the congested paths as a direct result of the multicast group pruning that takes place in the intervening routers.

It is an important goal for the proposed solution to behave in a friendly way to other network traffic. The receiver driven approach as described in the previous paragraph resembles the behavior of TCP as far as reaction to congestion is concerned and therefore it is applicable to the IP internet[1].

Remark: Receiver driven flow control was investigated in [12] [14] [15] [17] and [23]. Several works on this area such as [12] propose the application of receiver driven rate control techniques for one to many transmission of media. In the media streaming framework, the receiver driven rate control allows clients with different connection capabilities to receive the same transmission with different levels of detail. Each receiver subscribes to as many multicast channels as possible in order to receive the highest possible quality of the received transmission. With this scheme, the server sends different information on each channel in a way that further details are incrementally supplied to those clients that are capable of receiving them. In contrast, for bulk data distribution, it is not acceptable for the rate control to affect the level of detail as in the media transmission case. Bulk data has to be received intact by all clients regardless of their reception rate

1. In fact, it could be argued that when a packet is targeted to a large multicast group it should not be regarded as a regular point to point packet since it is serving a much wider number of receivers.

capabilities. For this reason, the application of receiver driven techniques in the bulk data scenario involves a trade-off on the reception time as opposed to the reception quality of the media case. Instead of receiving a higher quality when subscribing to more channels, we desire the file reception time to be shortened for clients with higher reception rates.

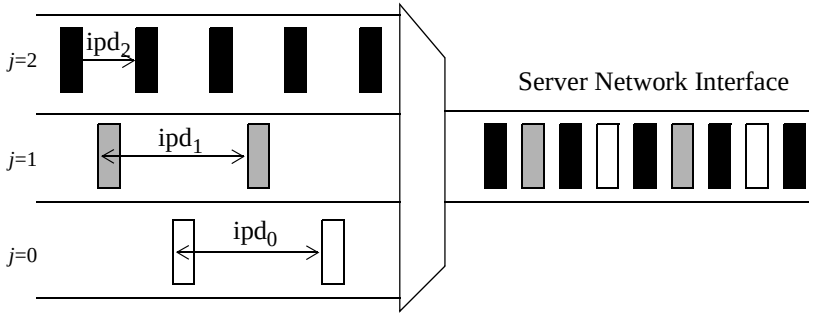
Summarizing, the multi-channel distribution of bulk data presented in this chapter is based on the receiver-driven selective subscription technique described so far. The rate for each of the multicast channels that belong to the same transmission, its packet schedule mechanism and the client subscription policy, are the three key components that define a proposed scheme. In the following sections we develop this three components to achieve our desired goals of minimum reception time and optimal utilization of network resources.

Notation and Implementation Aspects

The server transmits a file using multiple multicast ***Channels*** simultaneously as suggested above. In an IP internet context, every ***Channel*** is implemented using a single IP multicast address to which all packets belonging to the ***Channel*** are sent. Each ***Channel*** is denoted with a number: *j*.

The server sends the packets using a specific transmission rate for every channel. The transmission rate for channel *j* is denoted: R_j. In an IP internet context, desired transmission rates may be achieved by means of a specific inter-packet delay that is applied by the server on every channel. The channel is a logical concept meaning packets of all channels are sent out of the server through its (possibly only one) network interface as shown in Figure 4.1 on page 86. In this work we use channel rates that are all integer multiples of some base rate, for this reason, the required channel interleaving for emulation of concurrent transmissions at the respective rate is straightforward.

FIGURE 4.1 - Inter-packet delay for channel rate control



The choice of which data packets are to be sent on each channel at every point in time is defined by what we call a ***packet schedule mechanism*** which combined with the receiver's ***subscription policy*** constitute the determinant factor in the performance of the suggested approach.

Every receiver thus controls its reception rate by subscribing to one or more channels. Rate increases and drops are achieved by a receiver through changes to its subscription state. In an IP internet this may be implemented with regular multicast membership protocols such as IGMP [2].

Summarizing, in this chapter we are looking for:

- **Channel Rate Assignment:** Transmission rate (ipd) for each multicast channel
- **Packet Schedule Mechanism:** Which packet is sent on every channel at every point in time
- **Subscription Policy:** What are the rules used by a client to decide which channels should it subscribe to.

so that the reception time is minimized while network resources are optimally[2] utilized.

The Optimal Network Utilization Paradigm

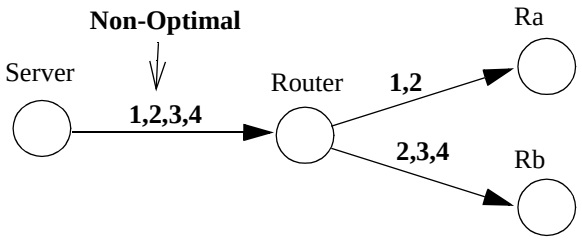
The receiver driven scheme suggested above, motivated by the problem of different clients rates, requires us to determine which packets will be transmitted at each channel. A very simple approach to the multi-layer schedule problem would be for the server to send simultaneously and over different channels, different parts of the file using the lowest network rate for each channel. In a previous chapter, while solving the packet loss problem, we divided the file into FEC groups for coding complexity reasons. A simple approach for the packet schedule could be to send the coded packets of each FEC group in a separate channel. A possible channel subscription policy suitable for this packet schedule could be for the client to join as many multicast channels as its reception rate allows and keep them until enough packets have been successfully received. Once finished receiving the group of the file at a specific channel, a client would unsubscribe from it and join other ones to continue until the whole file was received.

From a receiver's standpoint, this method sounds very appealing. If each FEC group is sent through a separate multicast channel, then by subscribing and unsubscribing once K packets have been successfully received, we achieve a scheme where the client perception is equivalent to that of the ideal selective retransmission case. However, precisely for the same reason as regular point-to-point reliable protocols are not suitable for the bulk data distribution problem, this simplistic approach is not adequate. The subscription/unsubscription mechanism takes the place of the selective retransmission feedback and has the similar effect of links overutilization and ultimately collapse of the network resources.

2. We define optimal network utilization in the next section

The problem of basically all the schedule mechanisms that send different parts of the file on different channels is that of network resource overutilization. The main motivation for using multicast in the first place was to reduce the amount of traffic on network links when more than one receiver downstream is interested in the same data. Two topologically close clients that are not coordinated in their subscription state result in non-optimal network bandwidth consumption.

FIGURE 4.2 - Non-Optimal Network Utilization with Multicast



The example in Figure 4.2 on page 88 shows a trivial case for non-optimal network resource utilization. The numbers on the links denote the channels that are being transmitted through it. Channels reaching a receiver are those for which it has subscribed. In the example, receiver *a* has subscribed to two channels and receiver *b* has requested three. Since they are not coordinated, the link connecting the server to the intermediate router is being filled with 4 channels worth of bandwidth.

When the file is divided so that different parts of it are sent through different multicast channels as in the simple scheme described above, non-optimal network utilization arises. This is a result of topologically close receivers which at a point in time subscribe to different channels. The non-coordinated subscription is caused by different reception rates that cause some client to advance before its neighbor, by clients starting reception at different points time and also by packet losses that affect some of the receiver but not the others.

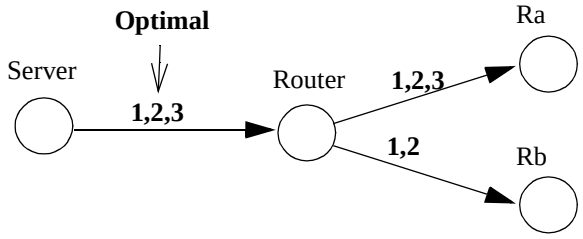
We desire our mechanism to be optimal in terms of network resources consumption.

Definition 4.1 - Optimal Network Utilization
We define a scheme to be of **optimal network utilization** when for every link in the multicast routing tree, there is at least one receiver downstream which is subscribed to all the channels that are being transmitted on the link.

The definition for **optimal network utilization** stems from the fact that when it holds it resembles a point to point connection between the server and the highest rate receiver downstream from that link. By achieving **optimal network utilization** as defined here, we obtain a scheme where the bandwidth consumed at each link is never higher than what would have been consumed if there was a single receiver downstream regard-

less of the number of clients simultaneously taking advantage of the transmission. Figure 4.2 on page 88 shows an example of **optimal network utilization**.

FIGURE 4.3 - Optimal Network Utilization with Multicast



Cumulative Subscription for Optimal Network Resource Utilization

In order to achieve **optimal network utilization** we will use a **cumulative subscription policy**. A **cumulative subscription policy** forces the heterogeneous clients to behave in a coordinated way regardless of their reception rate, experienced losses and starting point in time.

Definition 4.2 - Cumulative Subscription Policy
A **subscription policy** is called **cumulative** when in order to subscribe to channel *j*, a client has to subscribe to all channels *i* where $i<j$.

We prove in Theorem 4.1 - on page 89 that a **cumulative subscription policy** results in **optimal network utilization** as desired.

Theorem 4.1 - A Cumulative Subscription Policy Results in Optimal Network Resource Utilization

We merely need to prove that for every link in the multicast routing tree, there is at least one receiver downstream that subscribes to all the channels flowing through the link.

Proof:

Let us denote with *m* the highest channel flowing through the link. Since channel *m* is flowing through the link there is at least one receiver downstream that is subscribed to it. Let us denote one of these clients with *r*. Because of the **cumulative subscription policy**, *r* has to subscribe to all channels *i* where $i<m$. From here, all other channels flowing through the link are received by *r* and thus all channels flowing through the link are received by *r*.

qed

We define then the **subscription level** of a receiver to be the highest channel to which the receiver is currently subscribed. We denote the **subscription level** with *l*. When a cumulative subscription policy is used, the subscription level fully describes the subscription state of a receiver.

The **subscription rate** of a receiver is then mandated by its **subscription level** and is the sum of the rates of the channels from channel 0 up to and including channel *l*.

Having a cumulative subscription policy for optimal network utilization, we still need to assign transmission rates to the channels and define a **packet schedule** mechanism for each of them so that the transmission time is optimal from a receiver's standpoint. We develop this in the following sections.

4.2 - Exponential Cumulative Channels Scheme

Given a file of size *S*, we have proved in a previous chapter, that if no feedback is available and the best achievable FEC encoding scheme has parameters *K* and *N*, then the best transmission scheme from a receiver standpoint, is one where the file is partitioned into *G=S/K* groups, each group is encoded using FEC to obtain *N* packets and these encoded packets are transmitted using a cyclic group interleaving schedule.

Under the proposed receiver-driven rate control mechanism, each client subscribes to a subset of the multicast channels transmitted by the server. In this way, the packet stream received by each client is the union of the channels to which it subscribes. For the multi-channel mechanism to attain optimal results, we devise in this section a channel packet schedule that maintains the required interleaving properties of the combined received packet stream seen by each client. The real challenge stems from the fact that different clients subscribe to different channels and thus, for the results to be optimal for all clients, we need our scheme to maintain the required interleaving properties for all subscription levels simultaneously. Moreover, as it is also our goal to allow clients to join the transmission at any point in time, we build the packet schedule for the channels so that optimal results are attained regardless of the moment that a client started its reception.

As may be inferred from the presented requirements, a packet schedule for the channels, that satisfies all the presented conditions is not trivial. It is therefore reasonable to ask ourselves at this stage, whether or not is the building of such a precisely tailored packet schedule, justified by its expected benefits. The answer to this question is given in a previous chapter where we showed the **group interleaving** approach compared to the use of a random packet schedule. The results for the random are much worse than those of the proposed optimal scheme. Moreover, our scheme benefits from the bursty nature of losses which further enhances the difference in results as compared to a random schedule. As a closing remark on this matter, the scheme we present below satisfies all requirement while its implementation is trivial from a server's perspective so all its great benefits are achieved at no extra cost[3].

Exponential Channel Rates

We will set the transmission rate for the first channel (j=0) to be equal to a base rate denoted B. The rate for the subsequent channels R_j will be set to be equal to the sum of the rates of all previous ones. We call this channel rate assignment **exponential** and define it formally as follows:

Definition 4.3 - Exponential Channels
Channels are called exponential when their transmission rate R_j is defined as:

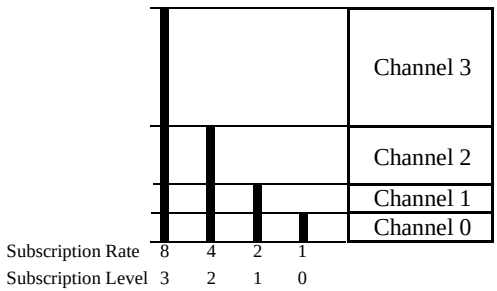
(EQ 1)

$$R_j = \begin{cases} B & j = 0 \\ \sum_{i=0}^{j-1} R_i & j > 0 \end{cases}$$

Note that a straightforward property of exponential channels is that the rate of all channels j>1 are double the rate of channel j-1.

Figure 4.4 on page 91 shows a schematic abstraction of the exponential channels and the corresponding rates achieved for every subscription level when a **cumulative** subscription policy is applied.

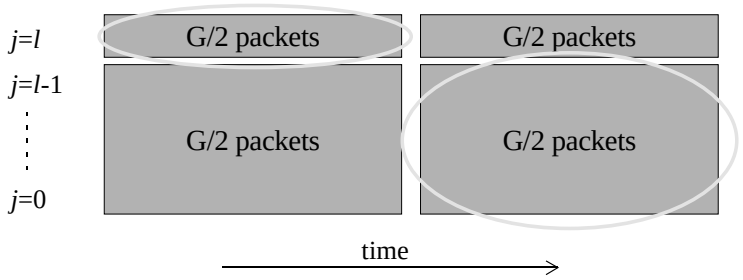
FIGURE 4.4 - Cumulative Exponential Channels



3. It may be even argued that a random schedule implementation may be more demanding than the optimal one that we are proposing.

Figure 4.5 on page 92 provides some motivation for using exponential channels in our schedule scheme. Basically, the channel schedule for optimal results is built by starting from the lowest rate channel (j=0) and sequentially introducing additional ones while taking care to keep the interleaving properties of the cumulative stream. Let us assume that up to level l-1 the schedule achieves the required interleaving properties. This means that out of G consecutive packets transmitted jointly over channels 0 through l-1 each belongs to a different FEC group. When channel l is added, the time it takes to transmit G packets in channels 0 through l is by definition shorter. By setting the rate in channel l to the sum of the rates of channels 0.. l-1, the time for G packets at level l becomes exactly half the time for G packets at level l-1. Then, in order to maintain the group interleaving property we schedule the packets on the second half of the G packets at level l-1 at channel l as shown in the figure. Consequently, the groups to which the packets in the first half of the time for G packets at level l-1 belong, will be scheduled in channel l for the time immediately following. The rate doubling resulted in exactly two times G packets being sent in the same time it took to send G packets before the addition of channel l. This exact doubling makes the packet ordering for interleaving tractable. The Details will be further understood after the forecoming sections.

FIGURE 4.5 - Motivation for exponential channels



Developing the Cumulative Exponential Packet Schedule

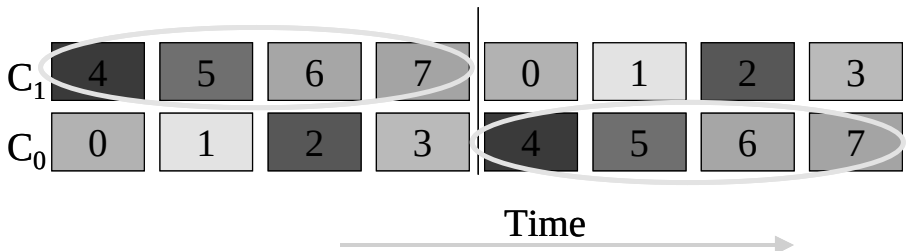
We have defined so far our **subscription policy** to be **cumulative** and assigned **exponential** rates to our channels. In this section we present the **packet schedule** for each channel that satisfies the requirements of:

- **Group Interleaving** for all subscription levels: The packet stream is such that out of any G consecutive packets each belongs to a different FEC group.
- **Packet Interleaving** for all subscription levels: The packet stream is such that any N consecutive packets that belong to the same FEC group (sent at an interval of G packets one from the other because of the group interleaving property) are all distinct.
- **Invariance to starting time**: The **Group Interleaving Property** and the **Packet Interleaving Property** hold regardless of the starting time.

Remark: The purpose of these three properties is to preserve in the multi-channel scenario the characteristics of the packet stream perceived by a client, regardless of its own rate and starting time. This is to obtain the same results that we proved to be optimal (and comparable to those of an ideal reliable unicast) in the previous chapter.

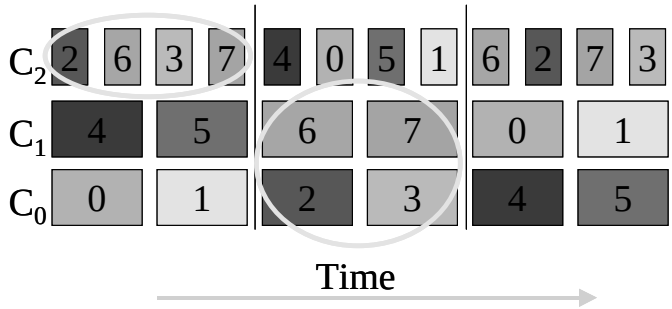
The intuition behind the derivation of the packet schedule formulas presented below is somehow related to the observations made above with regards to Figure 4.5 on page 92. Let us start, for the sake of simplicity, by considering only the group interleaving property. Since receivers at subscription level 0 receive only the packets sent on channel 0, it is clear then that packets in channel 0 must be scheduled so that every packet out of *G* consecutive ones belongs to a different group. Without loss of generality we define then our schedule to send packets from group 0 to *G*-1 and so on in a cyclic manner on channel 0. When considering now the clients at subscription level 1 we face the (still simple) problem of setting the packet schedule of channel 1 so that when combined with channel 0 the aggregated packet stream still complies to the group interleaving rule. For this we need to schedule the same packets as in channel 0 but with a "phase" offset of *G/2*. We keep the same packet order so that no matter the starting time, the group interleaving conditions are preserved. Figure 4.6 on page 93 shows this group index assignment for the first two channels when *G* is 8. The number within the blocks denotes the group to which the packet belongs.

FIGURE 4.6 - Assigning Group Index in the Packet Schedule - Channels 0 and 1



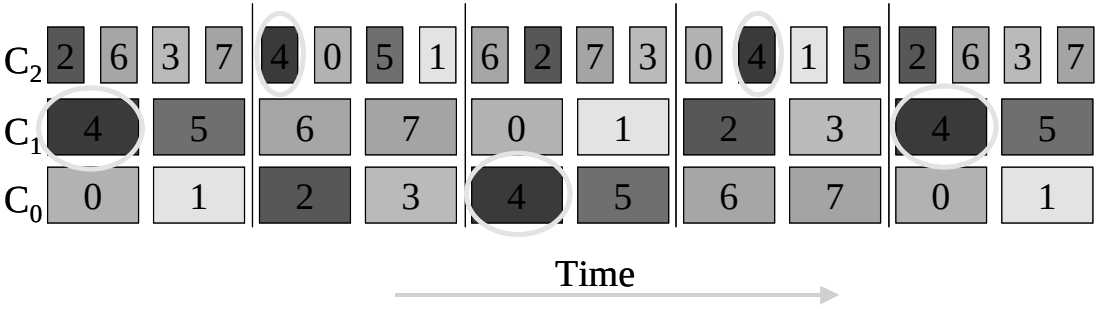
The building of subsequent channels is based on the same principle as can be inferred from Figure 4.7 on page 94. However the problem becomes trickier as the rate is increased and further channels are added. In contrast to what may have been assumed, consecutive periods of G packets at channels higher than 1 do not look the same. See channel 2 in Figure 4.7 on page 94. The resulting period for each channel *j* is not *G* but $G2^{j-1}$ this being a consequence of the interleaving requirements.

FIGURE 4.7 - Assigning Group Index in the Packet Schedule - Channel 2



The determination of the packet index within the selected group follows a similar process. The goal is to assign packet indexes so that two consecutive packets from the same group transmitted at any subscription level are different. As an example, Figure 4.8 on page 94 shows the consecutive packets that belong to group 4 at subscription level 2. For the schedule to attain the desired results, all these packets need to have a different packet index.

FIGURE 4.8 - Assigning Packet Index in the Packet Schedule



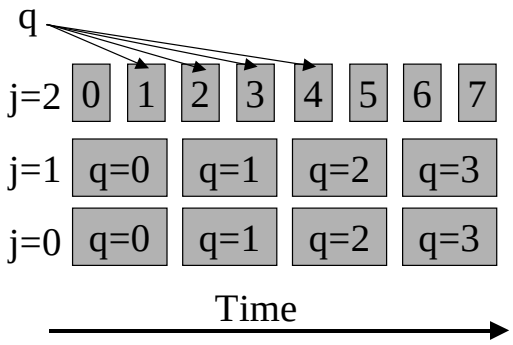
The group interleaved stream is then looked-at at a “lower resolution” meaning that groups of *G* packets are regarded as a block for this purpose. The same packet index is then assigned to all the packets within each of these blocks (which are guaranteed by the properties of the group index formula to belong to distinct groups) so that the packet interleaving property is attained using the same guidelines as above but for blocks instead of individual packets. Further insights into the building process of the packet schedule are provided within the proofs of its properties in sections to follow.

The Packet Schedule Formulas

Remark: The packet schedule formulas presented below together with the proofs of their properties in the next sections are one of the main contributions of the presented work.

In Figure 4.5 on page 92 we show the notation for the packet schedule formulas. Channels are denoted with *j* and packet position within each channel with *q*.

FIGURE 4.9 - Packet Schedule Parameters



Definition 4.4 - Cumulative Exponential Multi-channel Packet-Schedule
The packet in position *q* in channel *j* is from group:

(EQ 2)

$$g = \left| \left\lfloor \frac{q}{2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{G}{2^{j}} \right\rfloor + \max\left(1, \left\lfloor \frac{G}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left| q \right|_{2^{\max(0,j-1)}} \right|_{G}$$

and its packet index (out of the *N* coded packets in each group *g*) is given by:

(EQ 3)

$$p = \left| \left\lfloor \frac{q}{G \cdot 2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{N}{2^{j}} \right\rfloor + \max\left(1, \left\lfloor \frac{N}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left\| \left\lfloor \frac{q}{G} \right\rfloor \right|_{2^{\max(0,j-1)}} \right|_{N}$$

In the next few sections we prove the properties of the **"Cumulative Exponential Multi-channel Packet-Schedule" on page 95**. We show that for any starting time and any subscription level, the packet stream received by any client (which is the union of the channels to which it is subscribed) has the group and packet interleaving properties proved optimal in the previous chapter. In "Group Interleaving Properties" on page 96 we prove that at any subscription level, the aggregated packet stream is such that any given consecutive G packets belong each to one of the G FEC groups. In "Packet Index Interleaving Properties" on page 112 we prove that out of the packets received from each group, there are no repetitions of the same coded packet until all the different ones have been sent.

With this result we will have shown a mechanism with optimal network utilization (because of the cumulative subscription policy) that is also optimal from the standpoint of all receivers regardless of their subscription rate and their starting point in time.

4.3 - Group Interleaving Properties

The Group Interleaving Theorem

As we have shown in a previous chapter, for a schedule to be optimal from a receiver standpoint, packets have to be transmitted in a group interleaved fashion. We prove that this feature is accomplished by our proposed packet schedule for concurrently-active receivers at different subscription levels even when each started the file reception at a different point in time.

Let $G=2^J$ (the number of FEC groups into which the file was partitioned)[4]. We wish to prove then, that for every subscription level

(EQ 4)

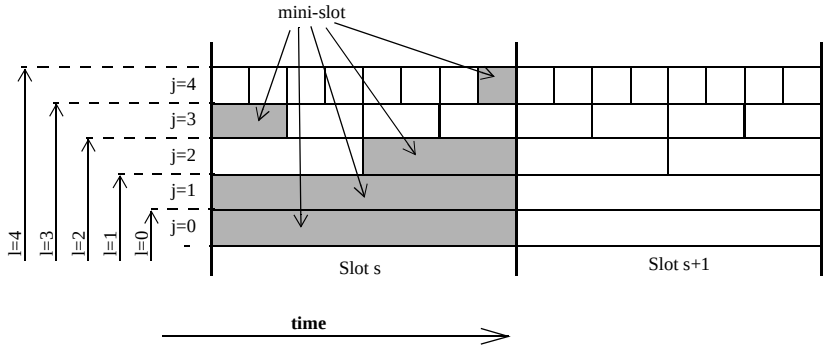
$$0 \le l \le J$$

if we take any consecutive G packets that were transmitted starting at any slot boundary, then each one of these packets belongs to a different group g. We make some definitions and prove some needed lemmas before proving the desired property in Theorem 4.3 - on page 104.

4. This property is proved here for G that is a power of 2. In a future section we prove this property for the generalized case.

We define a **slot** to be the time that takes to transmit a packet on the slowest channel. We number the slots starting from 0 and denote the slot number with *s*.

FIGURE 4.10 - slot and mini-slot definition



We define a ***j*-mini-slot** to be the time that takes to transmit a packet on channel *j*.

From the definitions above, there are 2^{j-1} *j*-mini-slots per slot in channel *j (0<j<=l)*. And a single *j*-mini-slot in channel 0. We number the 2^{j-1} *j*-mini-slots within a slot in channel *j* from 0 and denote the mini-slot number with:

(EQ 5)

$$t \in \{0,1,\cdots,2^{j-1}-1\} \quad j>0$$

and

(EQ 6)

$$t \in \{0\} \quad j=0$$

Clearly, as shown in Figure 4.11 on page 98, *q* (the packet position within a channel as defined above) can be written as:

(EQ 7)

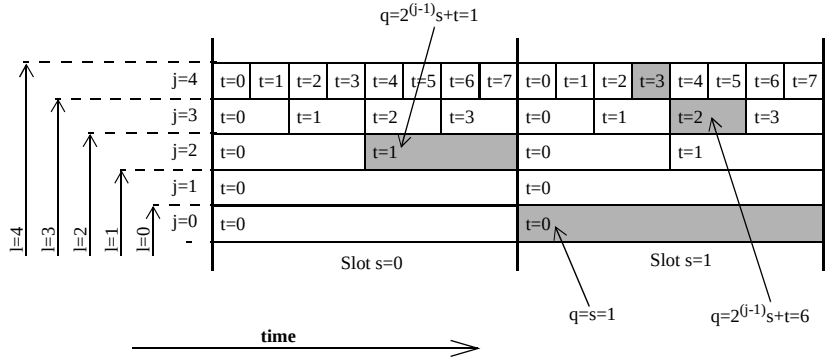
$$q=2^{j-1}s+t \quad j>0$$

and

(EQ 8)

$$q = s \quad j = 0$$

FIGURE 4.11 - packet position (q) as a function of slot (s) and mini-slot (t)



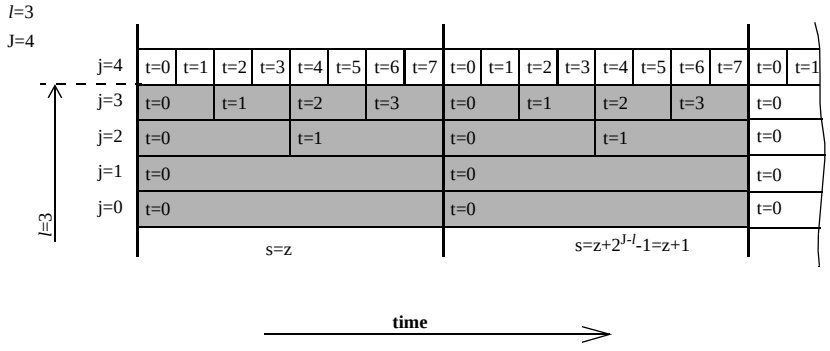
We now calculate the number of slots needed for the transmission of *G* packets at subscription level *l*.

Lemma 4.1 - Number of Slots needed for G packets at Level *l*

The number of slots required to transmit exactly *G* packets at level *l* is 2^{J-l}.

Proof: given in Appendix A on page 197.

FIGURE 4.12 - Slots for *G* packets at level *l*



Using the defined *s* and *t*, we can now rewrite the expression for the group index as:

(EQ 9)

$$\begin{aligned} \underset{j=0}{g} &= \left\| \lfloor q \rfloor + 2^J + 2^J \cdot \left| q \right|_{2^0} \right|_G = \\ &= \left\| \lfloor q \rfloor + G + G \cdot q \right|_G = \left| q \right|_G = \left| s \right|_G \end{aligned}$$

and:

(EQ 10)

$$\begin{aligned} \underset{0<j\le l}{g} &= \left\| \left\lfloor \frac{q}{2^{j-1}} \right\rfloor + 2^{J-j} + 2^{J-j+1} \cdot \left| q \right|_{2^{j-1}} \right|_G = \\ &= \left\| \left\lfloor \frac{2^{j-1}s + t}{2^{j-1}} \right\rfloor + 2^{J-j} + 2^{J-j+1} \cdot \left| 2^{j-1}s + t \right|_{2^{j-1}} \right|_G \overset{t<2^{j-1}}{=} \\ &= \left| s + 2^{J-j} + 2^{J-j+1} \cdot t \right|_G \end{aligned}$$

We aim to prove that the *G* packets in the shaded region of Figure 4.12 on page 98 include exactly one packet per group. In other words, we prove that if we start receiving packets at a slot boundary, then we will not see a packet of the same group for a second time before we have received one packet from all different groups once. Let *z* be the starting slot number as depicted in Figure 4.12 on page 98. We first prove this property using slot 0 as the starting point (*z*=0) in Theorem 4.2 - on page 104. We extend the proof to the general case (any *z*) in Theorem 4.3 - on page 104.

Lemma 4.2 - - Modulo Elimination

Starting at slot zero, when looking at *G* consecutive packets, the modulo in the group index formula (Eq. 9 on page 99) and (Eq. 10 on page 99) can be eliminated. Therefore, the group index formula can be rewritten as follows.

Given *z*=0 then:

(EQ 11)

$$\underset{j=0}{g} = s$$

(EQ 12)

$$\underset{0< j\le l}{g} = s + 2^{J-j} + 2^{J-j+1} \cdot t$$

Proof: given in Appendix A on page 198.

We aim to show now that given any two packets in the allowed range (of *s*, *j* and *t*) for *G* packets, these belong to different groups. For this purpose we use a binary representation of the group indexes and prove below by comparing its binary coefficients.

Given $a<2^J$ a natural number. The binary representation of a is defined as:

(EQ 13)

$$b(a,i) \in \{0,1\} \quad i \in \{0,1,\cdots J-1\}$$

such that:

(EQ 14)

$$a = \sum_{i=0}^{J-1} b(a,i) \cdot 2^i$$

it is a known fact that this representation exists and is unique for all *a* as defined above.

Figure 4.13 on page 100 and Figure 4.14 on page 101 show schematic views of the binary representation of *g* for *j>0* and *j=0* respectively.

FIGURE 4.13 - Binary representation of *g* (*j*>0)

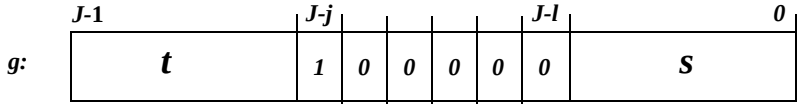
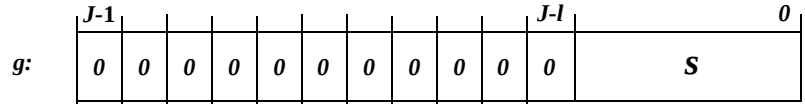


FIGURE 4.14 - Binary representation of *g* (*j*=0)



To further emphasize the effect of each of the representation parameters (*s*,*j* and *t*) in the binary representation of *g* we show a few examples. In Figure 4.15 on page 101 we see the binary representation for a packet from group 2. For *G* equal to 64 and subscription level 3, the first packet from group 2 is sent on channel 0 on slot number 2. The figure shows the meaning of the *j*, *s* and *t* components of the representation. We see that this packet is sent in channel 0 from the fact there is no bit set beyond the range occupied by *s* in the binary representation.

FIGURE 4.15 - Binary representation (Example: *G*=64 *l*=3 *g*=2)

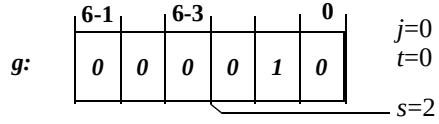
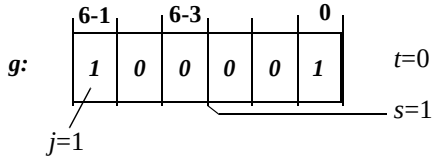


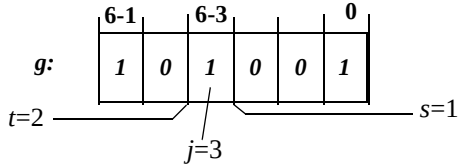
Figure 4.16 on page 101 shows a packet from channel 1. This is clearly identified by the fact that the first bit set beyond the range occupied by *s* is the MSb of the representation.

FIGURE 4.16 - Binary representation (Example: *G*=64 *l*=3 *g*=33)



Finally, in Figure 4.17 on page 101 we show a packet from channel 3.

FIGURE 4.17 - Binary representation (Example: *G*=64 *l*=3 *g*=41)



We derive now the values of the binary representation of *g* from the binary representation of its parameters *s, j* and *t*. For *j=0* we have:

(EQ 15)

$$g = s = \sum_{i=0}^{J-l-1} b(s,i) \cdot 2^i + \sum_{i=J-l}^{J-1} 0 \cdot 2^i$$

Clearly then:

(EQ 16)

$$b(g,i)_{j=0} = \begin{cases} b(s,i) & 0 \le i < J-l \\ 0 & J-l \le i < J \end{cases}$$

For *j>0* we have:

(EQ 17)

$$\begin{aligned} & g = s + 2^{J-j} + 2^{J-j+1} \cdot t = \\ & = \sum_{i=0}^{J-l-1} b(s,i) \cdot 2^i + \sum_{i=J-l}^{J-j-1} 0 \cdot 2^i + \sum_{i=J-j}^{J-j} 1 \cdot 2^i + 2^{J-j+1} \sum_{i=0}^{j-1} b(t,i) \cdot 2^i = \\ & = \sum_{i=0}^{J-l-1} b(s,i) \cdot 2^i + \sum_{i=J-l}^{J-j-1} 0 \cdot 2^i + \sum_{i=J-j}^{J-j} 1 \cdot 2^i + \sum_{i=0}^{j-1} b(t,i) \cdot 2^{J-j+1+i} = \\ & = \sum_{i=0}^{J-l-1} b(s,i) \cdot 2^i + \sum_{i=J-l}^{J-j-1} 0 \cdot 2^i + \sum_{i=J-j}^{J-j} 1 \cdot 2^i + \sum_{i=J-j+1}^{J-1} b(t,i-J+j-1) \cdot 2^i \end{aligned}$$

Clearly then:

(EQ 18)

$$b(g,i)_{0<j\le l} = \begin{cases} b(s,i) & 0 \le i < J-l \\ 0 & J-l \le i < J-j \\ 1 & i = J-j \\ b(t,i-J+j-1) & J-j < i < J \end{cases}$$

Now, using the presented binary representation of *g*, we prove the following lemma by comparing the binary coefficients of two packets with the same group index within the calculated range of *s*,*j* and *t*.

Lemma 4.3 - Uniqueness of the *j*,*s*,*t* representation of *g* within a range of *G* packets at any subscription level *l*.

For any group denoted with g_k, which can be represented as:

(EQ 19)

$$g_k = \begin{cases} s_k & j_k = 0 \\ s_k + 2^{J-j_k} + 2^{J-j_k+1} \cdot t_k & 0 < j_k \le l \end{cases}$$

with parameters in the range:

(EQ 20)

$$\begin{aligned} s_k &\in \{0,1,\cdots,2^{J-l}-1\} \\ j_k &\in \{0,1,\cdots,l\} \\ t_k &\in \{0,1,\cdots,2^{j_k-1}-1\} \quad j_k > 0 \\ t_k &\in \{0\} \quad j_k = 0 \end{aligned}$$

the representation above is unique which means that:

(EQ 21)

$$(g_1 = g_2) \rightarrow \begin{cases} s_1 = s_2 \\ j_1 = j_2 \\ t_1 = t_2 \end{cases}$$

Proof: given in Appendix A on page 200.

We can now prove the desired group interleaving property for starting slot 0 (*z*=0).

Theorem 4.2 - Group Interleaving when $G=2^J$ and Starting Slot *z* is 0.

For every subscription level *l*

(EQ 22)

$$0 \le l \le J$$

if we take any consecutive *G* packets that were transmitted starting at slot 0 (*z*=0), then each one of these packets belongs to a different group *g*, where $G=2^J$.

Proof:

We have shown that in the range defined by

(EQ 23)

$$\begin{aligned} s &\in \left\{0,1,\cdots,2^{J-l}-1\right\} \\ j &\in \left\{0,1,\cdots,l\right\} \\ t &\in \left\{0,1,\cdots,2^{j-1}-1\right\} \quad j>0 \\ t &\in \left\{0\right\} \quad j=0 \end{aligned}$$

there are exactly *G* packets (Lemma 4.1 - on page 98) and that all those packets belong to different groups (Lemma 4.3 - on page 103). We thus have proved that we have sent one packet from each group.

qed

The extension to any starting slot z follows

Theorem 4.3 - Group Interleaving when $G=2^J$ (for any Starting Slot).

For every subscription level *l*

(EQ 24)

$$0 \le l \le J$$

if we take any consecutive *G* packets that were transmitted starting at a **any** slot boundary (z), then each one of these packets belongs to a different group *g*, where $G=2^J$.

Proof:

When starting at a time slot *z* other than 0 the domain for s in our representation becomes:

(EQ 25)

$$s \in \left\{ z, z+1, \cdots, z+2^{J-l}-1 \right\}$$

Let us define *g#* to be *g-z*. Then for *g#* the proof for Theorem 4.2 - on page 104 holds and there is exactly one packet of each group within the *G* packets defined by the parameters domain. Clearly by adding the same *z* to every *g#* after applying modulo *G* we still have *G* different groups.

qed

Generalized G (not a power of 2)

The Group Interleaving Theorem was proved in Theorem 4.3 - on page 104 for *G* that is a power of two ($G=2^J$). This requirement may be somewhat restrictive in practical cases.

The file size in bytes is equal to *K* times *G* times the packet size in bytes. In actual implementations, we expect *K* (the number of data packets per FEC group) to be fixed (hard-coded in the implementation). As for the packet size, playing with it may involve intervention on lower layers of the transmission protocol which is highly undesired. We see then that *G* is mandated by the file size so restricting it to powers of 2 poses a high limitation. We show here that our schedule conserves the optimal group interleaving properties for any *G*.

In this section we generalize the Group Interleaving Theorem (Theorem 4.3 - on page 104) for a wider class of values of G. We wish to prove that our schedule scheme is an optimal packet schedule for any $G=W2^J$ for all subscription levels *l* up to (and including *J*) where *J* is an arbitrary non-negative integer number and *W* is an arbitrary odd integer number.

We follow a similar approach to the one we used for the proof above. We need to prove that at any subscription level *l* (smaller than or equal to *J*) for any *G* consecutive packets starting at a slot boundary, the packets belong to different FEC groups.

Using the same **slot** and ***j*-minislot** definitions as above, the number of slots needed to receive *G* packets is now given by Lemma 4.4 - on page 106

Lemma 4.4 - Number of Slots needed for G packets at Level *l* for Generalized G

The number of slots required to receive exactly *G* packets under subscription level *l* is $W2^{J-l}$

Proof: The proof is the same as for Lemma 4.1 - on page 98.

We showed already that:

(EQ 26)

$$\underset{j=0}{g} = \left|s\right|_G$$

and from the new definition of *G* follows:

(EQ 27)

$$\underset{0<j\le l}{g} = \left|s + W2^{J-j} + W2^{J-j+1}\cdot t\right|_G$$

We prove first in Theorem 4.4 - on page 111 the group interleaving property for slot 0 as the starting point. The results are extended to any starting slot in Theorem 4.5 - on page 111.

Following the results from Lemma 4.4 - on page 106 the range for *s* is now:

(EQ 28)

$$s \in \left\{0,1,\cdots,W2^{J-l}-1\right\}$$

Lemma 4.5 - Modulo Elimination (Generalized G)

In a similar manner as in Lemma 4.2 - on page 99, we wish to eliminate the modulo in the group index expression. We prove here that when the starting slot is 0, for *G* consecutive packets, the modulo in (Eq. 26 on page 106) and (Eq. 27 on page 106) can be eliminated. Therefore, the group index formula can be rewritten as follows.

(EQ 29)

$$\underset{j=0}{g} = s$$

(EQ 30)

$$\underset{0<j\le l}{g} = s + W2^{J-j} + W2^{J-j+1} \cdot t$$

Proof: given in Appendix A on page 203.

We show now that there are no two packets belonging to the same group within the range defined above that contains *G* packets and therefore the *G* different packets ought to be one from each group.

Our proof is valid for any *l* in the allowed range. Without loss of generality, once we pick one specific value for *l*, $W2^{J-l}$ is a constant in our proof. It is therefore equivalent to prove that the possible values of:

(EQ 31)

$$g^*(s,j,t) = \frac{g(s,j,t)}{W2^{J-l}}$$

are all different within the allowed range of parameters (that spans *G* values as proved by Lemma 4.4 - on page 106).

For the proof we express *g** as a sum of its integer and non integer parts (the non integer part is denoted *r(g*)*).

For *j*=0, the integer part of *g** is:

(EQ 32)

$$\left\lfloor \underset{j=0}{g^*(s,j,t)} \right\rfloor = \left\lfloor \frac{s}{W2^{J-l}} \right\rfloor \overset{s<W2^{J-l}}{=} 0$$

and for *j*>0:

(EQ 33)

$$\underset{j>0}{\lfloor g^*(s,j,t)\rfloor}=\left\lfloor \frac{s}{W2^{J-l}}+\frac{W2^{J-j}}{W2^{J-l}}+\frac{W2^{J-j+1}t}{W2^{J-l}}\right\rfloor=$$

$$=\left\lfloor \frac{s}{W2^{J-l}}+2^{l-j}+2^{l-j+1}t\right\rfloor \overset{l\geq j}{=}\left\lfloor \frac{s}{W2^{J-l}}\right\rfloor+2^{l-j}+2^{l-j+1}t \overset{s<W2^{J-l}}{=}$$

$$=2^{l-j}+2^{l-j+1}t$$

Let us define:

(EQ 34)

$$r(g^*)=g^*(s,j,t)-\lfloor g^*(s,j,t)\rfloor$$

which is the non-integer part of g^*.

Clearly:

(EQ 35)

$$\underset{j=0}{r(g^*)}=g^*(s,j,t)-\lfloor g^*(s,j,t)\rfloor=$$

$$\frac{s}{W2^{J-l}}-0=\frac{s}{W2^{J-l}}$$

and:

(EQ 36)

$$\underset{j>0}{r(g^*)}=g^*(s,j,t)-\lfloor g^*(s,j,t)\rfloor=$$

$$=\frac{s}{W2^{J-l}}+2^{l-j}+2^{l-j+1}t-2^{l-j}-2^{l-j+1}t=\frac{s}{W2^{J-l}}$$

We look now at the non-integer binary representation of g^* which is graphically shown in Figure 4.18 on page 109 and Figure 4.19 on page 109 for $j>0$ and $j=0$ respectively.

FIGURE 4.18 - Binary representation of g^* (j>0)

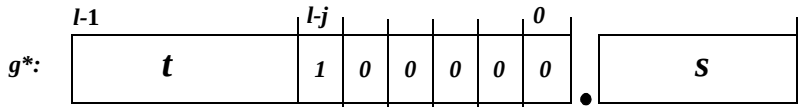
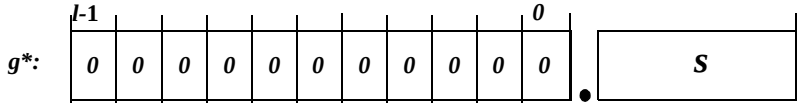


FIGURE 4.19 - Binary representation of g^* (j=0)



As an example, in Figure 4.20 on page 109 we see the representation for a packet from group 2. For G equal to 48 (J=4, W=3) and subscription level 3.

(EQ 37)

$$g^*(s,j,t)=\frac{g(s,j,t)}{W2^{J-l}}=\frac{2}{3\cdot 2^{4-3}}=\frac{2}{6}$$

The figure shows the meaning of the j, s and t components of the representation. We see that this packet is sent in channel 0 from the fact there is no bit set beyond the range occupied by s in the binary representation.

FIGURE 4.20 - Binary representation Example: G=48 (J=4, W=3), l=3 and g=2

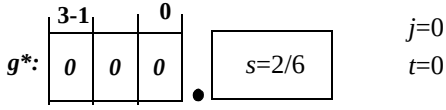


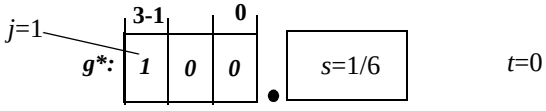
Figure 4.21 on page 110 shows the representation for g=25. In this case:

(EQ 38)

$$g^*(s,j,t)=\frac{g(s,j,t)}{W2^{J-l}}=\frac{25}{3\cdot 2^{4-3}}=4+\frac{1}{6}$$

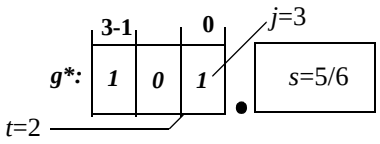
As we see, the packet is from channel 1. This is clearly identified by the fact that the first bit set beyond the range occupied by *s* is the MSb of the representation.

FIGURE 4.21 - Binary representation Example: *G*=48 (*J*=4, *W*=3), *l*=3 and *g*=25



Finally, in Figure 4.22 on page 110 we see an example for *g*=35.Which turns to be a packet from channel 3.

FIGURE 4.22 - Binary representation Example: *G*=48 (*J*=4, *W*=3), *l*=3 and *g*=35



Using the presented binary representation, we prove the following lemma by comparing the binary coefficients and the non-integer parts of two group indexes within the allowed range of parameters *s,j* and *t*.

Lemma 4.6 - Uniqueness of the *j,s,t* representation of *g within a range of *G* packets at any subscription level *l*.**

If $g^*(s_1,j_1,t_1)=g^*(s_2,j_2,t_2)$ then $s_1=s_2$, $j_1=j_2$ and $t_1=t_2$.

Proof: given in Appendix A on page 205.

We can now prove the desired property:

Theorem 4.4 - Group Interleaving when $G=W2^J$ and Starting Slot *z* is 0.

For every subscription level *l*

(EQ 39)

$$0 \le l \le J$$

if we take any consecutive *G* packets that were transmitted starting at slot 0 (*z*=0), then each one of these packets belongs to a different group *g*, where $G=W2^J$.

Proof:

We have shown that in the range defined by

(EQ 40)

$$\begin{aligned} s &\in \left\{0,1,\cdots,W\,2^{J-l}-1\right\} \\ j &\in \left\{0,1,\cdots,l\right\} \\ t &\in \left\{0,1,\cdots,2^{j-1}-1\right\} \quad j>0 \\ t &\in \left\{0\right\} \quad j=0 \end{aligned}$$

there are exactly *G* packets and that all those packets belong to different groups (Lemma 4.6 - on page 110). We thus have proved that we have sent each group exactly once.

qed

The extension to any starting slot *z* follows:

Theorem 4.5 - - Group Interleaving when $G=W2^J$ (for any Starting Slot).

For every subscription level *l*

(EQ 41)

$$0 \le l \le J$$

if we take any consecutive G packets that were transmitted starting at a **any** slot boundary (z), then each one of these packets belongs to a different group g, where $G=W2^J$.

Proof:

When starting at a time slot z other than 0 the domain for s in our representation becomes:

(EQ 42)

$$s \in \left\{ z, z+1, \cdots, z+W2^{J-l}-1 \right\}$$

Let us define *g#* to be *g-z*. Then for *g#* our proof above holds and there is exactly one packet of each group within the G packets defined by the parameters domain. Clearly by adding the same z to every *g#* after applying modulo G we still have G different groups.

qed

We have thus proved the group interleaving property of the proposed "Cumulative Exponential Multi-channel Packet-Schedule" on page 95 for any $G=W2^J$ (any file size), any subscription level[5] l up to and including J, and any starting time.

4.4 - Packet Index Interleaving Properties

The Packet Interleaving Theorem

In Theorem 4.5 - on page 111 we have proved that groups are interleaved in our schedule scheme as required for optimal results from a client's perspective. If the FEC encoding would have been capable of generating an unlimited number of coded packets per group, then the packet selection within each group would have been trivial. For each channel, every time that a specific group is scheduled, we could have picked a different packet (since we are assuming an unlimited supply of them). In such a case, since all packets are different, this would guarantee different packets at any subscription level as desired. However, as we have already seen, practical implementations of publicly available codes cannot provide an unlimited number of coded packets. Therefore, for optimal reception time, the schedule needs to make sure that the packet streams at all

5. We prove in a future section optimal interleaving for $l=J+1$ and show near optimal results for higher subscription levels.

subscription levels are such that all available coded packets are sent before any one of them is sent again (and so on). The fact that this property has to be maintained for all subscription levels simultaneously makes the solution non-trivial.

We have already proved that if this property is attained, such a schedule scheme behaves as close as possible to the ideal case for which there are infinite packets per group. We showed that when a finite amount of packets are scheduled in a way that each client never sees the same packet again unless he has already seen all the packets, then the results are the closest to the infinite coded packets case. The proof for this was based on the trivial observation that it is obviously better for a client to see a packet he has not yet seen than to receive one he might already have achieved (which would not bring him closer to completion). Moreover, as we will show in future sections, if this packet interleaving property is attained, for virtually all practical cases, the file reception will be completed by the receiver long before the same packet is scheduled again in its subscribed channels. We will conclude then that from the client's perspective, the situation is equivalent to the case where there are infinite code packets per group.

We wish to prove then that: Given $G=W2^J$ (the number of FEC groups into which the file was divided for encoding) and $N=2^M$ (the number of code packets per FEC group), then for every subscription level l

(EQ 43)

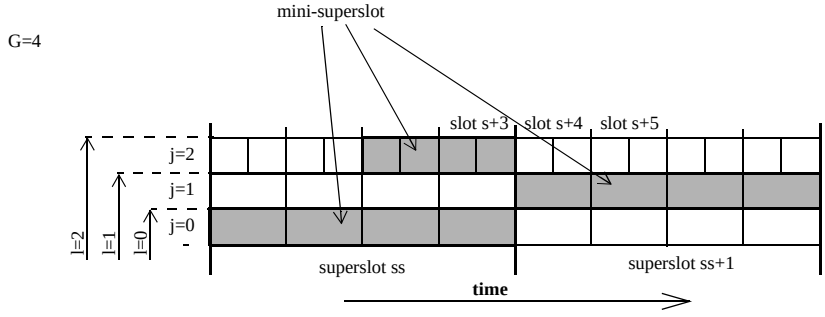
$$0 \le l \le \min(J, M)$$

each NG consecutively scheduled packets are all different ones.

Before the proof which is given in Theorem 4.6 - on page 118 we need the following definitions and auxiliary Lemmas.

We define a **superslot** to be the time that takes to transmit G packets on the slowest channel. We number the **superslots** starting from 0 and denote the **superslot** number with *ss*.

FIGURE 4.23 - superslot and mini-superslot definition



We define a ***j*-mini-superslot** to be the time that takes to transmit *G* packets on channel *j*.

From the definitions above, there are 2^{j-1} *j*-mini-superslots per superslot in channel *j (0<j<=l)*. And a single *j*-mini-superslot in channel 0. We number the 2^{j-1} *j*-mini-superslots within a superslot in channel *j* from 0 and denote the mini-superslot number with:

(EQ 44)

$$tt \in \{0,1,\cdots,2^{j-1}-1\} \quad j>0$$

and

(EQ 45)

$$tt \in \{0\} \quad j=0$$

As defined, there are *G* packets per *j*-mini-superslot for all *j* in the allowed range. Let use denote each of these with:

(EQ 46)

$$gg \in \{0,1,\cdots,G-1\} \quad 0 \le j \le l$$

As depicted in Figure 4.24 on page 115, clearly *q* (the packet position within a channel) can be written as:

(EQ 47)

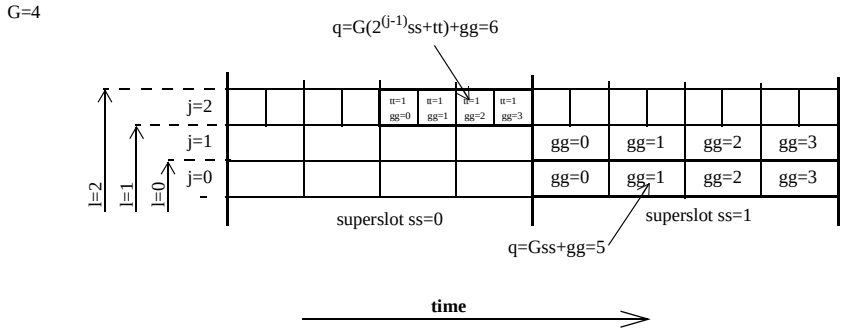
$$q = G(2^{j-1}ss + tt) + gg \quad j>0$$

and:

(EQ 48)

$$q = G \cdot ss + gg \quad j=0$$

FIGURE 4.24 - packet position (*q*) as a function of *ss*, *tt* and *gg*



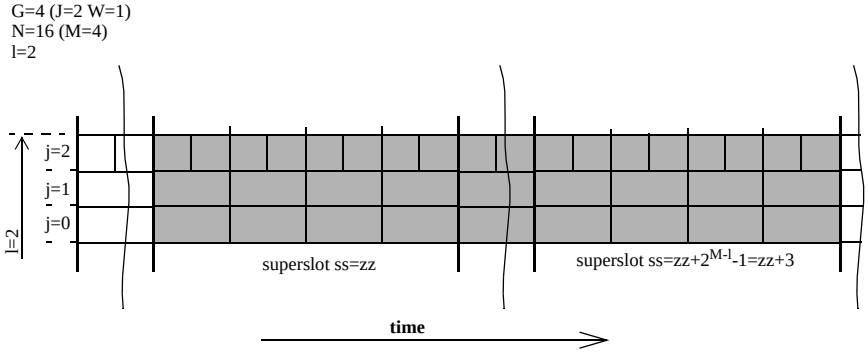
We start by calculating the number of superslots needed to transmit *GN* packets at level *l*.

Lemma 4.7 - Number of superslots for *GN* packets

The number of superslots required to receive exactly *GN* packets is 2^{M-l}.

Proof: given in Appendix A on page 209.

FIGURE 4.25 - superslots for *NG* packets at level *l*



We can then rewrite the packet index equation (Eq. 3 on page 95) for $j > 0$ as:

(EQ 49)

$$p = \left\| \left\lfloor \frac{q}{G \cdot 2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{N}{2^j} \right\rfloor + \max\left(1, \frac{N}{2^{\max(0,j-1)}}\right) \cdot \left\| \left\lfloor \frac{q}{G} \right\rfloor \right\|_{2^{\max(0,j-1)}} \right\|_N =$$
$$\left\| \left\lfloor \frac{G\left(2^{j-1} ss + tt\right) + gg}{G \cdot 2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left\| \left\lfloor \frac{G\left(2^{j-1} ss + tt\right) + gg}{G} \right\rfloor \right\|_{2^{j-1}} \right\|_N$$

and for $j=0$ as:

(EQ 50)

$$p = \left\| \left\lfloor \frac{q}{G \cdot 2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{N}{2^j} \right\rfloor + \max\left(1, \frac{N}{2^{\max(0,j-1)}}\right) \cdot \left\| \left\lfloor \frac{q}{G} \right\rfloor \right\|_{2^{\max(0,j-1)}} \right\|_N =$$
$$\left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor + N + N \cdot \left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor \right\|_{2^0} \right\|_N =$$
$$= \left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor \right\|_N$$

We aim to prove that the GN packets in the shaded region of Figure 4.25 on page 115 include exactly one instance of each packet index per group. In other words, we prove that if we start receiving packets at a superslot boundary, then we will not see the same packet for a second time before we have received all the possible different packets once. We prove this property for every subscription level l for which:

(EQ 51)

$$0 \le l \le \min(J, M)$$

The proof is based on considering a j-mini-superslot (which contains G consecutive packets in a channel) as a block for which a single packet index is assigned (same one to all packets in the block). We first show in Lemma 4.8 - on page 117 that this is the case. The motivation is then to regard the j-mini-superslots as if they were “packets” to which packet indexes were assigned using the same technique as for the group interleaving property to hold. We do a parameter transformation in Lemma 4.9 - on page 117 that enables us to use the group interleaving theorem (Theorem 4.5 - on page 111) that shows that the packet index assigned to each j-mini-superslot is such that out of N consecutive j-mini-superslots, each is assigned a different packet index. Finally we yet need to show that the G packets within each j-mini-superslot (that received the same

packet index) belong to G distinct groups. This is done in Lemma 4.11 - on page 118 after proving Lemma 4.10 - on page 118.

Lemma 4.8 - All packets have the same packet index within a *j*-mini-superslot.

For any specific channel *j*, *gg* can be eliminated from the packet index formula within a *j*-mini-superslot which means that all packets within the *j*-mini-super slot have the same packet index. What we wish to prove is formally expressed in (Eq. 52 on page 117) and (Eq. 53 on page 117).

(EQ 52)

$$j > 0 \quad gg \in \{0,1,\cdots,G-1\}$$
$$p = \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G \cdot 2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G} \right\rfloor \right\|_{2^{j-1}} \right|_N =$$
$$= \left\| \left\lfloor \frac{2^{j-1}ss + tt}{2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left| 2^{j-1}ss + tt \right|_{2^{j-1}} \right|_N$$

(EQ 53)

$$j = 0 \quad gg \in \{0,1,\cdots,G-1\}$$
$$p = \left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor \right\|_N = |ss|_N$$

Proof: given in Appendix A on page 209.

We prove now using Lemma 4.8 - on page 117 and Theorem 4.5 - on page 111 the following Lemma:

Lemma 4.9 - Each packet index is sent exactly *G* times (and in the same *j*-mini-superslot) during *GN* packets starting at a superslot boundary.

We wish to prove that out of *GN* packets sent at level *l*, starting at a superslot boundary, every packet index is sent exactly *G* times and that these *G* times are within the same single *j*-mini-superslot[6].

Proof: given in Appendix A on page 211.

6. since the G times are within the same mini-superslot then it means they are all in the same channel (since each mini-superslot belongs to one of the channels in the level)

We must still prove that these G occurrences of a packet index correspond to different groups. We do so with the help of the next two Lemmas.

Lemma 4.10 - Every *j*-mini-superslot begins at a slot boundary

Proof: given in Appendix A on page 212.

Using Lemma 4.10 - on page 118 we prove the following Lemma.

Lemma 4.11 - Within a *j*-mini-superslot there is exactly one packet from every group.

Proof: given in Appendix A on page 213.

We can now prove the desired packet interleaving property

Theorem 4.6 - Packet Interleaving when starting at a superslot boundary

Let $G=W2^J$ (the number of FEC groups into which the file was divided for encoding) and $N=2^M$ (the number of code packets per FEC group). Then, for every subscription level l

(EQ 54)

$$0 \le l \le \min(J, M)$$

each NG consecutively scheduled packets[7] starting at a superslot boundary are all different ones. In other words, the packet index that accompanies the group index for every packets is such that within NG consecutive packets starting at a superslot boundary at any subscription level in the allowed range there is no packet that has been scheduled twice.

Proof:

Lemma 4.9 - on page 117 and Lemma 4.11 - on page 118

qed

7. in the union of channels that forms subscription level l namely: 0..l

Packet Interleaving when Starting Reception at Any Slot Boundary

We have proved that by starting reception at a superslot boundary, a receiver at any subscription level will see every different packet once before any packet is received for the second time. This is a desired property since we desire to mimic the case at which there are infinite code packets for every FEC group. The Packet Interleaving Theorem (Theorem 4.6 - on page 118) showed that when starting at a superslot boundary our schedule scheme behaves as close as possible to the ideal case at which there are infinite packets per group (where there is no client for which the same packets is scheduled twice during the whole file reception).

In some practical cases the superslot boundary requirement may be somewhat restrictive. For very large files, G becomes large and the distance between superslot boundaries grows proportionally. It is one of our goals to provide a scheme where individual receivers may join the transmission at any point in time. In this section we show the packet index interleaving properties of the proposed schedule when a receiver joins the transmission between superslot boundaries.

Theorem 4.7 - on page 119 proves that when starting at a slot boundary that is not a superslot boundary, a receiver still get at least $N/2$ different packets per group before seeing a repetition. This is a very satisfactory result considering that N is typically one order of magnitude higher than K[8]. It follows from results of the previous chapter that virtually all receivers would have been completed their reception, way before the $N/2$ different packets per group were transmitted which means that $N/2$ different packets per group is for all practical cases equivalent to infinite.

Theorem 4.7 - - At least N/2 different packets per group when starting at any slot boundary

With this theorem, we prove that when starting at ANY slot boundary (as opposed to a superslot boundary), for any subscription level:

(EQ 55)

$$0 \le l \le \min(J, M-1)$$

a client will receive at least *N/2* different packets per group before seeing a packet for the second time.

Proof:

From the Packet Interleaving Theorem (Theorem 4.6 - on page 118) we know that when starting at a superslot boundary, N different packets per group will be seen before a repetition occurs. We also know that within a superslot there are 2^l packet indexes.

8. i.e K=32 and N=256

The total number of superslots until complete reception of *NG* packets at subscription level *l* (Lemma 4.7 - on page 115) is 2^{M-l}. In the remaining complete superslots (excluding the one we started in the middle of) there are then:

(EQ 56)

$$2^l\left(2^{M-l}-1\right)=2^M-2^l$$

different packet indexes.

Since $l<M$ then the smallest number of packet indexes is given by:

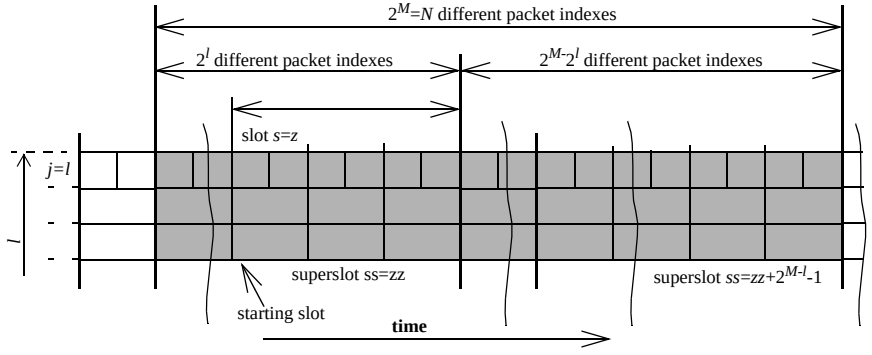
(EQ 57)

$$2^M-2^{M-1}=2^{M-1}=\frac{N}{2}$$

qed

Figure 4.26 on page 120 gives a graphical representation of Theorem 4.7 - on page 119. The figure depicts the situation wherein a receiver joins the transmission at a slot boundary which is not a superslot boundary. We claim this property to be very significant for virtually all practical cases. In a real scenario, *N* is on the order of 10 times *K* and therefore it is highly unlikely (as we have shown in our simulation results) that a client will need to remain tuned for more than half of the total amount of available packets in order to receive K different ones from each group which is all it needs.

FIGURE 4.26 - starting at a non-superslot boundary



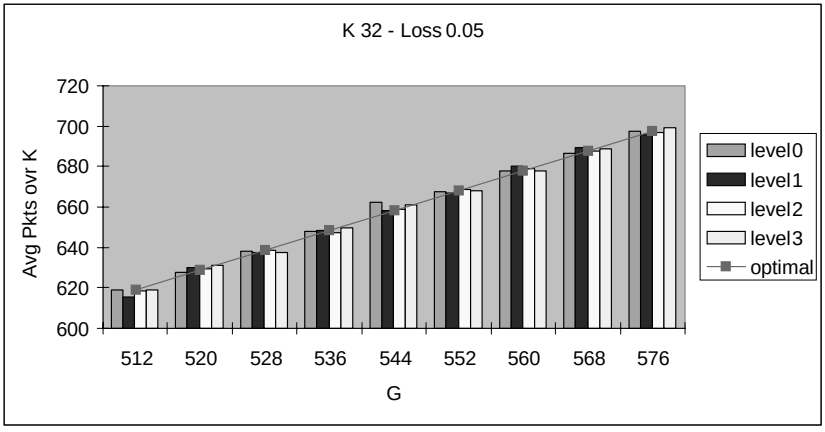
4.5 - Evaluation of the Cumulative Exponential Channels Scheme

Checking the Scheduler with a Simulated Model

Due to the complexity of the proof, we have simulated the“Cumulative Exponential Multi-channel Packet-Schedule” on page 95 and compared the obtained results to those calculated based on the model developed in the previous chapter. As expected, the results match.

In Figure 4.27 on page 121 we show some examples for G that is not a power of 2. We plot in the same chart the optimal calculated results. The results for the proposed schedule clearly match the optimal ones as expected from the proof above. We use $G = W2^J$ where J=3 and W runs from 64 to 72. We see that for subscription levels up to 3 the results are optimal as expected for all the plotted values of G. We further relax this restriction and analyze the results for higher rate channels in a forecoming section.

FIGURE 4.27 - Multilayer schedule - G not a power of 2



Higher Subscription Levels

We have proved so far that for subscription levels up to (and including) J, the schedule results are optimal. In this section we analyze the effect of higher subscription levels. A higher subscription level results in an

increase in the packet rate perceived by the client. On the other hand if the additional packets received are such that provide no help in recovering the file (i.e. same packets in the added channel as the ones already received in the other channels) then the subscription level increase is not useful. We have shown so far that the full benefits of the subscription level increase are achieved up to level *J*. In this section we show that beyond this subscription level the results are very close to optimal as well.

l=*J*+1

We have shown that given $G=W2^J$ our schedule is optimal in terms of group interleaving for every subscription level smaller than or equal to *J*. In this section we show that for *l*=*J*+1 the results are very close to optimal as well.

The results so far hold for any *W*, however the results in this section are clearly relevant for *W* which is an odd number (since otherwise we can keep diving it by two and incrementing *J* to use our previous optimality proof for higher subscription levels).

The number of slots needed for *G* packets at level *J*+1 is:

(EQ 58)

$$\frac{G}{2^l}=\frac{W2^J}{2^{J+1}}=\frac{W}{2}$$

However since we are assuming *W* to be an odd number we have no way of receiving the *G* packets with an integer number of slots. We will therefore move forward to receive a total of 2*G* packets which can clearly be received with *W* slots. We wish to show that these 2*G* packets are exactly 2 of each group. We prove this in Theorem 4.8 - on page 125 after the following definitions and intermediate Lemmas.

Lemma 4.12 - *G* packets of different groups at channel *j*=*J*+1 imply 2 of each group at level *l*=*J*+1

If the G packets sent in channel *j*=*J*+1 starting at any slot boundary belong to *G* different FEC groups, then the 2*G* packets sent at subscription level *l*=*J*+1 (which is the combination of channels *j*=0,1,...,*J*+1) starting at the same slot boundary include 2 packets from each FEC group.

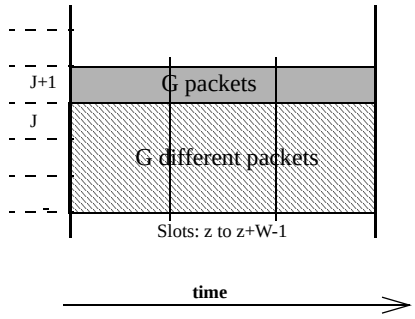
Proof:

We have already shown that for *l*=*J* there is exactly one of each of the G groups in *W* slots (Theorem 4.5 - on page 111). It follows that by adding one of each group (with channel *j*=*J*+1) we get 2 of each as desired

qed

Clearly from here it is enough for us to show that the *G* packets in channel *J*+1 are distinct (see Figure 4.28, “2G Packets at level l=J+1,” on page 123).

FIGURE 4.28 - 2*G* Packets at level *l*=*J*+1



We need to show then that the *G* packets in channel *J*+1 all belong to different groups.

(EQ 59)

$$\underset{j=J+1}{g} = \left| s + \left\lfloor \frac{G}{2^{J+1}} \right\rfloor + \left\lfloor \frac{G}{2^{J}} \right\rfloor \cdot t \right|_{G} = \left| s + \left\lfloor \frac{W}{2} \right\rfloor + Wt \right|_{G}$$

Once again we prove this for 0 as a starting slot (which can be extended in the same manner as done for the general case in Theorem 4.5 - on page 111).

Moreover, since

(EQ 60)

$$\left\lfloor \frac{W}{2} \right\rfloor$$

is a constant, it can be ignored as well during the proof.

Lemma 4.13 - - Modulo Elimination (*l*=J+1)

(EQ 61)

$$\underset{j=J+1}{g} = s + W \cdot t$$

Proof: given in Appendix A on page 214.

We next proceed to show that there are no two packets belonging to the same group within the range defined above that contains *G* packets and therefore the *G* different packets ought to be from G distinct groups.

W is a constant in our proof. It is then equivalent to prove that:

(EQ 62)

$$g^*(s, J+1, t) = \frac{g(s, J+1, t)}{W}$$

are all different within the allowed range of parameters (that spans *G* values).

Using the same definitions for the integers and non-integers parts of g* as in the previous section we obtain:

(EQ 63)

$$\left\lfloor \underset{j=J+1}{g^*(s, j, t)} \right\rfloor = \left\lfloor \frac{s}{W} + \frac{W \cdot t}{W} \right\rfloor = \left\lfloor \frac{s}{W} + t \right\rfloor \overset{t \in \mathrm{N}}{=} \left\lfloor \frac{s}{W} \right\rfloor + t \overset{s < W}{=} t$$

And:

(EQ 64)

$$\underset{j=J+1}{r(g^*)} = g^*(s, J+1, t) - \left\lfloor g^*(s, J+1, t) \right\rfloor = \frac{s}{W} + t - t = \frac{s}{W}$$

We now prove that the *j,s,t* representation of *g** is unique within the range of the parameters that spans *G* groups in the following Lemma.

Lemma 4.14 - Uniqueness of the *j,s,t* representation of *g within a range of G packets at channel *j=J+1*.**

If $g^*(s_1,J+1,t_1)=g^*(s_2,J+1,t_2)$ then $s_1=s_2$ and $t_1=t_2$.

Proof: given in Appendix A on page 215.

We can finally prove the desired property.

Theorem 4.8 - - 2 packets per group every 2*G* packets at level *l=J+1*

Let $G=W2^J$ be the number of FEC groups into which the file was divided for transmission. Then, any sequence of 2G consecutive packets transmitted at subscription level $l=J+1$ that begins at a slot boundary contains exactly two packets from each FEC group.

Proof:

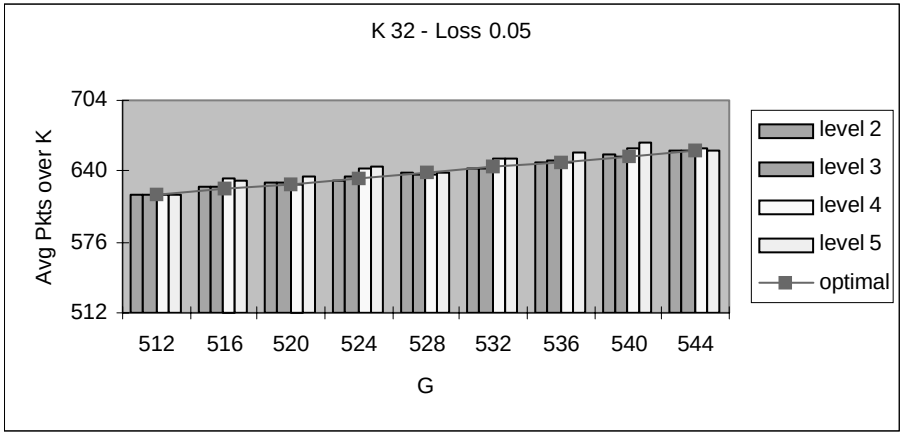
Lemma 4.12 - on page 122 and Lemma 4.14 - on page 125

qed

Higher Subscription-Level Simulations

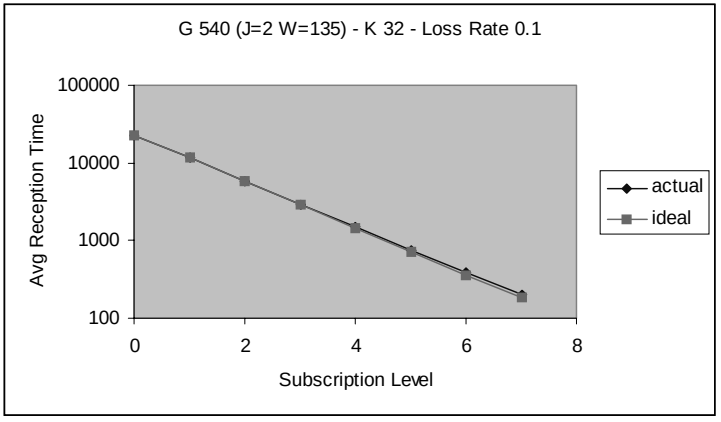
Figure 4.29 on page 126 shows the average amount of packets (divided by K which is a constant) sent by the server before the successful reception of the file by a receiver as a function of G (which is a direct consequence of file size) for various subscription levels. In this figure we can see the impact of subscribing to levels higher than the maximum one allowed for optimal interleaving. We use $G=W2^J$ where $J=2$ and W runs from 64 to 72. We see that for subscription levels up to 2 the results are optimal as expected. For level 3 we get optimal results as well. This matches the results of Theorem 4.8 - on page 125. For levels higher than 4 the results are not optimal (except for those values of G for which W has 2 as a factor and thus the results of Theorem 4.5 - on page 111 apply). However we see that the results degrade rather gracefully with the increase of the subscription level.

FIGURE 4.29 - Multilayer schedule - Rates beyond the optimal range



As can be seen, this non-optimality is mostly negligible (~1%). Moreover, it only affects receivers at the most higher rates where the reception time is extremely short anyway. We show this in Figure 4.30 on page 126, where we see that the average reception time attained by this fastest clients is barely distinguishable from the ideal case.

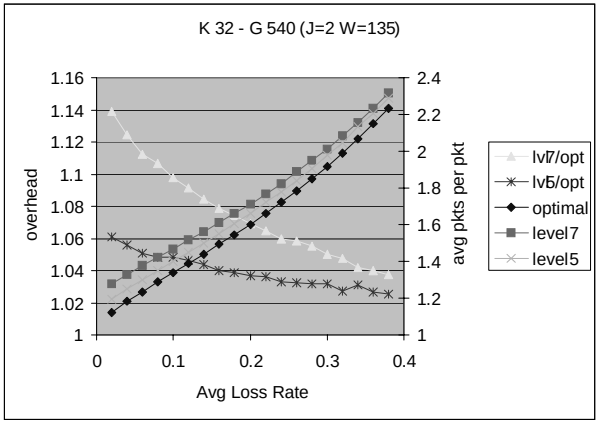
FIGURE 4.30 - Multilayer schedule - Rates beyond the optimal range



We further claim that it is far more important to provide perfectly optimal results to the lower subscription levels since clients receiving at these slower rates are those that are more impacted by small non-optimalities

in the schedule (in absolute time measures). A main property of our schedule is that these slower receivers achieve optimal results and are not affected whatsoever by faster clients that may trade slightly non-optimal schedule for a yet faster reception rate. We show an interesting property of this slight non-optimality in Figure 4.31 on page 127. In this figure we show that the negative effect decreases when the loss probability increases. This is also reasonable as the higher the loss probability the less the impact of a slight non-optimality in the packet schedule (more packets to cover for errors are needed anyway).

FIGURE 4.31 - Multilayer schedule - Rates beyond the optimal range



Strictly Optimal Schedules

Some alternative approaches can be followed if strictly optimal schedules are desired for an arbitrary *G*. A possible one would be to use the next closest value of *G* for which the schedule is optimal at all the desired levels. Then at the packet times corresponding to groups that do not exist in the real file the server would just not send anything. The effect of this approach is that of a virtual reduction in the rate of all channels by the ratio G/G* where G* is the next closest value of G that was used. This approach has the benefit of being self regulated. Receivers perceive a slower rate per channel and therefore would possibly subscribe to more channels to account for the difference[9].

9. With exponential channels and a cumulative approach like the one evaluated in this chapter, this is not likely to happen. Every jump in subscription level multiplies the rate by 2 and since the rate reduction caused by going to the next G is rarely 50% (unless we desire to support a max rate that is equal to G times the slower rate), receivers will likely stay at the same level. In chapters to follow we enhance the rate selection resolution and in that scenario receivers may get to increase their level in response to this approach.

A second alternative would be to divide the file prior to transmission so that the resulting smaller files result in values of G that match the desired maximum rate. The main drawback of this approach is related to network resources utilization. Different clients located close by in the multicast routing tree will be getting the separate file segments in an uncoordinated fashion. Then we will have links in the tree for which there is no single client downstream receiving all the file related data that is flowing through it.

Finally, a third alternative would be to tune the packet size and thus achieve a G that suits the needs of the maximum desired rate. A disadvantage of this approach is related to the fact that the packet size is commonly determined by lower protocol layers and changing it may involve some extra implementation issues. Apart from this, packets cannot be always enlarged since its size is limited from above by the Maximum Transfer Unit (MTU) of the network. Moreover increases in packet size augment the per packet loss probability. Reducing the packet size is on the other hand always possible. The problem with this is that smaller packets result in higher network overhead because of the fixed per packet price that is paid for its headers and trailers.

Related Work

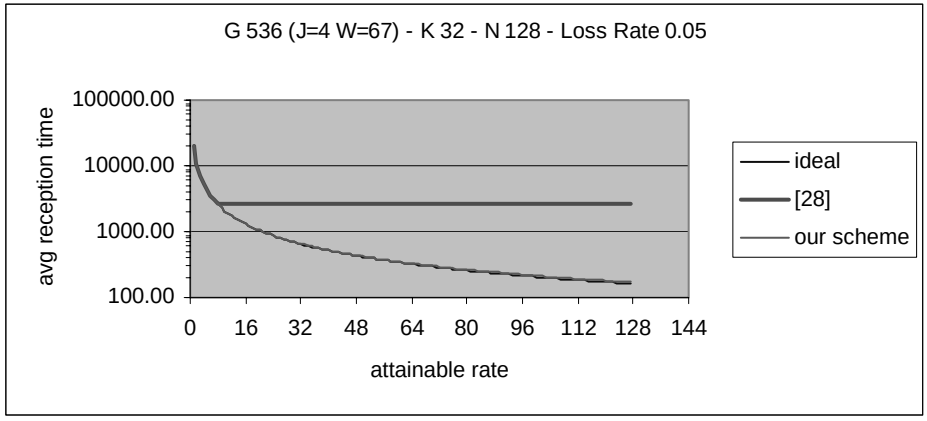
There is an unpublished related work on this field [29] where a multilayer schedule scheme is proposed. The suggested scheme allegedly provides maximum group interleaving for subscription levels up to J where $G=W2^J$. The unpublished work does not contain any proofs whatsoever so for the sake of comparison we implemented the proposed alternative mechanism and simulated its results.

The simulated results showed to be similar to those of our proposed scheme when the subscription level is smaller than or equal to J. However, the alternative scheme does not allow higher subscription levels. This is an important limitation as we have seen in previous sections. Arbitrary values of G (as mandated by the size of the file being transmitted) restrict the alternative scheme to a small range of subscription levels.

Figure 4.32 on page 129 shows the file reception time as a function of the attainable rate for our proposed scheme and the one in [29]. The ideal situation[10] is also shown for the sake of comparison. As we said above, the alternative scheme does not allow rates higher than those that result from a subscription level equal to J. Therefore, the curve showing its result is constant from that point and on meaning it can not take advantage of the increased attainable rate. In contrast, our scheme allows any rate as we can see in the chart. We can as well appreciate again that the non-optimality of our packet schedule for levels beyond J+1 is barely perceptible.

10. by ideal we mean that the file is received in time proportional to the attainable rate assuming a perfectly interleaved packet schedule

FIGURE 4.32 - Attainable Rate



Another salient drawback of the alternative scheme is the fact that the schedule for packets at each channel depends on the total amount of channels that are being used for the transmission. This is an undesirable property since it implies that once transmission has started there is no possibility of adding channels without affecting the schedule on the channels that were being used so far. Our schedule is independent of the total number of channels used.

We have shown that our scheme gracefully degrades when channels beyond the ones with optimal schedule are added. A key feature of our schedule is that this degradation affects exclusively those clients that are subscribing to these higher rate channels. Receivers that subscribe to levels up to and including J+1 were not affected in any way by the non-optimality of the added high rate channels. To the contrary, if [29] was to be trivially extended to support additional channels (beyond the allowed ones) then the slower channels would be greatly affected. The addition of channels beyond the optimal range would results in a performance degradation that would mostly affects the slower receivers. This is a highly non-desirable feature since these slower receivers are precisely those that are mostly adversely affected (in absolute terms) by a packet schedule non-optimality.

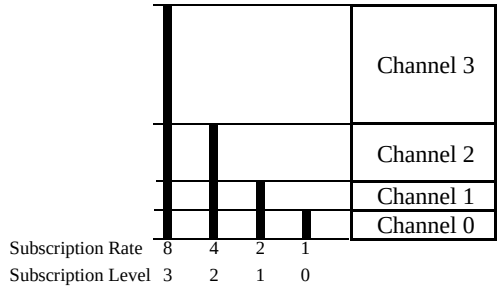
CHAPTER 5

Rate Resolution Enhancements for Layered Multicast

5.1 - Fine-Grain Rate Resolution for Layered Multicast

In the previous chapter, we presented a multichannel scheme and a cumulative subscription policy for receiver driven flow controlled multicast transmission of bulk data. We presented a per channel packet schedule mechanism that resulted in optimal reception time from the standpoint of each client for any of the allowed subscription rates and regardless of its starting time. The proposed schedule was developed over what we defined as exponential channels (where every channel rate is double the rate of the previous one).

FIGURE 5.1 - Cumulative Exponential Channels



The exponential nature of the channels combined with the cumulative subscription policy result in a restriction in the choice of available rates. Figure 5.1 on page 131 depicts this situation. The available rates are the integer powers of two multiplied by the base rate (which is the rate of the slowest channel C_0). With such a scheme, clients subscribe to a level that is the closest integer power of 2 smaller than or equal to their maximum attainable rate.

This restriction can be significant. The difference between the attainable rate and the obtained subscription level can be perceived by a client as a loss in performance. This is particularly relevant for the slowest receivers. In absolute terms, the slower receivers pay a high price for the rate selection restriction. As an example, a receiver with an attainable rate equal to 3 times the base rate that could have gotten the file in T seconds will receive it in about 1.5 times T seconds because it is limited to subscription level 1 where the rate is 2 times the base rate (since it cannot reach level 2). Clearly, this 50% increase in reception time can be noteworthy when the reception time is long as is the case with the slow clients. For faster receivers, even though the increase in reception time can approximate 100% in the worst case, its effect becomes negligible because of the very short reception times.

Related work on this field includes [30] where non-cumulative subscription policies are combined with channels whose rates are not exponential to achieve fine grain resolution. However, [30] is based on the use of proprietary error correction codes such as those presented in [28] and assumes that the whole file can be entirely encoded without partitioning. In contrast, our solution applies to publicly available erasure correcting codes. [16] suggests a router based filtering mechanism for congestion control and deals with the rate selection resolution as well. We suggest in this chapter two techniques that do not involve any change to the network infrastructure and are therefore easier to deploy.

In this chapter we extend our own scheme to provide more flexibility in the choice of the desired rate. We explore two alternatives in order to accomplish this goal namely: **channel sampling** and **non-cumulative subscription**.

With the first approach, we achieve higher resolution in the subscription rates by generating slower channels out of the exponential channels of the previous chapter. We show that the sender can generate a new set of channels all of the same rate by "sampling" our previously optimal exponential channels. These channels, together with a cumulative subscription policy, behave very close to the optimal case shown in the previous chapter while at the same time provide complete freedom in the rate selection. By "sampling" in this chapter we mean the generation **at the sender** of new channels with lower rates than the original ones. This should not be confused with the sampling that could be attempted by a receiver that is not able to cope with its received data rate (a thing that would waste valuable network resources).

The main drawback of the channel sampling technique suggested in the previous paragraph is the increase in the overall amount of multicast channels used. Our second approach to rate resolution enhancement uses a different technique that keeps the number of multicast channels to the same amount as in the original cumulative exponential scheme developed in the previous chapter. For this, we relax the restriction of cumulative

channels and evaluate a selective (non-cumulative) approach. With this approach, every receiver is free to select among the exponential channels, the desired combination that best matches its attainable reception rate. We test this selective (non-cumulative) approach using our proposed exponential channels with their packet schedule and show results that are very close to optimal. As expected, under a non-cumulative subscription scheme like this, clients may pick any combination of the transmitted channels and that may result in a non-optimal utilization of network resources. We show that for our suggested non-cumulative scheme, the network over utilization is bounded by a very small number, and analyze the implications of this fact for practical cases.

We further evolve our exponential non-cumulative scheme into a combined approach where intermediate nodes in the distribution tree (such as routers) consolidate the channel requirements from downstream agents in order to achieve optimal network utilization. This is done at the expense of not satisfying in some cases the exact requirement of one or more of the downstream agents. We present a consolidation heuristic that guarantees to each client at least the same rate as the one that would have been achieved under the exponential cumulative scheme. Our heuristics are based on an attempt to satisfy the slower receivers as close as possible to their exact requirement. This is motivated by the observation that these receivers are the ones that will mostly benefit (in absolute reception time) from an increase in their reception rate. Unlike the previous simulations where results were analyzed from a single receiver standpoint, for the router consolidation heuristics, we build a simulation infrastructure that enables us to mimic the effect of multiple receivers in a randomly generated multicast tree. We then apply our proposed heuristics to very large sets of receivers and show the improvement achieved over the exponential cumulative scheme of the previous chapter.

5.2 - Channel Sampling for Resolution Enhancements with Cumulative Layers

Introduction

In this section we propose a mechanism where the server constructs the transmission channels by regular sampling of the exponential channels that we presented in the previous chapter. We refer to this technique as **channel sampling**. The attained channels all have the same transmission rate. In this way, using a cumulative subscription policy, every transmission rate that is a multiple of the base rate can be attained by any receiver.

This is done at the expense of having to use more multicast channels in order to attain the same reception rate and some optimality loss in the aggregated packet schedule. In this section we show that the loss of performance that results from this non-optimality is negligible. With regards to the number of multicast chan-

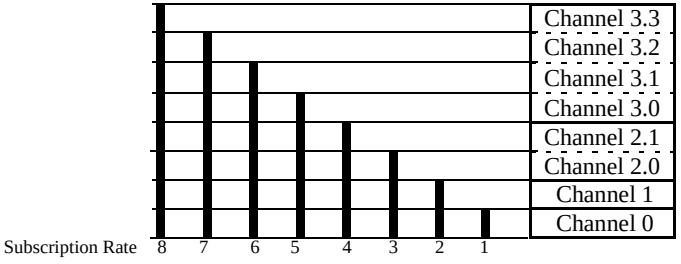
nels used, we address this issue in “Selective Channel Sampling” on page 141 where we suggest a trade-off between rate resolution and number of multicast channels required.

Cumulative Sampled Exponential Channels

In this section we use the exponential channels of the previous chapter to generate channels of a unique rate (the rate of C_0). When combined with a cumulative subscription policy, these channels result in performance close to the optimal proven for the exponential channels. The added benefit is that clients have much higher freedom in the selection of their reception rate. With exponential channels, only rates that are powers of 2 could be achieved. The channels presented below allow a client to subscribe to any desired rate.

Figure 5.2 on page 134 depicts a schematic view of the sampled channel construction and the corresponding attainable subscription rates.

FIGURE 5.2 - Cumulative Sampled Exponential Channels



In order to obtain sampled channels of the same rate, each exponential channel j is used to generate 2^{j-1} sampled channels denoted $j.t$ where:

(EQ 1)

$$t = 0,1,\cdots,2^{j-1}-1$$

Definition 5.1 - Sampled Channels Packet Schedule
For channel *j.t* the packet in position *q* is from the FEC group given by:

(EQ 2)

$$g = \left\lfloor \frac{q \cdot 2^{\max(0,j-1)} + t}{2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{G}{2^j} \right\rfloor + \max\left(1, \left\lfloor \frac{G}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left| q \cdot 2^{\max(0,j-1)} + t \right|_{2^{\max(0,j-1)}} \Bigg|_G =$$

$$= \left| q + \left\lfloor \frac{G}{2^j} \right\rfloor + \max\left(1, \left\lfloor \frac{G}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot t \right|_G$$

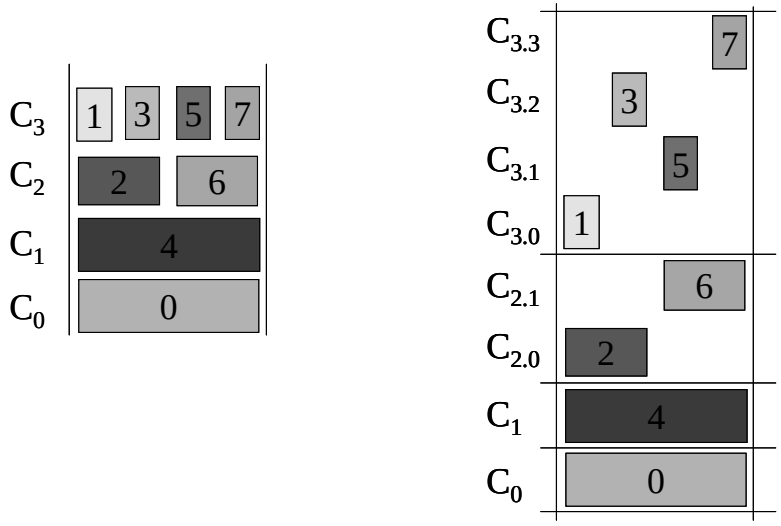
and its packet index is given by:

(EQ 3)

$$p = \left| \left\lfloor \frac{q \cdot 2^{\max(0,j-1)} + t}{G \cdot 2^{\max(0,j-1)}} \right\rfloor + \left\lfloor \frac{N}{2^j} \right\rfloor + \max\left(1, \left\lfloor \frac{N}{2^{\max(0,j-1)}} \right\rfloor\right) \cdot \left\lfloor \frac{q \cdot 2^{\max(0,j-1)} + t}{G} \right\rfloor_{2^{\max(0,j-1)}} \right|_N$$

Figure 5.3 on page 136 shows a graphical representation of the channel sampling for 4 exponential channels when *G* (the number of FEC groups) is 8. The numbers in the packets denote the FEC group to which they belong. Channels 0 and 1 are not changed. Channel 2 is used to generate Channels 2.0 and 2.1. Out of channel 3, channels 3.0, 3.1, 3.2 and 3.3 are generated.

FIGURE 5.3 - **Exponential Channels Sampling**



We now combine the sampled channels with the cumulative subscription policy and thus attain a scheme where each client can subscribe to any multiple of the base rate.

Our packet schedule was carefully built in the previous chapter so to attain optimal interleaving with exponential channels. Clearly when a receiver's subscription rate is equal to an integer power of 2 multiplied by the base rate the results will be the same as before as can be easily inferred from Figure 5.2 on page 134. However when this is not the case (as with a client with a subscription rate equal to 5) we expect the packet stream received by the client to lack some of the optimal interleaving properties. In the following section we show that as an additional property of the exponential channel schedule, the results of the presented scheme are very close to those of the optimal case.

Sampled Channels Simulation Results

The following charts show the results of the presented approach. We want to asses the impact of channel sampling on the per receiver performance. For this, we plot the normalized file reception time as a function of the subscription rate. The normalized reception time is obtained from simulation. We desire this measurement to behave proportionally to one over the subscription rate. This would be the ideal case. Because of the sub-optimality introduced in the received stream packet schedule when the rates are different from an integer power of 2 times the base rate, the actual results are expected to be somewhat higher than the ideal case.

In order to isolate the degradation of the optimal multilayer schedule when sampling is applied to achieve finer grain rate selection, we first run our simulator with loss rate equal to zero. We count the total number of packets seen by a receiver until successful reception of the whole file as a function of its subscription rate. Since the error rate is set to zero the only cause for a result higher than the optimal is the non optimality introduced by sampling. We show the same simulation for different FEC configurations (different values of *K,N*) to show that even with very little redundancy (*N* slightly higher than *K*) the results of the sampling scheme are very satisfactory. We moreover claim that this measurement of the sampling performance for loss rate 0 shows the worst case scenario. We later show that as expected when the loss rate is higher than zero the sampling has a yet smaller effect on the final result. The normalized time shown in the figures is the time that takes to complete the reception of 32 packets assuming each packet takes 1 unit of time to be transmitted at the base rate.

FIGURE 5.4 - Channel Sampling - Schedule Degradation Check (norm time)

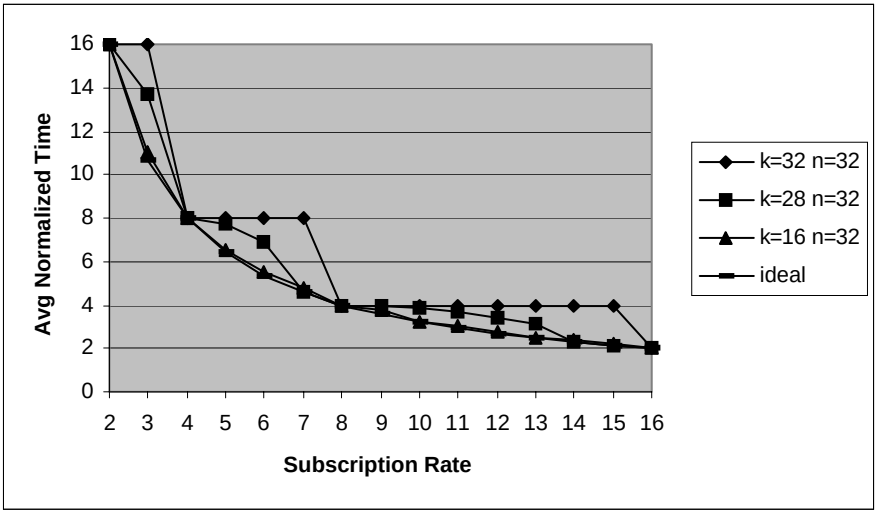


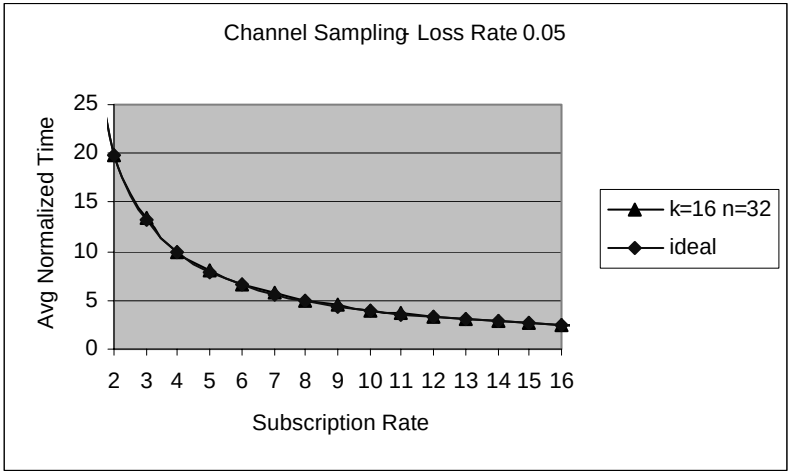
Figure 5.4 on page 137 shows this results for loss rate 0. The simulation is run for 3 different FEC configurations namely: no FEC (K=N), very little redundancy (N=32, K=28) and easy[1] FEC (N=32, K=16). We also plot the ideal behavior that would have been attained had the sampling resulted in an optimal packet schedule. As expected, when the rate is equal to an integer power of 2, the results coincide with those of the ideal case. As can be seen, the worst case results are obtained when no FEC is used. This was expected, with

1. We call this “easy” FEC because practical implementations allow N to be much higher than K which makes results even better.

no FEC, every single packet is required for completion and the non-optimality introduced causes some specific packets to be seen by the receiver at a later point in time. Even in such a case, the results are never worse than what would have been if the closest smaller integer power of 2 were picked. This is a trivial consequence of the cumulative subscription policy. The interesting result is that for very small redundancy configurations (*N* slightly higher than *K*) the normalized time behaves close to the ideal. Moreover for typical redundancy configurations such as the ones presented so far in this work the sampled schedule is barely distinguishable from the ideal case. Note that *N* is typically one order of magnitude greater than *K* in practical FEC implementations discussed so far while we attain results very close to optimal when *N* is only 2 times *K*.

We now simulate the more real case in which the loss rate is not zero and attain even better results as expected. Figure 5.5 on page 138 shows the time reduction achieved as a function of the subscription rate when the average loss rate is 5% and N=2K. As it can be seen, the results are indistinguishable from those of the ideal case.

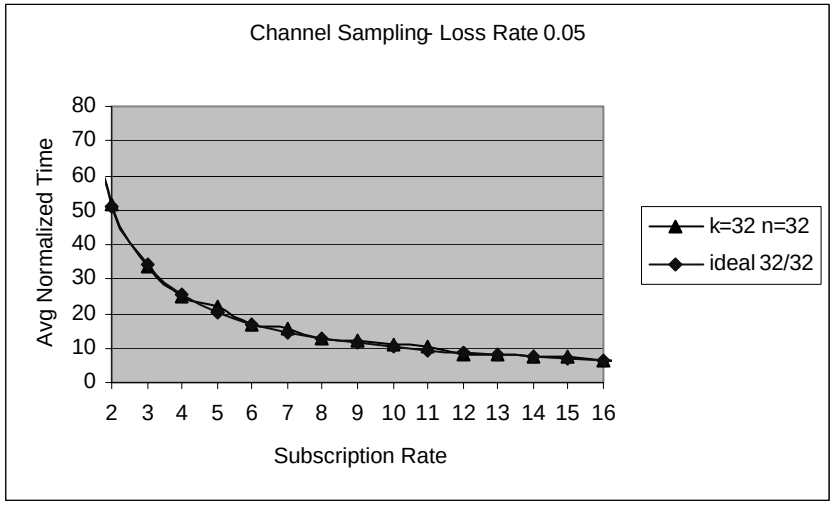
FIGURE 5.5 - Channel Sampling - FEC



In Figure 5.6 on page 139 we show that practical results (when the loss rate is not 0), even when no FEC is applied, are very satisfactory as compared to their corresponding ideal case[2].

2. which is trivially worse than the ideal case when FEC is applied and this is something we already investigated in a previous chapter

FIGURE 5.6 - Channel Sampling - No FEC



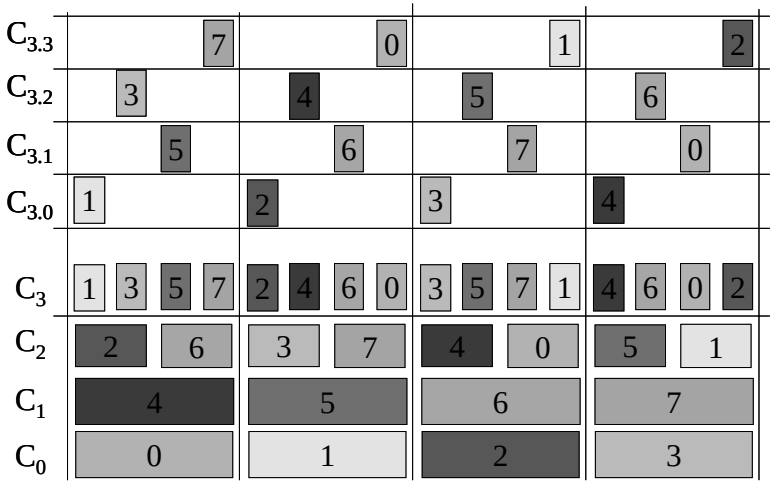
Sampled Channels Discussion

The results shown above for the sampled channels are really satisfactory. The performance is very close to that of the optimal schedule presented in the previous chapter for the exponential channels. The source for this achievement is in the original schedule itself. Packets are scheduled in the original exponential channels in a way that even when sampled the group interleaving properties are to a great extent preserved.

Figure 5.7 on page 140 depicts the intuition behind this preservation. Four exponential channels of a multi-layer transmission scheme are shown with the corresponding sampling of the highest rate channel that produces channels 3.0, 3.1, 3.2 and 3.3. The numbers in the packets denote the group to which the packet belongs. The packet schedule formulas were built so that when sampling is applied the resulting channels are as interleaved as possible themselves as it can be seen in Figure 5.7 on page 140. This is attained by the fact that the packet stream within each channel is cyclic with a period that is $2^{j-1}G$.

The theorem below proves the group interleaving property for the sampled channels.

FIGURE 5.7 - Group Interleaving for Channel Sampling



Theorem 5.1 - : Sampled Channels Group Interleaving

Any *G* subsequent packets from a sampled channel *j.t* belong to *G* different FEC groups.

Proof:

This property follows from the group index formula:

(EQ 4)

$$g = \left| q + \left\lfloor \frac{G}{2^{j}} \right\rfloor + \max\left(1, \left\lfloor \frac{G}{2^{\max(0,\, j-1)}} \right\rfloor\right) \cdot t \right|_{G}$$

clearly, *q* is the only element that varies. We can then remove all the other constant components to get:

(EQ 5)

$$g^{*} = \left| q \right|_{G}$$

which clearly results in *G* different values for any range of *q* of length *G.*

Finally, the addition of the constants that we removed will still leave us with *G* different values.

qed

Selective Channel Sampling

One drawback of the presented sampled channels technique is the increase in the number of multicast channels that need to be used to transmit the file. Multicast channels (and its group management infrastructure) can be a scarce resource in a global network such as the Internet. With exponential channels, the number of multicast channels needed for transmission is proportional to the log of the maximum desired transmission rate. With sampled channels, the number is proportional to the maximum rate[3].

We suggest that a trade-off can be attained between the amount of multicast channels used and the granularity in the selection of the subscription rate. A reasonable configuration would use sampled channels for the lower rates so that the slower receivers can subscribe to their exact attainable rate and leave a coarse resolution at the higher levels where clients are less impacted (in absolute terms) from a slower than attainable subscription rate.

FIGURE 5.8 - Trade-off between Number of Channels and Rate Resolution

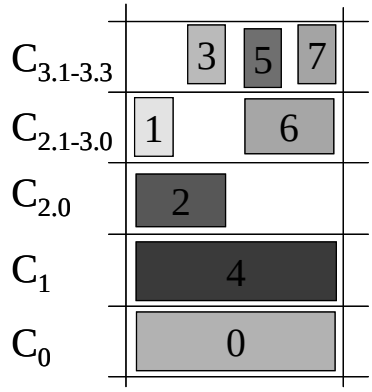


Figure 5.8 on page 141 shows an example of a such a trade-off. Under this scheme, the available rates are 1, 2, 3, 5 and 8.

3. we assume that the minimum rate and the desired resolution are equal to the base rate.

5.3 - Non-cumulative Exponential Channels for Enhanced Rate Resolution

Introduction

In this section we develop an additional approach to the rate resolution enhancement problem. This time, we relax the restriction of cumulative channels and evaluate a selective (non-cumulative) approach. Every receiver is free to select among the exponential channels, the desired combination that best matches its attainable reception rate. Clearly, given exponential channels, every multiple of the base rate can be attained by selecting the channels whose sum of rates is the desired value.

The packet schedule developed in the previous chapter was carefully built for a cumulative subscription rule. By using a selective approach, we expect to see packet streams, that for rates that are not an integer power of 2 multiplied by the base rate, do not comply with perfect interleaving as required for optimal results. In this section, we nonetheless show that this selective (non-cumulative) approach combined with our proposed exponential channel schedule obtains results that are very close to optimal from a receiver's perspective.

As expected, under a non-cumulative subscription scheme like this, clients may pick any combination of the transmitted channels and that may result in a non-optimal utilization of network resources as discussed in the previous chapter. We show below that for our suggested non-cumulative scheme, the network over utilization is bounded and very small and analyze the implications of this fact for practical cases.

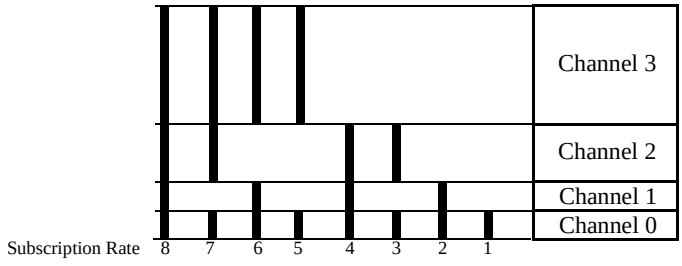
Non-Cumulative Exponential Channels

Since our exponential channels include 2 channels at which the rate is equal to the base rate (channels 0 and 1), there exists two possible subscription combinations for each desired rate[4]. Since we want receivers to be as coordinated as possible in the selection of the channels[5] we pick one of the possible combinations for each rate. Our choice is for the one that includes channel 0. The reason for this is related to the fact that when a client's attainable rate is equal to an integer power of 2 times the base rate, we desire its results to be those of the optimal case of the previous chapter. By requiring the selection of channel 0 in all cases we clearly achieve this result.

Figure 5.9 on page 143 depicts the described subscription rule for an example with 4 exponential channels.

4. For each rate, there is one channel combination that includes channel 0 and another one that does not.
5. For optimal network utilization

FIGURE 5.9 - Selective Exponential Channels



With the non-cumulative subscription, when a receiver subscribes to a rate that is not equal to an integer power of 2, we expect the packet stream received to lack some of the optimal interleaving properties. In the following section we show by simulation that as an additional property of the exponential channel packet schedule, the results of the suggested non-cumulative scheme are very close to those of the optimal case.

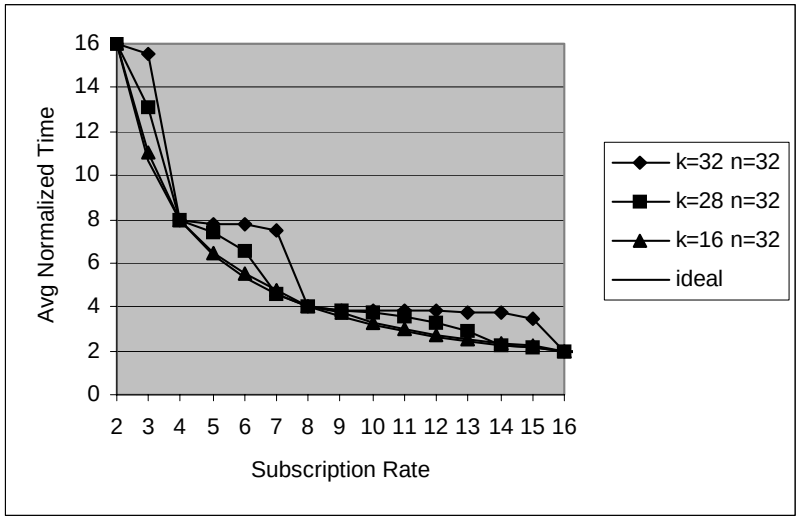
Non-cumulative Subscription Simulation Results

The following charts show the results of the presented approach. We want to asses the impact of non-cumulative subscription on the per receiver performance. For this, we plot the average normalized file reception time as a function of the subscription rate. The normalized reception time is obtained from simulation. We simulate the reception of 32 packets where each packet takes one unit of time at the base rate.

As in the previous section where the channel sampling method was evaluated, we first run our simulator with loss rate equal to zero in order to isolate the degradation of the optimal multilayer schedule when a non-cumulative subscription policy is applied to achieve finer grain rate selection. We count the total number of packets seen by a receiver until successful reception of the whole file as a function of its subscription rate. Since the error rate is set to zero the only cause for a result higher than the optimal is the non optimality introduced by the non-cumulative subscription. We show the same simulation for different FEC configurations (different values of K,N) to show that even with very little redundancy (N slightly higher than K) the results of the non-cumulative scheme are very satisfactory. We moreover claim that this measurement of the non-cumulative performance for loss rate 0 shows the worst case scenario. We later show that as expected when the loss rate is higher than zero the non-cumulative approach has a yet smaller effect on the final result.

Figure 5.10 on page 144 shows this results for loss rate 0. The simulation is run for 3 different FEC configurations namely: no FEC (K=N), very little redundancy (N=32, K=28) and easy[6] FEC (N=32, K=16). We also plot the ideal behavior that would have been attained should the non-cumulative scheme had resulted in an optimal packet schedule. As expected, when the rate is equal to an integer power of 2, the results coincide with those of the ideal case. As it can be seen, the worst case results are obtained when no FEC is used. This was expected, with no FEC, every single packet is required for completion and the non-optimality introduced causes some specific packets to be seen by the receiver at a later point in time. The interesting result is that for very small redundancy configurations (*N* slightly higher than *K*) the normalized time behaves close to the ideal. Moreover for typical redundancy configurations such as the ones presented so far in this work the non-cumulative method is barely distinguishable from the ideal case. Note that *N* is typically one order of magnitude greater than *K* in practical FEC implementations discussed in a previous chapter while we attain results very close to optimal when *N* is only 2 times *K*.

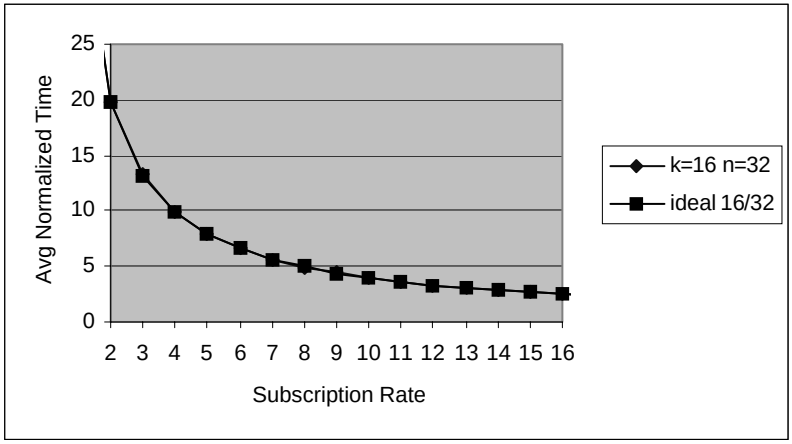
FIGURE 5.10 - Non-cumulative - Schedule Degradation Check (norm time)



We now simulate the more real case in which the loss rate is not zero and attain even better results as expected. Figure 5.11 on page 145 shows the time reduction achieved as a function of the subscription rate when the average loss rate is 5% and N=2K. As it can be seen, the results are indistinguishable from those of the ideal case.

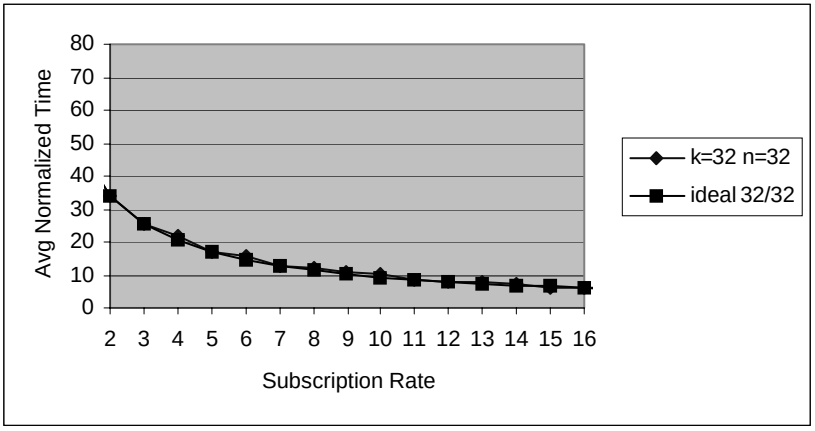
6. We call this “easy” FEC because practical implementations allow N to be much higher than K which makes results even better.

FIGURE 5.11 - Non-cumulative - FEC - (norm time)



In Figure 5.12 on page 145 we show that practical results (when the loss rate is not 0), even when no FEC is applied, are very satisfactory as compared to their corresponding ideal case[7].

FIGURE 5.12 - Non-cumulative - No FEC - (norm time)

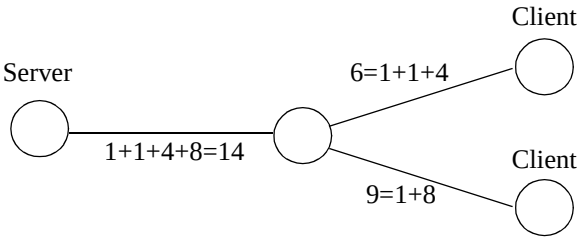


7. which is trivially worse than the ideal case when FEC is applied and this is something we already investigated in a previous chapter

Link Over-utilization with Non-cumulative Exponential Channels

As expected, with a non-cumulative subscription rule, two close clients with different attainable rates may cause non-optimal network utilization. This was extensively explained in the previous chapter. Figure 5.14 on page 147 shows an example of this situation.

FIGURE 5.13 - Network Resources Over-utilization Example



We define the over-utilization of a link to be the ratio between the sum of the rates of the channels being transmitted over the link and the subscription rate of the fastest of the receivers downstream that link.

$$over - utilization = \frac{\text{sum of bw of channels delivered through link}}{\text{subscription rate of fastest receiver}}$$

In the previous approaches we used cumulative schemes that guaranteed optimal network utilization. We trade-off optimal network utilization for rate selection resolution with the non-cumulative scheme suggested in this section. In the theorem that follows we bound the link over-utilization for the suggested scheme.

Theorem 5.2 - Link Over-utilization for Non-cumulative Exponential Channels

The link over-utilization for the non-cumulative exponential channels approach described above is smaller than 2.

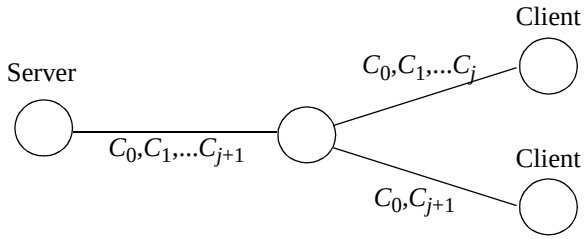
Proof:

The worst case over-utilization is attained when the fastest receiver subscribes to C_0 and C_{j+1} to obtain an overall rate of 2^j+1 while the channels requested by its neighbors downstream the considered link include all

channels from C_0 to C_j inclusive. This follows from the fact that the numerator of the over-utilization ratio is highest when all channels up to the maximum taken are present. Under that condition, the smallest denominator is achieved when the highest rate receiver takes C_0 (which all receivers take) and C_{j+1} which is necessarily taken by the one with the highest rate. (since the rate of C_{j+1} is equal to the sum of that of all the channels below and then if taken by one then that one is the one with the highest rate).

Figure 5.14 on page 147 shows an example of such a scenario.

FIGURE 5.14 - Link over-utilization bound for non-cumulative exp channels



The over-utilization is then:

(EQ 6)

$$\text{over}-\text{utilization} = \frac{1+1+2+\cdots+2^j}{1+2^j} = \frac{2^{j+1}}{1+2^j} = 2\cdot\frac{2^j}{1+2^j} < 2$$

qed

The bound on the over-utilization is very satisfactory. An over-utilization smaller than 2 may be perfectly reasonable for a multicast transmission. Remember we desire optimal network utilization to behave in a friendly manner towards other network traffic, and in this case an over-utilization of 2 may be completely justified by the fact that the packets are being received by multiple receivers[8]. In a slightly different context, the comment that it is reasonable for multicast flows to over-utilize the links up to a certain extent was pointed out in [31].

8. In fact many more than 2 clients will receive the same packet in most relevant cases. The extreme case where two close receivers subscribe in a way that the over-utilization is close to 2 and each packet is used by a single receiver is very rare.

5.4 - Router Consolidation for Optimal Utilization with Non-cumulative Channels

Introduction

We further evolve our exponential non-cumulative scheme into a combined approach whereby intermediate nodes in the distribution tree (such as routers) consolidate the channel requirements from downstream agents in order to achieve optimal network utilization.

Remark: The results in this section are also relevant to single group FEC schemes (such as those based on tornado codes for example [28]). Multicast distribution techniques that make use of proprietary codes which enable the whole file to be sent using a single FEC group still need to cope with the heterogeneity of the client rates. For this, multiple multicast groups can be used as described in the previous chapter. Exponential channels are a good candidate for this purpose since, combined with a non-cumulative subscription rule, they allow every rate to be selected while keeping the number of channels small (log of the max rate). The consolidation heuristics presented below are applicable to this case in order to eliminate the network resources over-utilization that stems from a non-cumulative scheme.

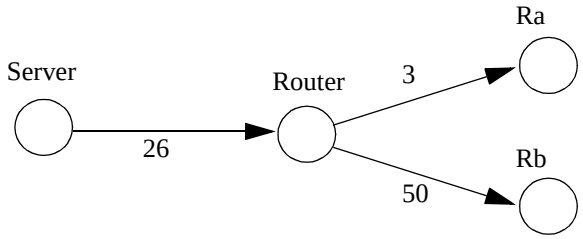
Optimum Network Usage with Non-Cumulative Exponential Channels

As we have seen in the previous section, with a non-cumulative subscription policy there are cases at which two receivers may cause non-optimal network utilization. We have shown that the over-utilization in such a case is bounded by 2 and that this results may be completely justified by the nature of multicast transmissions. In this section we develop a method for solving this small over-utilization that is suitable for cases where optimal utilization is desired by all means. This is done at the expense of achievable rate for some of the receivers which will not be satisfied to their total extent. We show that no receiver will get less than half of what it would have gotten on an equivalent unicast connection to the server. From the receiver perspective this is not worse than the cumulative approach and in practical cases could be much better as we will show below. Since network resources are optimally utilized we conclude this method to be better than the strictly cumulative one presented in the previous chapter.

The proposed algorithm is based on the intermediate routers consolidating the channel requests from their downstream agents. By definition, the maximum downstream requirement will never exceed the upstream capabilities since every receiver never requires more than what could be reaching it in an equivalent unicast connection to the server. The router’s task is to consolidate the downstream requirements so that network resources are optimally utilized. We remind that optimal network resources utilization for a link is defined as a channel subscription scheme that results in at least one receiver downstream requiring all the packets flowing through the link.

The router consolidation problem has more than one possible solution. In order to develop the motivations for the proposed solution, let us look at the example in Figure 5.15 on page 149 wherein a router has to consolidate the channel requirements from two receivers so that the resulting subscription achieves optimal network utilization.

FIGURE 5.15 - Router Consolidation Example



The numbers along the links represent the available link capacities for the file multicast in question. Receiver Ra will clearly require 3 units of bandwidth and Rb will require 26 (even though his directly attached link has 50 units of available bandwidth, receivers request according to what is available all the way up to the server).

According to our scheme for non-cumulative exponential channels, for its 3 required units of bandwidth, Ra needs C_0 and C_2. Rb needs C_0,C_1,C_4 and C_5. The router needs to consolidate them in a way that there is at least one of them receiving all channels.

In this case, the router cannot simply require the union of the two requirement sets. This is firstly because it would simply not fit the link limitation from the Server to the Router and secondly because the resulting subscription would not be network resource optimal. So clearly some requested channels will need to be dropped.

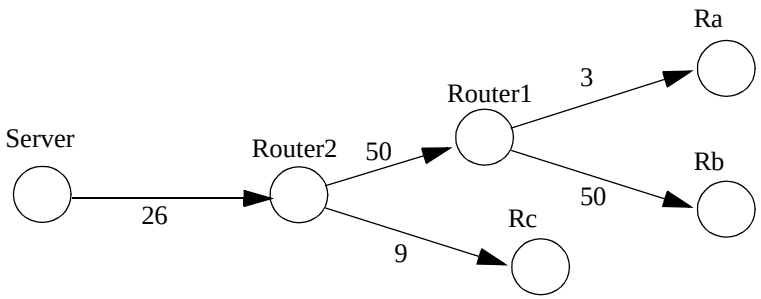
It doesn't make a lot of sense for the router to drop C_0 since it is required by both Ra and Rb (according to our channel selection per rate C_0 is always required). A possible approach would be for the router to drop the requirement of Ra for C_2. The resulting set does fit within the upstream bandwidth and indeed complies to the optimal network resource condition. One problem with this approach is that we penalized the weakest receiver. The results as far as Ra is concerned have been worsened by a factor of 3. From a different perspective the slower receiver penalization has another undesired result. In a large heterogeneous network it looks like it would be better for the router to keep slower channels since those give more freedom on the desired rate selection. In other words the lower the rate of a channel that is requested upstream, the higher the odds the request will go through and not be dropped by an intermediate router.

If the router is going to drop a channel other than C_2 then Rb will get less than what he asked for. However, if faced with a channel restriction, Rb could take advantage of other channels that could be made available still complying to the rate limitation and network optimality conditions. So in our second approach and motivated by the heuristics suggested in the previous paragraph, the router will drop C_4 and notify Rb he should subscribe to C_2 in exchange whereby keeping network utilization optimal. Under this subscription scheme, Ra will get its desired 3 units and Rb will get 20 instead of 26. This solution looks better than the previous one. The faster receiver was penalized but will receive at about 80% its desired rate which is not as bad as compared to the previous impact on Ra.

There is still more to be done by our consolidating router. Once C_4 has been marked for dropping it is clear from the exponential nature of our channels that Rb could subscribe to all channels below C_4. Therefore upon notification that C_4 has been dropped Rb will get subscription to C_3 and C_2 (C_1 and C_0 were already on its list). With this scheme, Rb will get 24 units of bandwidth which is over 90% of its required 26. We show below that in the worst case, a receiver will get more than or equal to 50% of its requested units of bandwidth.

Let us look at the following example in Figure 5.16 on page 150 to further emphasize the properties of the suggested method:

FIGURE 5.16 - Router Consolidation Example

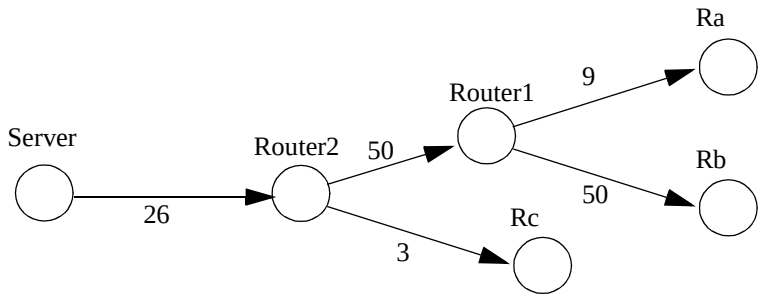


From the previous example we know that Router1 will consolidate the requests from Ra and Rb dropping C_4 and adding C_3 to the consolidated request. Rc will request C_0 and C_4. Following our algorithm, Router2 will need to drop the request from Router1 for C_5 and send him C_4 instead (C_3, C_2, C_1 and C_0 were already requested by Router1 anyway). Summarizing, Rc will get its requested 9 units of bandwidth. Ra will also be completely satisfied with its 3. Rb will get in this case 16 (out of its required 26). We can see again that optimality is achieved at the expense of the faster receivers and that the worst case reduction does not go over 50% of the attainable bandwidth for any receiver.

By looking at the previous example from a qualitative standpoint we can appreciate an interesting consequence of the proposed scheme. High speed receivers sort of pave the way for channels to flow where slower branches attached to the high speed tree have more freedom in selecting their desired connection rate.

There are still some more interesting insights to be noted. Let us examine an example in Figure 5.17 on page 151 which is intentionally similar to the last one:

FIGURE 5.17 - Router Consolidation Example



Now Ra has a limit on 9 units of bandwidth and Rc is the one with 3. Applying the same ideas we have been developing so far Router1 does not need to drop any channel request from either Ra or Rb. Router1 will then request C_0,C_1,C_4 and C_5 (this is what Rb requires where Ra requires C_0 and C_4). Then Router2 faces the same exact situation as the single router in our first example. We decided then to drop C_4 and to send C_3 and C_2 in exchange. By applying the same decision to this new situation we will slightly penalize both Ra and Rb. Ra will get C_0,C_2 and C_3 (7 units out of its desired 9) and Rb will get C_0,C_1,C_2,C_3 and C_5 (24 units out of 26). The result in itself does not look that bad however the scenario in the previous example resulted in a different result which may be regarded as better in the sense that the slower receivers were not penalized at all. A possible extension to our consolidation heuristics that addresses this issue and guarantees that the slower receivers are never penalized involves a full round-trip from the leaves up the tree to the source and back. Our goal is to keep the algorithm simple and limit to the minimum the number of control packets that are used to make it work. We leave the development of the consolidation heuristics so that in ALL cases the slower receivers are never penalized while minimizing the control overhead as an interesting case for future work.

In the following section we formalize the definition of the described heuristics and further elaborate on some of their features.

The Router Consolidation Algorithm

Our consolidation algorithm is naturally distributed. Every router does its part according to the rules described below.

The algorithm works in two stages. In the first stage, called **upwards consolidation**, agents notify their channel requests to their immediate upstream router. Upon reception of requests from all its directly connected downstream agents, each router consolidates these requests. The consolidated request becomes the channel request for the router itself and this is in turn submitted to its own next upstream router. This distributed process continues until the source (root of the tree) is reached.

The second stage is the **downwards update process**. In this stage that starts at the source when the upwards consolidation is completed, a router notifies its directly connected downstream agents the list of available channels after upwards consolidation and update. Each of these agents then updates (if needed) its own consolidated request and sends the updated list to all its own directly connected downstream agents. The downwards update process is completed when all the clients (leaves of the tree) have updated their channel requests.

The next sections describe in detail the algorithms implemented for **upwards consolidation** and **downwards update**.

Upwards Consolidation Algorithm

We denote with **consolidated request** the resulting request for the router in question. The algorithm below builds the **consolidated request** out of the requests of the individual directly connected downstream agents. Once calculated, the **consolidated request** is in turn requested from the next upstream router and so on up till the source.

```
1  upon reception of channel requests from all directly connected downstream agents:
2
3      sort all directly connected downstream agents according to their total requested rate in descending order.
4
5      the starting consolidated request is that of its highest requiring-rate directly connected downstream agent.
6
7      mark this agent as already consolidated.
8
9      while there are yet agents not yet consolidated.
10
11         pick the highest requiring-rate not-yet-consolidated agent. denote this agent with current agent.
12
13         pick highest rate channel requested by current agent which does not appear in the consolidated request.
14
15         if there is not such a channel
16             we are done with the current agent since all its requirements are already fulfilled by the consolidated request
17         else
18             denote this channel with c.
19             pick lowest rate channel higher than c in the consolidated request which is not requested by the current agent.
20             denote this channel with d.
21             remove channel d from the consolidated request.
22             add all channels slower than d to the consolidated request that were not there so far.
23
24         mark current agent as already consolidated
25
26     ask for consolidated request to next router upstream
27
```

Once all directly connected downstream agents are marked as consolidated the obtained **consolidated request** is the desired one and as such it is sent to the next upstream router for the distributed algorithm to continue.

Downwards Update

After the upstream consolidation algorithm is run by each router upwards in the tree up to the root, a downward update algorithm has to be run in order to notify the changes that the consolidation process may have produced. This is a simple procedure by which each router notifies its directly connected downstream agents of which is the final list of channels that are available after consolidation. We call this list the **updated consolidated request** and is generated by each router using the one received from its next upstream router and the algorithm below. The root has nothing to update so it simply renames its **consolidated request** into **updated consolidated request** and starts the downwards process by sending it to all its directly connected downstream agents. These in turn update their **consolidated requests** and continue the process down towards the leaves of the tree.

The downward update algorithm is described below:

```
upon reception of the upstream updated consolidated request from the next upstream router:

    if there is a channel in the consolidated request that is not present in the upstream updated consolidated request
        drop channel that is not present from consolidated request
        add all channels[9] below the one that was not present which are not already in the consolidated request

    send the updated consolidated request to all directly connected downstream agents.
```

Correctness of the Algorithm

For the **upwards consolidation** algorithm above to work, we need to prove that: given that there is a channel (denoted with c) requested by the **current agent** that is not in the **consolidated request,** then there has to be a channel (denoted with d) with a rate higher than c, that is in the **consolidated request** and is not requested by the **current agent**.

In order to prove that channel d exists we need to prove first the following intermediate properties.

We denote with **consolidation step** the part of the consolidation algorithm above that is within the while loop.

Lemma 5.1 - consolidated request rate after consolidation step that dropped a channel is never lower than that of current agent

Assuming that the channel to be dropped (denoted d) as defined by the upwards consolidation algorithm exists for a specific **consolidation step** where a channel ends up being dropped, the rate of the **consolidated request** after the **consolidation step** is never lower than the rate required by the **current agent** involved in the **consolidation step**.

Proof:

If a channel is dropped (denoted d), then all channels higher than c that were requested by the **current agent** were not dropped (because the dropped channel d is by definition not requested by the **current agent**) and all channels higher than c by the **current agent** were by definition part of the **consolidated request** already. In addition, all channels lower than c, are part of the consolidation requets for sure because of the dropping of d (which is higher than c) and results in the addition of all channels lower than d to the **consolidated request**. Therefore, the channels in the **consolidated request** after a **consolidation step** include all channels

9. we prove later that these channels must be part of the upstream updated consolidated request

requested by the **current agent**. Since all channels requested by the **current agent** are present in the **consolidated request** after the consolidation step then the rate of the **consolidated request** is higher than or equal to that of the **current agent**.

qed

Lemma 5.2 - consolidated request rate higher than that of the current agent implies that channel d exists.

Given that the rate of the **consolidated request** is higher than that of the **current agent.** If there is a need to drop a channel (denoted d) in the **consolidation step** that involves the **current agent** then the channel exists.

Proof:

Since there is a need to drop one, then there is a channel (denoted c) which is requested by the **current agent** and is not present at the **consolidated request**. However we know that the rate of the consolidated request is higher than that of the **current agent** so there must be one (or maybe more) channel in the **consolidated request** which is not requested by the **current agent** whose rate (or sum of rates) are greater than the rate of c. From the exponential nature of the channels, the sum of the rates of all channels below c is equal to that of c. Then for the rate of the **consolidated request** to be higher than that of the **current agent** there has to be a channel d with rate higher than c that is part of the **consolidated request** and is not requested by the **current agent**.

qed

We can now prove the existence of d with the following theorem:

Theorem 5.3 - Correctness of the Upwards Consolidation Algorithm

Given that there is a channel (denoted c) requested by the **current agent** which is not part of the **consolidated request**, then there is another channel (denoted d) in the **consolidated request** with higher rate than c such that d is not requested by the **current agent**.

Proof:

We prove by induction on the **consolidation step** that needs to drop a channel.

Let us first prove for the first such **consolidation step**. By definition the rate of the initial **consolidated request** is greater than or equal to that of the first **current agent**. For as long as the requests of the subsequent **current agents** (that are taken in descending order of rates) are such that no channel drop is required in their respective **consolidation step,** the **consolidated request** does not change and therefore it remains

being greater than or equal to the rate of the subsequent **current agents**. Then at the first **consolidation step** for which a channel is dropped, the rate of the **consolidated request** is greater than that of the **current agent** (cannot be equal otherwise a channel would not have been dropped) and the conditions of Lemma 5.2 - on page 155 are met so channel d exists and the base case is proved.

Let us assume now that the hypothesis holds after a **consolidation step**, we show that it will still hold after the next one.

Assuming then that channel d exists, if there is no further **consolidation step** where a channel needs to be dropped then the proof is done. If there is, then by Lemma 5.1 - on page 154, (which can be applied since we are assuming that d exists) the rate of the attained new **consolidated request** (after the channel drop) is greater than or equal to that of the **current agent** involved in the **consolidation step**. Then, from the descending order of rate at which the **current agent** is selected, the rate of the **consolidated request** is greater than or equal to that of the next **current agent**. This brings us to the same situation as proved for the initial step of the induction.

qed

We now wish to prove an additional property of the presented heuristics. We desire to show that the consolidation algorithm never results in the dropping of more than one channel of those requested by its directly connected agents.

Lemma 5.3 - Dropped channel rate never decreases in subsequent consolidation steps

Given there are two **consolidation steps** i and j where a channel was dropped. Assuming with no loss of generality that i>j then the rate of the dropped channel in **consolidation step** i is greater than that of that in consolidation step j.

Proof:

When a channel is dropped in **consolidation step** j all channels below the dropped one are added to the **consolidated request**. Therefore, in the subsequent **consolidation step** i, when picking the highest rate channel requested by the **current agent** (denoted c) which is not part of the **consolidated request**, none of the channels from the lowest one up to (and including) one less of the one dropped in step j will satisfy this condition (since they are all already part of the **consolidated request**). We conclude then that c has to be higher than or equal to the channel that was dropped in **consolidation step** j. Since the channel dropped in each consolidation step is higher than the one picked from the **current agent**, then the channel dropped in **consolidation step** i has to be higher than the one dropped in **consolidation step** j.

qed

Theorem 5.4 - A Single channel is dropped by consolidation algorithm

The upwards consolidation algorithm at any given agent drops at most one single channel from those in the requests of all its directly connected agents.

Proof:

The proof is by induction on the **consolidation steps** that result in a channel being dropped.. At the beginning there are no channels dropped. There is nothing to prove if none or only one channel is ever dropped by the consolidation algorithm. Let us assume that one single channel is currently dropped and see what is the result after a **consolidation step** where an additional channel is picked to be dropped.

By definition of the algorithm, when the first channel was dropped, all the channels below were taken. If in a subsequent step another channel happens to be dropped, this will be of higher rate than the one that was dropped before as follows from Lemma 5.3 - on page 156 which in turn will result in the channel dropped before to be added again to the consolidated request. This will leave us again with just one single channel dropped.

qed

The downwards update process is based on the assumption that the **upstream updated consolidated request** received from the next upstream router includes all channels in the **consolidated request** except possibly one. It further assumes that if there is one channel missing, then all channels with a lower rate than that one are present in the **upstream updated consolidated request**. We prove these properties in Theorem 5.5 - on page 158. For its proof, we need the following Lemma:

Lemma 5.4 - Dropped channel rate never decreases in subsequent consolidations of upstream routers

Given two routers denoted Router1 and Router2 such that Router1 is upstream from Router2. If both drop a channel during their upwards consolidation, then the rate of the channel dropped in Router1 is higher than the one dropped in Router2.

Proof:

At the end of its consolidation, Router2 dropped at most one channel as proved by Theorem 5.4 - on page 157. When Router2 drops its channel, it adds all the channels with rate lower than the one dropped. Applying induction now on the subsequent **consolidation steps** until Router1, it is clearly seen that the channels below the one dropped by Router2 are all present and therefore channel d (candidate for dropping) will never be one of them.

qed

We can now prove the desired property.

Theorem 5.5 - At most one single channel is dropped by whole upwards/downwards process at each agent

After the whole upwards consolidation process and the subsequent portion of the downwards update that occurs until an agent is reached for its own downwards update, there is at most one single channel from its own **consolidated request** that was dropped. Moreover, if a channel was dropped, then all channels slower than that one, are present in the updated channels list received from the next upstream router in the downwards update process.

Proof:

Theorem 5.4 - on page 157 proves that at most one channel is dropped by the consolidation algorithm of the agent immediately above in the upwards consolidation stage. From the consolidation algorithm, it follows that if one channel was dropped then all slower channels were added. Again through the use of induction, assuming there is one channel that was dropped at some point above our agent during the upwards consolidation, if there is a new channel that is going to be dropped by a router higher in the tree this channel will be higher than the one that was dropped before (as follows from Lemma 5.4 - on page 157). Once this new channel is dropped, all lower channels are added so the previous one is restored leaving us again with just one dropped.

During the downwards update process no channels are effectively dropped. The update is done based on what was already dropped during the upwards consolidation stage.

qed

We have proved the correctness of both the upwards consolidation stage and the downwards update process. We have thus proved the correctness of the presented consolidation algorithm.

Properties of the Algorithm

The motivation for the presented consolidation heuristics is to obtain a scheme where network resources are optimally utilized while enabling receivers to obtain a higher resolution in the selection of their subscription rate as compared to what can be achieved with a strictly cumulative scheme such as the one presented in the previous chapter. In order for the router consolidation mechanism to satisfy its requirements, the following two conditions have to be met:

- The router consolidation algorithm results in optimal network utilization
- Under the router consolidation algorithm, all receivers achieve at least the rate that would have attained if the strictly cumulative subscription scheme of the previous chapter would have been used.

We prove this two properties in Theorem 5.6 - on page 159 and Theorem 5.7 - on page 160.

Theorem 5.6 - The router consolidation algorithm results in optimal network utilization

The channel subscription obtained with the router consolidation algorithm is such that for all links in the tree, there is at least one client that is subscribed to all channels that flow through the link.

Proof:

Let us prove by induction on the distance from the leaves (clients). For this we start with the routers all of whose directly attached downstream agents are clients (as opposed to other routers). The construction of the upwards consolidation algorithm is such that the directly connected agent with the highest request rate will always subscribe to the **consolidated request** of the router. This is because its rate is either equal to the rate it requires (since its request is used as the initial **consolidated request**) or it is smaller (because of some channel drop during the consolidation process). As for the update that happens during the downwards update process, this never increases the **consolidated request** so the highest capable client will keep receiving all channels that remain. We have shown then that for this last hop routers there is at least one client that receives all the channels that flow through the link.

Let us now assume that the theorem holds for all the intermediate **agents** which are the ones directly connected downstream agents of another router. From the assumption, all the agent's updated consolidated requests are being received by at least one client downstream. The situation is therefore equivalent to that in the previous paragraph (if we replace the **agents** are the receivers at the end of their chain that subscribe to all their **updated consolidated request**) which means that the **updated consolidated request** of the router in question is also received by at least one client downstream.

qed

For the proof of the second property we need the following intermediate lemma.

Lemma 5.5 - Adjacent channels are never dropped

Let us define **adjacent channels** to be a set of channels whose indexes are consecutive. Adjacent channels that start at channel 0 are never picked to be dropped by the consolidation algorithm.

Proof:

The consolidation algorithm picks the channel to be dropped (denoted d) by selecting the lowest rate channel that is part of the **consolidated request** whose rate is higher than that of a channel (denoted c) that does not appear on the **consolidated request** and is part of the **current agent** request. From that, it follows that there is a "hole" in the consolidated request (channel c) so channel d cannot be part of **adjacent channels** from

previously consolidated agents. In particular since channel d is not present in the request from the **current agent** it cannot be part of adjacent channels in its request either.

qed

Theorem 5.7 - No receiver will get less than what would have gotten on a strictly cumulative approach

The consolidation mechanism never results in a receiver obtaining a rate which is less than the closest integer power of 2 that is lower than its desired rate.

Proof:

When a receiver requests an integer power of 2 then by the exponential nature of the channels there are no "holes" in the subscription request. In such a case, no channel will be dropped by the consolidation algorithm as follows from Lemma 5.5 - on page 159.

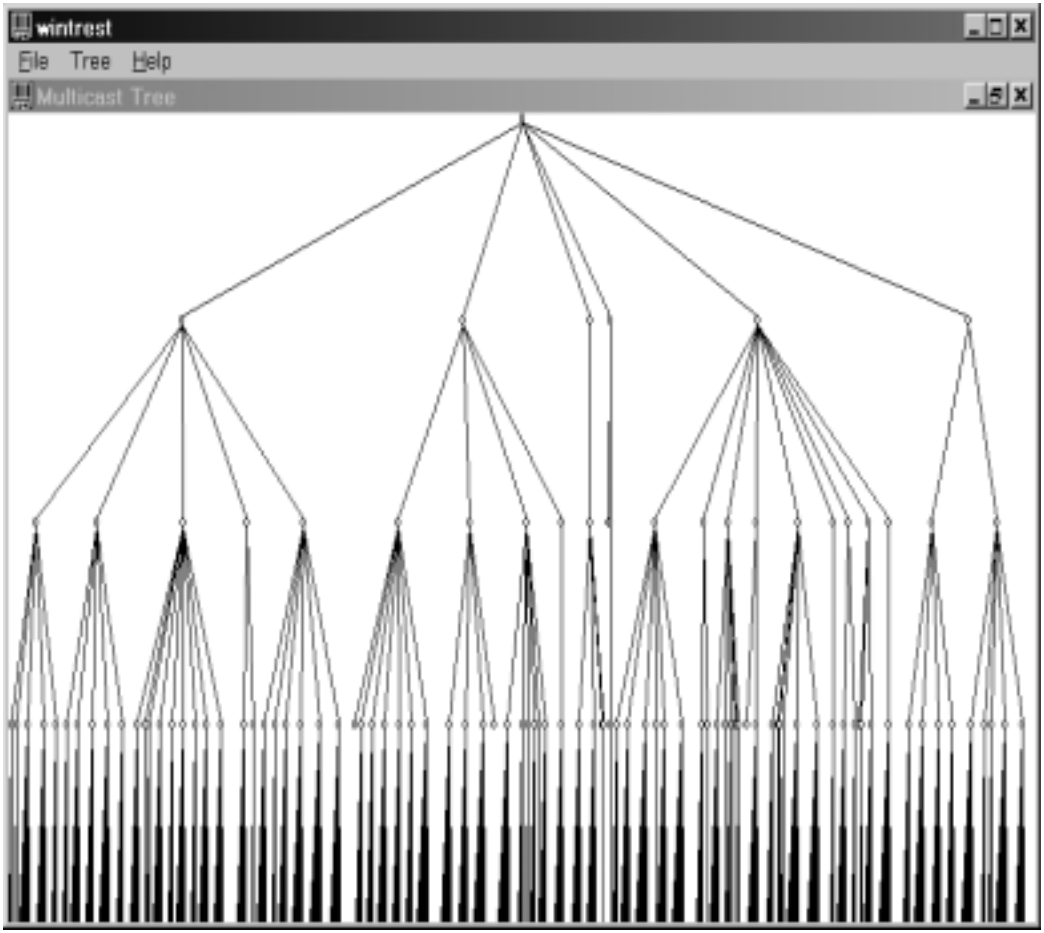
We have already proved that for each client at most one single channel is dropped from its request. The worst case is then when the highest rate channel is the one dropped. In such a case, from the definition of the downstream consolidation algorithm, all channels below the one dropped are added to the request thus making an adjacent set of channels up to channel 0. From the exponential nature of the channels, the sum of rates of this adjacent channels is equal to the closest power of 2 lower than the originally requested rate

qed

The Tree Generator

In order to simulate the proposed consolidation algorithm, the previous techniques that were used along this work are not suitable anymore. So far, in order to asses the performance perceived by a receiver we had to model the behavior of a single client. For this case we need a mechanism that models the interaction among a vast number of receivers with different attainable rates. We wish to evaluate the consolidation process using a rich scenario. For this we developed a random tree generator that is described with more details in Section B.1, "Tree Generation Algorithm," on page 217. The fanout of each node is randomly generated with a uniform distribution over a programmable range. An example of the output of the generator is shown in Figure 5.18 on page 161.

FIGURE 5.18 - Random Tree Generation Example



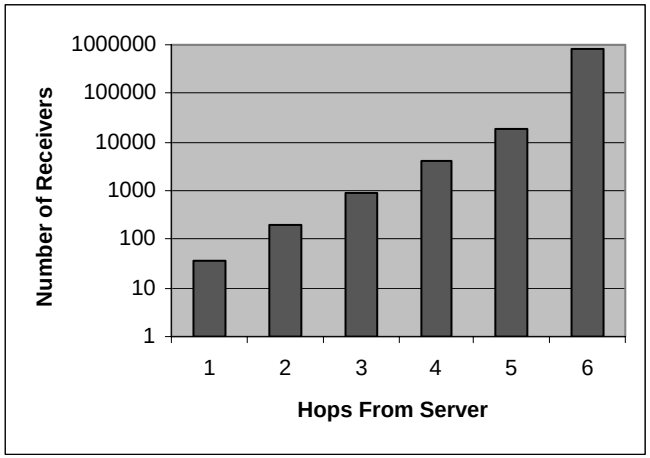
The capacity of the links are also uniformly distributed with a programmable parameter. The rate requested by each receiver (leaves in the tree) is then the minimum capacity found on the path up the tree towards the source (root of the tree). An additional control implemented in the generator is that of the maximum tree depth.

We used this generator to create scenarios where the consolidation heuristics were tried. The simulation results are shown in the next section.

Simulation Results

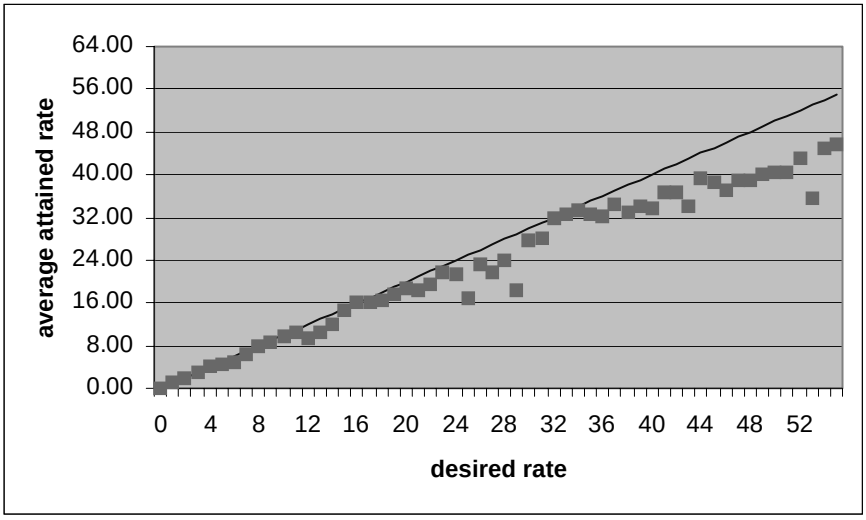
Using the tree generator presented above, we generated several distribution trees and simulated our algorithm. The results presented below correspond to an average over 100 generated trees totalling about 1 million receivers. Figure 5.19 on page 162 shows how far these receivers are from the server in units of hops.

FIGURE 5.19 - Distance from Server Histogram



The results in Figure 5.20 on page 163 show the main advantage of using the proposed consolidation heuristics. As it can be seen, the slower receivers, those with low desired rates, achieve pretty much exactly what they requested.

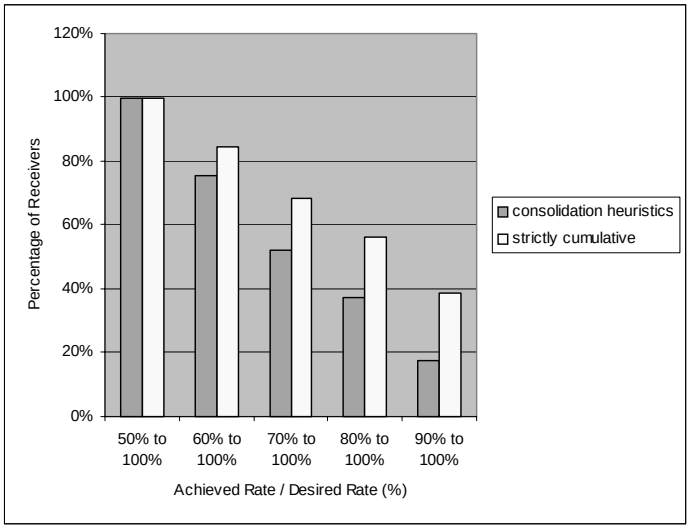
FIGURE 5.20 - Consolidation Heuristics - Attained Rate vs. Desired Rate



As desired rate grows, in favor of optimal network resources utilization, there are some receivers that do not get their exact request. However, as we proved above, there is no receiver that will get less than what would have gotten under the cumulative subscription approach described in the previous chapter (which is the closest integer power of 2 times smaller than the desired rate).

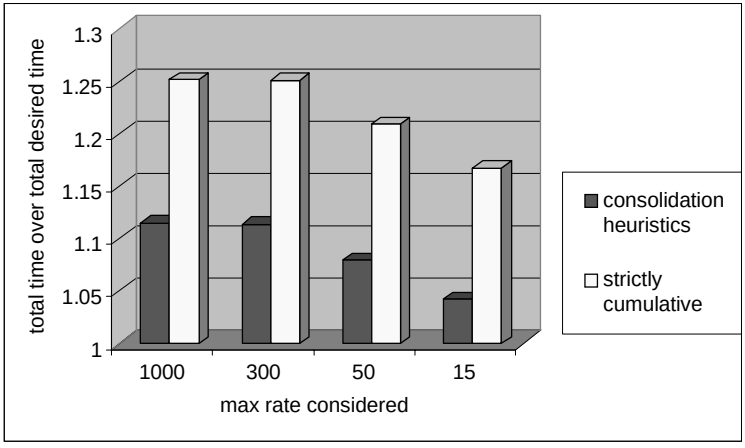
Figure 5.21 on page 164 compares the histogram of the number of receivers per ratio between desired and attained rates between the consolidation heuristics and the strictly cumulative subscription system of the previous chapter. As it can be appreciated, the number of receivers that obtain from 90% to 100% of the their desired rate is more than doubled with the consolidation algorithm.

FIGURE 5.21 - Consolidation Heuristics - Customer "Satisfaction" Histogram



Finally, in Figure 5.22 on page 165 we show the benefits of the consolidation mechanism as compared to the strictly cumulative scheme of the previous chapter from the server's perspective. We calculate for this figure the sum of the total avg transmission times for all the receivers. We do this for the ideal desired rates (as if each client could receive the file using its desired rate), for the rates attained through the use of the presented consolidation heuristics and finally for the case where the attained rates are the closest smaller integer powers of two as in the previous chapter. We plot then the ratio between the two approaches (heuristics and strictly cumulative) and the ideal case. We do this for different receiver sets. For example, the second set of columns represents the data that follows from the analysis of all receivers whose attainable rates do not exceed 50. As it can be seen, the consolidation mechanism results in better results. As expected the better results are more salient when the slower receivers (which are those contributing the highest to the total average reception time) are considered.

FIGURE 5.22 - Consolidation Heuristics - The Server's Perspective



Conclusions

We have presented a distributed consolidation algorithm and proved that it improves the results of our previous scheme from a client perspective by allowing some clients to attain rates closer to their available ones. We have also proved that the presented algorithm achieves optimal network utilization.

We do not claim the presented heuristics to be an optimal solution to the problem of assigning subscription rates to a set of heterogeneous clients so that network resources are optimaly utilized while minimizing the overall reception time for the set of receivers. We believe the optimal solution to be a problem of non-tractable complexity, prove is beyond the scope of this work.

As an additional remark, dynamic application of the consolidation algorithm can lead to subscription changes even if the network conditions remain static. Consider the case where one client joins the transmission when some others have already been receiving the file for a while. This subscription change scenario is equivalent from a receiver's perspective to the one that results from changes in the congestion status of the network. A evaluation of the network dynamics scenario is presented in "Network Dynamics" on page 167.

5.5 - Fine Grain Rate Resolution Conclusions

We have developed in this chapter methods for increasing the resolution in the selection of a subscription rate by a receiver of a multi-rate cyclic multicast. Two techniques where presented namely: channel sampling and non-cumulative subscription. Both techniques achieve near-optimal results from a receiver's perspective. The difference among the two methods is related to the utilization of network resources.

The channel sampling technique maintains the optimal network resource utilization through its use of a cumulative subscription policy. This technique is suitable to cases where the extra number of multicast channels used does not present a serious problem.

Alternatively, when the number of available multicast channels is scarce, the non-cumulative subscription technique is a good option. Its only drawback is in the small over-utilization of network resources that may result from its inherently non-coordinated subscriptions. Finally when even a small over-utilization becomes a problem, the presented router consolidation technique can be applied to attain optimal network utilization results.

CHAPTER 6

Network Dynamics

6.1 - Introduction

In previous chapters we presented a packet schedule mechanism for receiver driven rate control multicast distribution of bulk data. In this chapter we deal with the effects of network dynamics.

Congestion control in computer networks is usually implemented by means of injection rate control asserted at the transmission endpoints. Upon detection of congestion (either by perceived loss rate or through explicit notification) endpoints reduce their injection rate according to a predefined scheme. With this, congestion is alleviated. The congestion control protocol defines as well the recovery mechanism by which the rate may be increased when no congestion is detected.

This is basically the way most traffic is controlled in the IP Internet. The success of this scheme is based on consistent behavior of all intervening agents. The term TCP friendly is usually used to describe a congestion control mechanism that competes fairly for bandwidth with TCP streams in the IP internet.

In our scenario, the server is not capable of controlling the injection as described above. For that purpose we showed a receiver driven mechanism that works by means of subscription to different multicast groups (channels) that form a transmission. When congestion is detected (packets are lost) our clients decrease their reception rate by dropping subscriptions to one or more multicast channels. The multicast pruning mechanism results in effective rate reduction as a consequence of the subscription drops. In turn, a client subscribes to more channels when no packet losses have been detected during a programmed amount of time.

Research that has taken place recently on techniques for achieving TCP fairness with receiver driven multicast schemes as described above include [15] and [17].

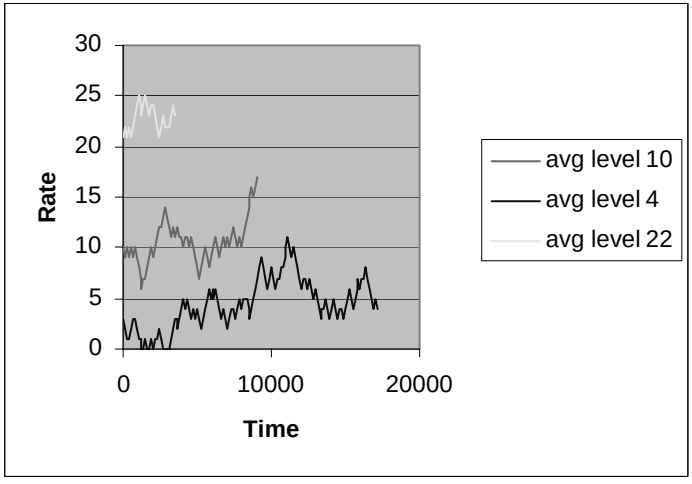
In this chapter we show the impact of network dynamics on the performance perceived by mulitcast receivers that use the channel subscription schemes presented so far. We model the rate changes using a probabilistic approach.

6.2 - The Effect of Network Dynamics

The Dynamic Network Model

As stated above, in practical cases, the network conditions vary over time. It is therefore expected that a client will be subject to many different attainable rates during the reception of a long file. In order to evaluate the behavior of the presented techniques under such a dynamic scenario, we simulated the rate changes and measured its impact in the file reception time.

FIGURE 6.1 - Subscription Rate Changes



We adapted our simulator so that every programmable amount of time the reception rate may be changed. We use a uniformly distributed random variable with three programmable ranges to determine at the end of

each of these time intervals, whether the rate is increased, decreased or left unchanged. This model resembles the situation in real networks[1] as can be seen in Figure 6.1 on page 168. In this figure we show the subscription rate as a function of time for 3 different clients that started at different rates. Clearly, the overall reception time for a faster receiver is shorter and hence its time-line terminates before that of the slower ones.

In the example of Figure 6.1 on page 168, the uniformly distributed random variable that controls the dynamic rate changes obtains values between 0 and 1. If its value is lower than 0.45 then the rate is increased. If it is greater than 0.45 but smaller than 0.9 the rate is decreased. Finally if the value is over 0.9 the rate is not changed.

As a consequence of rate changes, the packet stream received by each client is in a certain way distorted. The packet scheduling presented in previous chapters was carefully tailored so that the combined packet stream had some optimal interleaving properties (for all subscription rates simultaneously). When the rate is changed during the file reception some of these properties are affected. In the following sections we measure the impact of this effects in the performance results perceived by the clients.

Network Dynamics Simulations

We measure in this section the impact of rate changes in the performance perceived by the receivers. For this we calculate the number of packets transmitted by the server during the reception process of a client taking into account the different rates to which it subscribed. Since the subscription rate is changed during the reception process, the packets counted include those that were sent on all the channels to which the receiver subscribed while the subscription was in place. In other words, the packets counted are those that were received by the client and those that would have been received if they would not have been lost in the network. This number is divided by the one obtained when a single rate was used for the whole file reception. The ratio attained is what we denominate the overhead of network dynamics which we show in the following figures as a function of all relevant parameters.

In Figure 6.2 on page 170 we show the overhead of network dynamics as a function of the average loss rate. As it can be seen, for a very wide range of loss rates the results are fairly constant and around a very small overhead of 2.5%[2].

1. In TCP congestion control, additive increase and multiplicative decrease are used.
2. The slight decrease in overhead perceived as the loss rate increases is clearly a consequence of the fact that the higher the error-rate, the lesser the effect of a carefully tuned packet scheduling. When lots of packets are lost (such as with a rate of 40%), almost every packet is useful.

FIGURE 6.2 - Network Dynamics - Function of Loss Rate

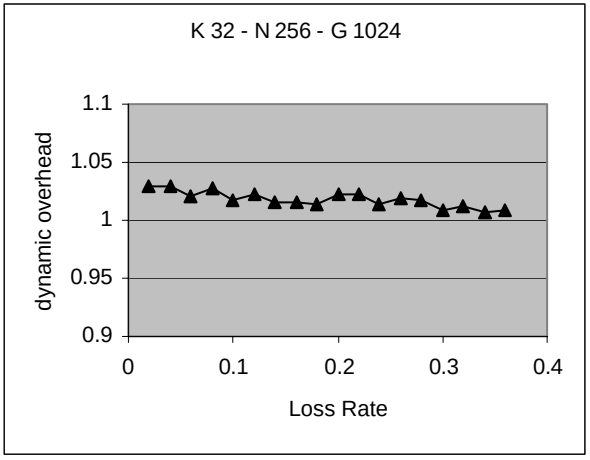
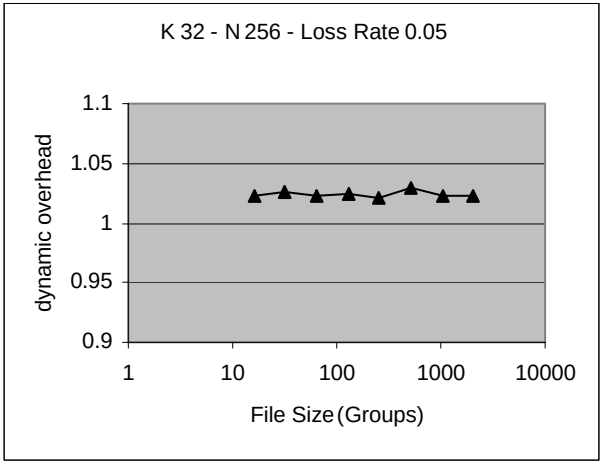


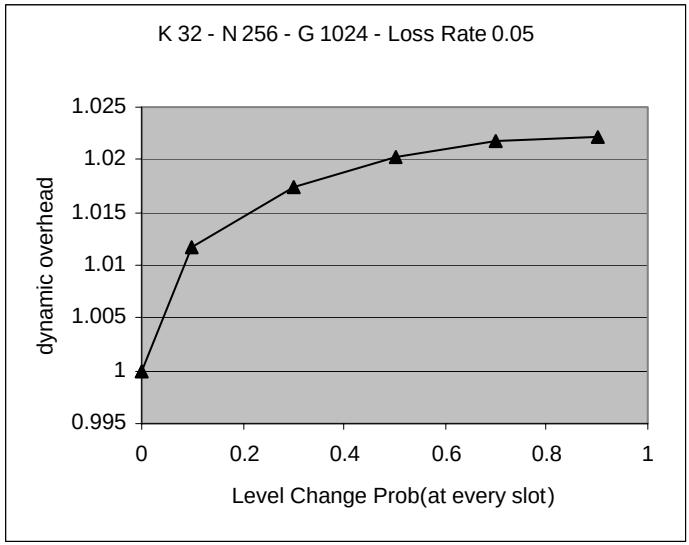
Figure 6.3 on page 170 shows the network dynamics overhead as a function of the file size. We see again the same qualitative behavior. The overhead is practically constant and around 2.5%.

FIGURE 6.3 - Network Dynamics - Function of File Size



In Figure 6.4 on page 171 we show the network dynamics overhead as a function of the probability of rate change at the end of every time slot. As stated above, all the previous simulations were done using a high probability for the rate change (0.45+0.45=0.9). Even with this high amount of changes, the results shown in the figures above are very satisfactory. As expected, when we lower the rate change probability, the overhead becomes even smaller

FIGURE 6.4 - Network Dynamics - Function of Rate Changes

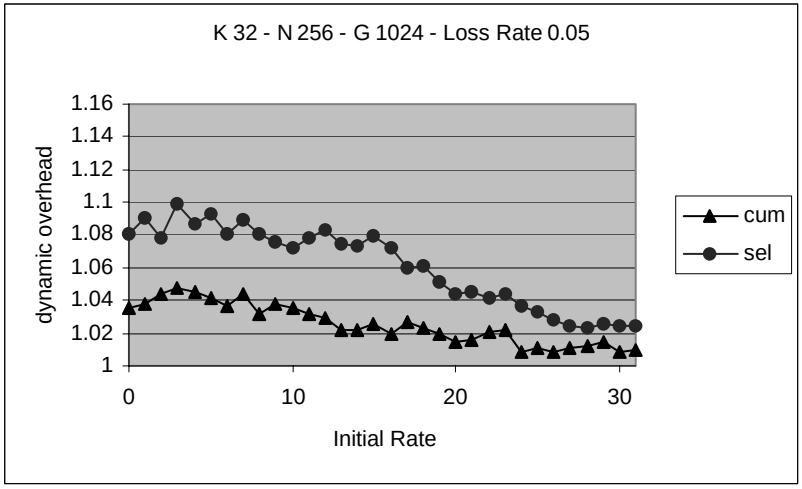


Finally, we show in Figure 6.5 on page 172 the effect of the initial subscription rate in the dynamic network overhead. We plot the results for both the sampled channel scheme (cumulative) and the selective (non-cumulative) approaches.

As can be seen, the sampled channels give somewhat better results (lower overhead) than the non-cumulative ones since on the sampled channels when adding or removing one unit of bandwidth the new received stream is pretty close to the previous one. With non-cum exponential channels, a change in the reception rate by one unit (such as from 8 to 9) implies changing almost all channels and therefore the impact on the overall received packet schedule is bigger.

An additional property that can be appreciated in Figure 6.5 on page 172 is that results are better when the average subscription rate is higher. This is related to the fact that in such cases the impact of the rate changes is smaller. When changing from rate 1 to rate 2 and back we are halving or duplicating the rate. When moving among higher rates the relative change is smaller and therefore the impact on the scheduling is reduced.

FIGURE 6.5 - Network Dynamics - Function of Initial Rate



Network Dynamics Conclusions

We finally conclude that the overhead in reception time that results from network dynamics is reasonable and therefore our presented schemes are suitable for practical implementations where attainable rate varies with time during the file reception.

CHAPTER 7 *Summary*

In this work we dealt with the problem of multicast distribution of very large bulk data files to a large and number of concurrent, non-synchronized, heterogeneous receivers. Our goal was to devise a mechanism that, from the perspective of every receiver, achieves results comparable to those of a unicast reliable transmission in the sense that the number of packets received by any given client until it can reconstruct the entire file should be equal to that of a private reliable point-to-point scheme, or nearly so. At the same time, our objective was to develop solutions that result in optimal network utilization. We looked for transmission schemes for which the load in each link of the multicast routing tree would never be greater than what it would have been if only the fastest receiver downstream that link were downloading the file using a point-to-point reliable protocol.

For scalability reasons, our proposed schemes do not involve any kind of feedback from clients to server. Packet loss is handled through the use of redundancy in the transmission. We analyzed the simplest of those no-feedback schemes as seen from a client. We proved that cyclic scheduling of all the packets in the file is the best possible approach when error correction techniques are not available. We devised an approximation for the expressions that model the performance, from which it can be appreciated that the file size is a significant factor affecting the results. We backed our statistical models with simulations, and compared the results for this simple approach to those of an ideal selective retransmission point-to-point protocol. We have also shown that burst type correlation among packet losses improves the results of the simple cyclic scheme.

We then analyzed a more advanced scheme that involves the use of publicly available forward error correction (FEC) techniques for overcoming packet loss. For practical reasons, related to the implementation of these codes, the file was divided into groups, each with a tractable number of packets. We proved that the best packet schedule under such a scheme is one whereby the coded packets are sent in a cyclic group-interleaved manner. Using a statistical model, we have shown the significant benefits attained by the use of FEC

as compared to the previous simple approach. We have also shown that the packet schedule has a large impact on the performance. This was done by comparing the results attained with the group interleaving schedule to those of a random schedule of packets.

We backed our statistical model with simulations, and also compared the results attained by this technique to those of an ideal unicast reliable protocol. We showed comparable performance for a wide relevant range of file sizes and average loss rates. We showed that, for files of tens of Megabytes and average loss rates around 8%, the time needed by a receiver to complete reception of a file under the proposed scheme is greater than the time that would have taken it to complete with an ideal reliable unicast protocol by no more than 20%. We claim these results to be highly satisfactory considering that in practice, the reliable unicast approach is not applicable as it would result in a collapse of the network when concurrently attempted by a huge number of receivers. Alternative solutions achieving results similar to those of reliable unicast, such as [23], involve the use of proprietary error correction codes. Our solutions, in contrast can be implemented using publicly available codes and related hardware, thereby reducing cost and fostering interoperability and openness.

We developed a bound for the mathematical expressions that model the performance of the described group interleaving approach. These bounds are closer than the ones that can be attained using known methods such as the DeMoivre-Laplace approximation. We further simplified this bound to achieve an approximation that behaves very closely to the calculated expression and provides meaningful insights about the impact of the different parameters on the performance results.

Next, we addressed the issues that stem from the heterogeneous characteristics of the receivers. Different connection rates, locations in the network and processing capabilities such as local storage access rate, result in different attainable effective rates for each of the clients. We used a scheme whereby the server sends the file using multiple channels in a way that, by subscribing to a specific set of them, each client effectively tunes its own reception rate. This so-called receiver driven flow control was combined with a cumulative subscription policy, which we proved to attain optimal network utilization.

In order to maintain the interleaving properties of the combined packet stream as seen by all clients regardless of their current reception rate, we assigned a geometric sequence of transmission rates to the channels, and co-developed carefully crafted packet schedules for every one of them. These co-designed per-channel packet schedules are one of the main contributions of the present work. We proved the perfect interleaving properties of our packet schedule for any number of clients concurrently receiving at different subscription rates and regardless of their starting time. We have thus shown a scheme with comparable results to those of unicast reliable transmission from a receiver's perspective, while attaining optimal network resource utilization as desired.

We compared the results of the presented packet schedule to those attainable using an unpublished scheme defined in [29] that came to our attention lately. We observed our schedule to allow the selection of higher

reception rates for arbitrary file sizes than the ones allowed by the scheme in [29]. We have also shown our scheme, unlike the other one, to have graceful degradation beyond the range for which it is proved optimal.

We then extended our multi-channel mechanism so that clients could obtain a higher resolution in the selection of their subscription rate. We presented a channel-sampling technique for our packet schedule. This technique achieves near-optimal results (about 2% overhead) from a client's perspective while maintaining the cumulative subscription rule that guarantees optimal network utilization. We explained the near optimal results by proving a relaxed interleaving property of the packet schedule that is maintained after the sampling. Alternatively, we proposed a non-cumulative policy for increasing the rate selection resolution that also attains near optimal results (about 2% overhead) with far fewer channels. We proved that the network resources over-utilization that results from using this non-cumulative subscription policy is bounded by a factor of two, which is very reasonable in the framework of multicast transmissions to vast receiver groups that share network resources with TCP-like point-to-point flows.

We further extended our non-cumulative subscription mode with the introduction of a router consolidation heuristic that combines the channel subscription requests from downstream agents to attain optimal network utilization. We described our consolidation algorithm and proved its main properties of optimal utilization and rate-selection resolution improvements. The application of the presented consolidation mechanism is not restricted to our packet schedule scheme. Rather, it is suitable to other multicast distribution methods including those based on proprietary error correction codes [28] which need to provide a way for receivers to subscribe to different rates with optimal utilization of network resources. We built a random tree generator and simulated the consolidation algorithm for very large receiver sets to show very satisfactory results that back our analysis.

Lastly, we evaluated the impact of network dynamics on the performance results attained so far. We simulated a scenario wherein the subscription rate for a client is changed during its reception time, and measured the effect of this rate change on its perceived performance results. We have shown that the impact of rate changes is considerably low (about 2.5%) and therefore claim that our proposed methods are applicable to the practical case wherein network dynamics affect the rate selection.

Summarizing, we presented a receiver driven multi-rate bulk-data distribution scheme with the following properties:

- Attains results comparable to those of reliable unicast protocols from a receiver's perspective.
- Scales to any number of receivers as it makes no use of feedback.
- Optimal in the utilization of network resources.
- Uses standard publicly available erasure correction codes
- Supports dynamic changes in the subscription rate with no significant impact on the client's performance.
- Provides fine-grain resolution in the selection of the desired reception rate (allowing also TCP friendly congestion control).

References and Bibliography

References

[1] S. Deering, "Host Extensions for IP Multicasting", RFC1112, August 1989.

[2] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

[3] D. Waitzman, C. Partridge, S. Deering, “Distance Vector Multicast Routing Protocol”, RFC 1075, November 1988.

[4] J. Moy, "Multicast Extensions to OSPF" RFC 1584, March 1994.

[5] A. Agrawal P. Klein, R. Ravi, “When trees collide: An approximation algorithm for the generalized Steiner Problem on Networks”, CS-90-32, Department of Computer Science Brown University, June 1994.

[6] Metzner, J., "An Improved Broadcast Retransmission Protocol", IEEE Transactions on Communications, Vol. 32, No. 6, June 1984, pp. 679-683.

[7] J-M. Chang, N. F. Maxemchuk, "Reliable Broadcast Protocols," ACM Transactions on Computer Systems, Vol. 2, No. 3, Aug. '84, pp. 251-273.

[8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L, and Zhang, “A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing”, IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803

[9] D. Towsley, J. Kurose, S. Pingali, “A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", IEEE Journal on Selected Areas in Communications, VOL. 15, NO. 3, April 1997, pp 398-406

[10] 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, April 1997, pp 277-290

[11] P. Sanjoy, K. Sabnani, J. Lin, S. Bhattacharyya, ``Reliable Multicast Transport Protocol (RMTP)", IEEE Journal on Selected Areas in Communications, VOL. 15, NO. 3, April 1997, pp 407-421

[12] S. McCanne, V. Jacobson, M. Vetterli. “Receiver-driven Layered Multicast”, In Proceedings of ACM Sigcomm, 1996.

[13] S.K. Kasera, J.F. Kurose, D. Towsley. "Scalable Reliable Multicast Using Multiple Multicast Groups.", Proceedings ACM Sigmetrics, 1997.

[14] S. Bhattacharyya, J. F. Kurose, D. Towsley, R. Nagarajan, ``Efficient Rate- Controlled Bulk Data Transfer using Multiple Multicast Groups", In Proc. of INFOCOM '98, San Francisco, April 1998.

[15] L. Vicisano, L. Rizzo, and J. Crowcroft. ``TCP-like congestion control for layered multicast data transfer." In Proc. of INFOCOM '98, San Francisco, April 1998.

[16] Michael Luby, Lorenzo Vicisano, Tony Speakman, “Heterogeneous multicast congestion control based on router packet filtering”, RMT[1] meeting, Pisa, March 1999.

[17] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, W. Shaver, “FLID-DL: Congestion Control for Layered Multicast”, Proc. of the Second International Workshop on Networked Group Communication (NGC 2000), Stanford, CA, November 2000, pp. 71-81

[18] Eve Schooler, Jim Gemmell, "Using Multicast FEC to Solve the Midnight Madness Problem", MSR-TR-97-25, Microsoft Corporation

1. RMT is the “Reliable Multicast Transport” working group of the IETF (Internet Engineering Task Force).

[19] S. Acharya, M. Franklin, S. Zdonik, ``Dissemination-based Data Delivery Using Broadcast Disks", IEEE Personal Communications, 2(6), December, 1995

[20] Luigi Rizzo, Lorenzo Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques", Proceedinngs of the Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems", HPCS'97, Chalkidiki, Greece, June 1997.

[21] D. Rubenstein, J. Kurose, D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction", Technical Report 98-19, Department of Computer Science, University of Massachusetts Amherst, MA 01003, March 1998

[22] J. Nonnenmacher, E. Biersack, Don Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", ACM SIGCOMM '97, Sept. 1997 Cannes, France, pp. 289-300. (also available as Technical Report 97-17, Department of Computer Science, University of Massachusetts Amherst, MA 01003, March 1997)

[23] J. Byers, M. Luby, Michael Mitzenmacher, Ashutosh Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", Proceedings of ACM SIGCOM '98, Vancouver, B.C., September 1998. Also available as TR-98-013 (UC BERKLEY International Computer Science Institute ICSI).

[24] J. Gemmell, ``ECSRM -- Erasure Correcting Scalable Reliable Multicast," Microsoft Research Technical Report MS-TR-97-20, June 1997.

[25] J. Nonnenmacher, E.W.Biersack, ``Reliable Multicast: Where to use Forward Error Correction", Proc. IFIP 5th Int'l Workshop on Protocols for High Speed Networks, pp.134-148, Sophia Antipolis, France, Oct.1996.

[26] R.E.Blahut, ``Theory and Practice of Error Control Codes" Addison Wesley, MA, 1984

[27] Luigi Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM Computer Communication Review, Vol. 27, n.2, Apr 97, pp 24-36.

[28] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann, "Practical Loss-Resilient Codes." In Proceedings of the 29th ACM Symposium on Theory of Computing, 1997.

[29] L.Vicisano, ``Notes on a cumulative layered organisation of data packets across multiple streams with variable-rate", unpublished notes

[30] J. Byers, M. Luby, Michael Mitzenmacher, Fine-Grained Layered Multicast, Proceedings of IEEE INFOCOM 2001, Anchorage, April 2001.

[31] L.Vicisano, M. Handley, J.Crowcrof, ``B-MART, Bulk-data (non real time) Multiparty Adaptive Reliable Transfer Protocol '', ftp://cs.ucl.ac.uk/darpa/bulk.ps.Z

[32] S. Ross, “Probability Models for Computer Science”, to be published by Harcourt/Academic Press, Burlington MA, USA

Other Bibliography

[33] R. Yavatkar, J. Griffoen and M. Sudan, ``A Reliable Dissemination Protocol for Interactive Collaborative Applications.'' In Proceedings of ACM Multimedia '95, San Francisco, 1995, pp. 333-344.

[34] Z. Wang, J. Crowcroft, C. Diot, A. Ghosh, "Framework For Reliable Multicast Application Design", HIPPARCH ‘97

[35] J. Nonnenmacher, M. Lacher, M. Jung, G. Carl, and E.W. Biersack, ``How Bad is Reliable Multicast Without Local Recovery?'' In Proc. of INFOCOM '98, San Francisco, April 1998.

[36] "Reliable IP Multicast - MFTP Overview", Stardust Forum Report, Stardust Technologies Inc.

[37] "Reliable IP Multicast - PGM Overview", Stardust Forum Report, Stardust Technologies Inc.

[38] W. Dang, "Reliable File Transfer in the Muilticast Domain", University of Hawaii Technical Report, August 1993

[39] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, ``Broadcast Disks: Data Management for Asymmetric Communication Environments", Proc. ACM SIGMOD Conf., San Jose, CA, May, 1995

[40] M. Franklin, S. Zdonik, ``A Framework for Scalable Dissemination-Based Systems", Proc. ACM OOPSLA Conf. (Invited Paper), Atlanta, GA, October, 1997.

[41] M. Ammar, K. Almeroth, R. Clark, and Z. Fei, "Multicast Delivery of Web Pages OR How to Make Web Servers Pushy", Workshop on Internet Server Performance (WISP '98), Madison, Wisconsin, USA, June 1998.

[42] Almeroth, K. C., Ammar, M. H., Fei, Z., ``Scalable Delivery of Web Pages Using Cyclic Best- Effort (UDP) Multicast,'' Proceedings of IEEE INFOCOM 98, March 1998.

[43] M. Yajnik, S. Moon, J. Kurose, D. Towsley, "Measurement and Modelling of the Temporal Dependence in Packet Loss", Technical Report 98-78, Department of Computer Science, University of Massachusetts Amherst, MA 01003

[44] M. Yajnik, J. Kurose, and D. Towsley, ``Packet Loss Correlation in the MBone Multicast Network." In Proceedings of IEEE Global Internet '96, London, November 1996.

[45] L.Vicisano, J.Crowcrof, ``One to Many Reliable Bulk-Data Transfer in the MBone", Proc. of HIPPARCH '97 the Third International Workshop on High Performance Protocol Architectures, Uppsala, Sweden, June 12-13, 1997.

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

[47] S. Armstrong, A. Freier, K. Marzullo, "Multicast Transport Protocol" RFC1301, February 1992.

[48] V. Thomas, "IP Multicast in RealSystem G2", Report, Real Networks Inc.

[49] M. Handley, S. Floyd, B. Whetten, R. Kermode, L. Vicisano, M. Luby, "The Reliable Multicast Design Space for Bulk Data Transfer", RFC 2887, August 2000.

APPENDIX A

Assorted Proofs

A.1 - Simple Cyclic Multicast Distribution of Bulk Data

Optimality of the Simple Cyclic Schedule for the Whole Cycle Model

We denote with $d_{c,i}$ the number of times that packet i has been sent after exactly c cycles. Clearly, $d_{c,i}$ fully defines the transmission schedule. Our desire is to find the optimum schedule:

(EQ 1)

$$d_{c,i} \quad c \in \{0,1,2 \cdots \infty\} \quad i \in \{0,1,2 \cdots G-1\}$$

so that the average reception time for a client:

(EQ 2)

$$E(N_G) = \sum_{c=0}^{\infty} (1 - P(N_G \le c)) = \sum_{c=0}^{\infty} \left(1 - \prod_{i=0}^{G-1} \left(1 - q^{d_{c,i}}\right) \right)$$

is minimized.

Subject to:

(EQ 3)

$$\forall c \quad \sum_{i=0}^{G-1} d_{c,i} = c \cdot G$$

and:

(EQ 4)

$$\forall c \quad d_{c,i} \le d_{c+1,i}$$

Theorem A.1 - Optimality of Packet Interleaving for the Whole Cycle Model

The optimal packet scheduling from a receiver perspective (the one that minimizes the average reception time) is:

(EQ 5)

$$d_{c,i} = c$$

Proof:

In order to use Lagrange Multipliers we define:

(EQ 6)

$$H(d_{c,i}, \lambda_c) = \sum_{c=0}^{\infty} \left(\left(1 - \prod_{i=0}^{G-1} \left(1 - q^{d_{c,i}} \right) \right) - \lambda_c \left(\sum_{i=0}^{G-1} d_{c,i} - G \cdot c \right) \right)$$

For now we leave the second constraint out on purpose.

(EQ 7)

$$\frac{\partial H}{\partial d_{c,i}} = \ln q \cdot q^{d_{c,i}} \cdot \prod_{\substack{j=0 \\ j \neq i}}^{G-1} \left(1 - q^{d_{c,j}} \right) - \lambda_c = 0 \quad i \in \{0,1,\cdots G-1\}, c \in \{0,1,\cdots \infty\}$$

(EQ 8)

$$\lambda_c = \ln q \cdot q^{d_{c,i}} \cdot \prod_{\substack{j=1 \\ j \neq i}}^{G} \left(1 - q^{d_{c,j}}\right) \qquad i \in \{0,1,\cdots G-1\}, c \in \{0,1,\cdots \infty\}$$

(EQ 9)

$$\ln q \cdot q^{d_{c,i}} \cdot \prod_{\substack{j=1 \\ j \neq i}}^{G} \left(1 - q^{d_{c,j}}\right) = \ln q \cdot q^{d_{c,k}} \cdot \prod_{\substack{j=1 \\ j \neq k}}^{G} \left(1 - q^{d_{c,j}}\right) \qquad k,i \in \{0,1,\cdots G-1\} \;\; c \in \{0,1,\cdots \infty\}$$

clearly:

(EQ 10)

$$q^{d_{c,i}} \cdot \left(1 - q^{d_{c,k}}\right) = q^{d_{c,k}} \left(1 - q^{d_{c,i}}\right) \qquad k,i \in \{0,1,\cdots G-1\} \;\; c \in \{0,1,\cdots \infty\}$$

Assuming *q* is not 1, we conclude:

(EQ 11)

$$q^{d_{c,i}} = q^{d_{c,k}} \Rightarrow d_{c,i} = d_{c,k} = d_c$$

Using the first constraint:

(EQ 12)

$$\forall c \quad \sum_{i=0}^{G-1} d_{c,i} = \sum_{i=0}^{G-1} d_c = c \cdot G \Rightarrow d_c = c$$

And our second constraint is satisfied as well.

qed

Optimality of the Simple Cyclic Schedule for the Partial Cycle Model

To show the optimality of the proposed scheduling using the partial cycle model we use an alternative approach. (the use of lagrange multipliers is not straightforward here since the variables cannot take non integer values and this is what would have resulted from the previous proof scheme applied in this case).

Theorem A.2 - Optimality of Packet Interleaving for the Partial Cycle Model

The packet schedule, denoted by *O* with *Oi* being the packet sent in position i, defined as:

(EQ 13)

$$O_i = i \bmod G \qquad i = 0,1,2,\cdots\infty$$

minimizes the average reception time for all clients regardless of their starting reception time.

Proof:

We will use two packet schedules in the proof denoted *C* and *B*.

We denote with:

(EQ 14)

$$N_S^G$$

the random variable that represents the number of packets transmitted until successful reception of the *G* packets in the file under scheduling *S*. Where *S* is one of: *O* (the optimal packet schedule), *C* or *B*.

We further define *F* to be the number of times a specific packet *g* has been sent during the first k packet-transmissions of the schedule S.

(EQ 15)

$$I_{S_i}^{G,g} \equiv \begin{Bmatrix} 1 & S_i = g \\ 0 & S_i \neq g \end{Bmatrix} \; g \in \{0,1,\cdots G-1\} \, i \in \{0,1,\cdots\infty\}$$

(EQ 16)

$$F_{S,k}^{G,g} \equiv \sum_{i=0}^{k-1} I_{S_i}^{G,g} \;\; g \in \{0,1,\cdots G-1\} \, k \in \{0,1,\cdots\infty\}$$

We will show that for any scheduling scheme *C* (the candidate schedule) with:

(EQ 17)

$$C_i \in \{0,1,2,\cdots G-1\} \quad i = 0,1,2,\cdots\infty$$

different from *O*, there exists another scheduling *B* (the better schedule) with:

(EQ 18)

$$B_i \in \{0,1,2,\cdots G-1\} \quad i = 0,1,2,\cdots\infty$$

such that *B* results in a smaller average than *C*.:

(EQ 19)

$$\forall C \neq O \ \exists B \ / E\left(N_B^G\right) < E\left(N_C^G\right)$$

Let us divide the scheduling scheme into cycles of *G* packets each. Since *C* is different from *O* there is at least one cycle of *G* packets for which in *C* there is at least more than one instance of the same packet. Let us pick the first such cycle.

(EQ 20)

$$C \neq O \Rightarrow \exists c,i,j,k \ / C_{G\cdot c+j} = C_{G\cdot c+k} = i \quad c \in \{0,1,2,3,\cdots\infty\} \ \ i,j,k \in \{0,1,2,\cdots G-1\}$$

Case 1: (*c*=0) The first cycle in which there is more than one instance of the same packet is the first transmitted cycle. Let us look now at the first point in the proposed scheduling *C* at which for the first time all the packets have been sent at least once. Let us denote the number of packets sent until this point with *m*.

(EQ 21)

$$m = \min_{i \in \{0,1,2,\cdots\infty\}} / \forall j \in \{0,1,2,\cdots G-1\} \exists k \leq i \,/\, C_k = j$$

Note that if *C* is such that *m* does not exist (i->infinite) then our suggested *O* is of course better than *C* since E(*C*) is infinite. So *B* will chosen to be equal to *O* in such a case.

If *m* exists, our suggested improved scheduling *B* will send the exact same packets as *C* after the first *m* packets. For the first *m* our suggested improvement will reorder them so that the first G packets are one of each and the remaining *m*-*G* are randomly ordered.

(EQ 22)

$$B_k = \begin{cases} k & 0 \le k < G \\ \text{any order from the packets in } C \text{ not used in the first } G. & G \le k \le m \\ C_k & m < k \end{cases}$$

The probability of success at or before packet *i* is received for *i* greater than or equal to *m* is the same for both schemes. However for *C* the probability of success at or before packet *i* for *i* smaller than *m* is zero where for the suggested improved scheme it is greater than zero for all *i* greater than or equal than *G*-1.

(EQ 23)

$$\begin{aligned} P\left(N_B^G \le k\right) &= P\left(N_C^G \le k\right) = 0 && 0 \le k < G-1 \\ P\left(N_B^G \le k\right) &> P\left(N_C^G \le k\right) = 0 && G-1 \le k < m \\ P\left(N_B^G \le k\right) &= P\left(N_C^G \le k\right) && m \le k \end{aligned}$$

Case 2: (*c*<>0) The first cycle in which there is more than one instance of the same packet is not the first transmitted cycle. Let us pick within cycle *c* the first time that a packet is transmitted for the second time and denote its position in the packet stream with *n*.

(EQ 24)

$$n = \min_{i \in \{cG, cG+1, cG+2, \cdots cG+G-1\}} / \exists j \in \{c \cdot G, c \cdot G+1, c \cdot G+2, \cdots i-1\} / C_i = C_j$$

Let us go further down the packet stream and denote with *m* the first packet position after *n* for which a packet that has not yet been sent on cycle *c* is transmitted in *C*.

(EQ 25)

$$m = \min_{i \in \{n+1, n+2, \cdots, \infty\}} / \forall j \in \{c \cdot G, c \cdot G+1, c \cdot G+2, \cdots i-1\} / C_i \ne C_j$$

(just for closeness: if *C* is such that never sends a packet that has not yet been sent in the current cycle so that position *m* does not exist then its resulting avg is infinite so our suggested optimal scheme is of course better).

The improved scheduling *B* will look exactly like *C* with packets in positions *m*-1 and *m* interchanged.

(EQ 26)

$$B_k = \begin{cases} C_k & k \neq m-1, k \neq m \\ C_m & k = m-1 \\ C_{m-1} & k = m \end{cases}$$

Since the packets sent so far are the same, the probability of success at or before packet *i* is received for *i* other than *m*-1 is the same for both schemes.

(EQ 27)

$$P\left(N_B^G \leq k\right) = P\left(N_C^G \leq k\right) \quad k \neq m-1$$

However, as proved by Lemma A.1 - on page 190, the probability of success at or before packet *m*-1 is greater for *B* than for *C*.

We have shown then for both case 1 and case 2 that:

(EQ 28)

$$P\left(N_B^G \leq k\right) \geq P\left(N_C^G \leq k\right) \quad k = 0,1,\cdots \infty$$

and also that there is at least one *k* for which:

(EQ 29)

$$\exists k \,/\, P\left(N_B^G \leq k\right) > P\left(N_C^G \leq k\right)$$

From this and from:

(EQ 30)

$$E\left(N_S^G\right) = \sum_{k=0}^{\infty}\left(1 - P\left(N_S^G \leq k\right)\right)$$

We get:

(EQ 31)

$$E\left(N_B^G\right) < E\left(N_C^G\right)$$

We have shown so far that for every scheduling *C* other than *O* there is at least one scheduling *B* which results in a better Avg. Since there are infinite schedules we yet need to show that our scheduling improvement scheme converges to *O* when applied to the resulting improved scheduling on and on. This result follows from the construction of *B*.

Clearly when the original *C* is such that it is covered by case 1 the resulting *B* is such that if regarded as a new *C* it will be covered by case 2. Therefore we only need to prove that the *B* proposed for case 2 converges to *O*.

From the construction of *B* we can see that if applying the same construction to *B* the resulting schedule would have resulted in *m* being one less than the one before (until *m* becomes *n*+1, this can happen after the first iteration) and then *n* would have advanced at least one position. After some more iterations *n* would cross a cycle boundary thus incrementing *c* and leaving the previous cycle equal to *O* in that range.

Finally, the scheduling of packets within each cycle follows from the fact that we desire a client to start receiving at any point in time. Obviously the specific identity of each packet is not relevant. However once the packets are named from 0 to G-1 then we need the packet order to be the same in all cycles so that the same properties are achieved when starting at any point in time.

qed

Lemma A.1 -

(EQ 32)

$$P\left(N_B^G \le m-1\right) > P\left(N_C^G \le m-1\right)$$

Notation reminder: *G* is the number of packets in the file, *C* is the candidate schedule and *B* is a different one which we prove better than *C* (for any *C* different from *O*).

Proof:

The probability of success at or before k packets have been sent can be written as:

(EQ 33)

$$P\left(N_S^G \le k\right) = \prod_{g=0}^{G-1}\left(1 - e^{F_{S,k}^{G,g}}\right)$$

We can clearly see that:

(EQ 34)

$$P\left(N_C^G \le m-1\right) = \left[\prod_{\substack{g=0 \\ g\neq C_{m-1} \\ g\neq C_m}}^{G-1}\left(1-e^{F_{C,m-1}^{G,g}}\right)\right]\cdot\overbrace{\left(1-e^{c+d}\right)}^{for\ C_{m-1}}\cdot\underbrace{\left(1-e^{c}\right)}_{for\ C_m}$$

where $d>1$. This follows from the definition of m because the packet sent at m was not yet sent in cycle c and the packet sent at m-1 was sent at least c+2 times.

And:

(EQ 35)

$$P\left(N_B^G \le m-1\right) = \left[\prod_{\substack{g=0 \\ g\neq B_{m-1} \\ g\neq B_m}}^{G-1}\left(1-e^{F_{B,m-1}^{G,g}}\right)\right]\cdot\overbrace{\left(1-e^{c+d-1}\right)}^{for\ B_m}\cdot\underbrace{\left(1-e^{c+1}\right)}_{for\ B_{m-1}}$$

then:

(EQ 36)

$$\frac{P\left(N_C^G \le m-1\right)}{P\left(N_B^G \le m-1\right)} = \frac{\left(1-e^{c+d}\right)\cdot\left(1-e^{c}\right)}{\left(1-e^{c+d-1}\right)\cdot\left(1-e^{c+1}\right)} = \\ = \frac{1+e^{2c+d}-e^{c}\left(1-e^{d}\right)}{1+e^{2c+d}-e^{c}\left(e-e^{d-1}\right)}$$

(EQ 37)

$$\left.\begin{array}{l} d>1 \\ e<1 \end{array}\right\} \Rightarrow \left(1-e\right) > e^{d-1}\left(1-e\right) \Rightarrow \\ \Rightarrow \left(1+e^{d}\right) > \left(e+e^{d-1}\right) \Rightarrow \\ \Rightarrow P\left(N_B^G \le m-1\right) > P\left(N_C^G \le m-1\right)$$

qed

A.2 - Multicast with FEC for Bulk Data Distribution

Average Reception Time Analysis

Lemma A.2 - Average for G=1

(EQ 38)

$$E\left(N^{1,K}\right)=\frac{K}{p}$$

Proof:

(EQ 39)

$$E\left(N^{1,K}\right)=\sum_{c=0}^{\infty}\sum_{g=0}^{0}\left[(1\cdot c+g+1)\cdot P_{c,g}^{1,K}\right]=\sum_{c=0}^{\infty}\left[(c+1)\cdot P_{c,0}^{1,K}\right]=$$

$$=\sum_{c=K-1}^{\infty}\left[(c+1)\cdot p\cdot p^{K-1}q^{c-K+1}\frac{c!}{(K-1)!\cdot(c-K+1)!}\right]$$

shifting the counting index and combining some of the terms:

(EQ 40)

$$E\left(N^{1,K}\right)=\sum_{c=K}^{\infty}\left[(c)\cdot p\cdot p^{K-1}q^{c-K}\frac{(c-1)!}{(K-1)!\cdot(c-K)!}\right]=\sum_{c=K}^{\infty}\left[p^{K}q^{c-K}\frac{c!}{(K-1)!\cdot(c-K)!}\right]$$

(EQ 41)

$$E\left(N^{1,K}\right)=\frac{K(1-q)^{K}}{q^{K}}\sum_{c=K}^{\infty}\left[q^{c}\frac{c!}{K!\cdot(c-K)!}\right]$$

Substituting the index according to

(EQ 42)

$$c = j + K$$

we get:

(EQ 43)

$$E\left(N^{1,K}\right) = \frac{K(1-q)^K}{q^K} \sum_{j=0}^{\infty}\left[q^{j+K} \frac{(j+K)!}{K! \cdot j!} \right] = K(1-q)^K \sum_{j=0}^{\infty}\left[q^j \frac{(j+K)!}{K! \cdot j!} \right]$$

Now let us look at the MacLaurin series for the function:

(EQ 44)

$$f(q) = (1-q)^{-(K+1)}$$

(EQ 45)

$$\frac{\partial f(q)}{\partial q} = (K+1)(1-q)^{-(K+2)}$$

(EQ 46)

$$\frac{\partial^2 f(q)}{\partial q^2} = (K+2)(K+1)(1-q)^{-(K+3)}$$

(EQ 47)

$$\frac{\partial^j f(q)}{\partial q^j} = \frac{(j+K)!}{K!}(1-q)^{-(K+1+j)}$$

(EQ 48)

$$f(q) = (1-q)^{-(K+1)} = \sum_{j=0}^{\infty}\left[\frac{\partial^j f(0)}{\partial q^j} \frac{q^j}{j!} \right] = \sum_{j=0}^{\infty}\left[\frac{(j+K)!}{K!} \frac{q^j}{j!} \right]$$

then:

(EQ 49)

$$E\left(N^{1,K}\right)=K(1-q)^{K}(1-q)^{-(K+1)}=K/p$$

qed

Optimality of the Group Interleaving Packet Schedule

Lemma A.3 -

(EQ 50)

$$P\left(N_B^{G,K}\le m-1\right)>P\left(N_C^{G,K}\le m-1\right)$$

Notation reminder: *G* is the number of FEC groups into which the file was divided for transmission. *K* is the number of packet per FEC group (which means the file size in packet units is GK), *C* is the candidat schedule and *B* is a different one which we prove better than *C* (for any *C* different from *O*).

Proof:

Clearly:

(EQ 51)

$$P\left(N_S^{G,K}\le m-1\right)=P\left(N_S^{G,K}\le m-2\right)+P\left(N_S^{G,K}=m-1\right)$$

Since:

(EQ 52)

$$P\left(N_C^{G,K}\le m-2\right)=P\left(N_B^{G,K}\le m-2\right)$$

It is enough for us to prove that:

(EQ 53)

$$P\left(N_B^{G,K}=m-1\right)>P\left(N_C^{G,K}=m-1\right)$$

We know from the model that:

(EQ 54)

$$P\left(N_C^{G,K}=m-1\right)=\left[\prod_{\substack{i=0\\ i\neq C_{m-1}\\ i\neq C_m}}^{G-1}P\left(N_C^{1,K}\le F_{C,m-1}^{G,i}\right)\right]\cdot\overbrace{P\left(N_C^{1,K}=c+d\right)}^{for\,C_{m-1}}\cdot\underbrace{P\left(N_C^{1,K}\le c\right)}_{for\,C_m}$$

and:

(EQ 55)

$$P\left(N_B^{G,K}=m-1\right)=\left[\prod_{\substack{i=0\\ i\neq B_{m-1}\\ i\neq B_m}}^{G-1}P\left(N_B^{1,K}\le F_{B,m-1}^{G,i}\right)\right]\cdot\overbrace{P\left(N_B^{1,K}=c+1\right)}^{for\,B_{m-1}}\cdot\underbrace{P\left(N_B^{1,K}\le c+d-1\right)}_{for\,B_m}$$

Where $d>1$ (from the definition of m).

(EQ 56)

$$\frac{P\left(N_C^{G,K}=m-1\right)}{P\left(N_B^{G,K}=m-1\right)}=\frac{\overbrace{\left[\left(1-e\right)\cdot\binom{K-1}{c+d-1}\left(1-e\right)^{k-1}e^{c+d-k}\right]}^{P\left(N_C^{1,K}=c+d\right)}\cdot\overbrace{\left[\sum_{j=K}^{c}\binom{j}{c}\left(1-e\right)^{j}e^{c-j}\right]}^{P\left(N_C^{1,K}\le c\right)}}{\underbrace{\left[\left(1-e\right)\cdot\binom{K-1}{c}\left(1-e\right)^{k-1}e^{c+1-k}\right]}_{P\left(N_C^{1,K}=c+1\right)}\cdot\underbrace{\left[\sum_{j=K}^{c+d-1}\binom{j}{c+d-1}\left(1-e\right)^{j}e^{c+d-1-j}\right]}_{P\left(N_C^{1,K}\le c+d-1\right)}}$$

After some trivial simplification:

(EQ 57)

$$\frac{P\left(N_C^{G,K}=m-1\right)}{P\left(N_B^{G,K}=m-1\right)}=\frac{\left[\binom{K-1}{c+d-1}e^{d-1}\right]\cdot\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right]}{\left[\binom{K-1}{c}\right]\cdot\left[e^{d-1}\sum_{j=K}^{c+d-1}\binom{j}{c+d-1}(1-e)^j e^{c-j}\right]}$$

And after some more:

(EQ 58)

$$\frac{P\left(N_C^{G,K}=m-1\right)}{P\left(N_B^{G,K}=m-1\right)}=\frac{\left[\frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)\cdot c!}{(c-K+d)\cdot(c-K+d-1)\cdots(c-K+2)\cdot(c-K+1)!(K-1)!}\right]\cdot\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right]}{\left[\frac{c!}{(c-K+1)!(K-1)!}\right]\cdot\left[\sum_{j=K}^{c+d-1}\binom{j}{c+d-1}(1-e)^j e^{c-j}\right]}$$

And yet a bit more:

(EQ 59)

$$\frac{P\left(N_C^{G,K}=m-1\right)}{P\left(N_B^{G,K}=m-1\right)}=\frac{\left[\frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d)\cdot(c-K+d-1)\cdots(c-K+2)}\right]\cdot\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right]}{\left[\sum_{j=K}^{c+d-1}\frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-j+d-1)\cdot(c-j+d-2)\cdots(c-j+1)}\binom{j}{c}(1-e)^j e^{c-j}\right]}$$

From here:

(EQ 60)

$$
\begin{aligned}
&(K \le j \le c+d-1) \Rightarrow \\
&\Rightarrow \frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-j+d-1)\cdot(c-j+d-2)\cdots(c-j+1)} \ge \\
&\ge \frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d-1)\cdot(c-K+d-2)\cdots(c-K+1)} \Rightarrow \\
&\Rightarrow \left[\sum_{j=K}^{c+d-1} \frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-j+d-1)\cdot(c-j+d-2)\cdots(c-j+1)}\binom{j}{c}(1-e)^j e^{c-j}\right] \ge \\
&\ge \frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d-1)\cdot(c-K+d-2)\cdots(c-K+1)}\left[\sum_{j=K}^{c+d-1}\binom{j}{c}(1-e)^j e^{c-j}\right] \overset{d\ge 1}{>} \\
&> \frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d-1)\cdot(c-K+d-2)\cdots(c-K+1)}\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right] \overset{d\ge 1}{>} \\
&> \frac{(c-K+1)}{(c-K+d)}\cdot\frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d-1)\cdot(c-K+d-2)\cdots(c-K+1)}\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right] = \\
&= \left[\frac{(c+d-1)\cdot(c+d-2)\cdots(c+1)}{(c-K+d)\cdot(c-K+d-1)\cdots(c-K+2)}\right]\cdot\left[\sum_{j=K}^{c}\binom{j}{c}(1-e)^j e^{c-j}\right]
\end{aligned}
$$

qed

A.3 - Multi-rate Distribution to Heterogeneous Clients

Group Interleaving Properties

Lemma 4.1 - Number of Slots needed for G packets at Level l (Page 98)

The number of slots required to receive exactly G packets under subscription level l is 2^{J-l}.

Proof:

The number of packets per slot is the sum of the number of packets at each channel up to channel *l* in one slot:

(EQ 61)

$$1+\sum_{j=1}^{l}2^{j-1}=1+\frac{2^{l}-1}{2-1}=2^{l}$$

clearly then in 2^{J-l} slots we have:

(EQ 62)

$$2^{l}\cdot 2^{J-l}=2^{J}=G$$

packets.

qed

Lemma 4.2 - - Modulo Elimination (Page 99)

Starting at slot zero, when looking at *G* consecutive packets, the modulo in the group index formula above can be eliminated. Therefore, the group index formula can be rewritten as follows.

Given z=0 then:

(EQ 63)

$$\underset{j=0}{g}=s$$

(EQ 64)

$$\underset{0<j\le l}{g}=s+2^{J-j}+2^{J-j+1}\cdot t$$

Proof:

For **(*j*=0):** When starting at slot *s=0*, *s* is trivially not smaller than 0. The maximum value that s can attain when sending no more then G packets is given by on page 197:

(EQ 65)

$$s \in \left\{0,1,\cdots,2^{J-l}-1\right\} \quad 0 \le l \le J$$

Clearly the maximum value for the expression above is attained when *l=0*. In such a case the maximum s is $2^{J-0}-1=G-1<G$.

then:

(EQ 66)

$$g = \underset{j=0}{|s|_G} = s$$

For (*j*>0): Provided we start at slot *s=0* and send no more than G packets, from on page 197 and the definitions of *j* and *t*:

(EQ 67)

$$\begin{aligned} s &\in \left\{0,1,\cdots,2^{J-l}-1\right\} \\ j &\in \left\{1,2,\cdots,l\right\} \\ t &\in \left\{0,1,\cdots,2^{j-1}-1\right\} \end{aligned}$$

g is clearly not smaller than 0.

Let us see what is the maximum value of *g* that can be attained given the restricted domain for the parameters *s*,*j* and *t*.

(EQ 68)

$$\begin{aligned} &\max\left(s+2^{J-j}+2^{J-j+1}\cdot t\right)= \\ &=2^{J-l}-1+2^{J-j}+2^{J-j+1}\cdot\left(2^{j-1}-1\right)= \\ &=2^{J-l}-1+2^{J-j}+2^{J}-2^{J-j+1}= \\ &=2^{J-l}-1-2^{J-j}+2^{J}=G-1+2^{J-l}-2^{J-j} \end{aligned}$$

and since

(EQ 69)

$$j \le l$$

then:

(EQ 70)

$$G-1+2^{J-l}-2^{J-j} \le G-1 < G$$

therefore:

(EQ 71)

$$\underset{0<j\le l}{g} = \left| s+2^{J-j}+2^{J-j+1}\cdot t \right|_G = s+2^{J-j}+2^{J-j+1}\cdot t$$

qed

Lemma 4.3 - Uniqueness of the j,s,t representation of g within a range of G packets at any subscription level l. (Page 103)

For any group denoted with g_k, which can be represented as:

(EQ 72)

$$g_k = \begin{cases} s_k & j_k = 0 \\ s_k + 2^{J-j_k} + 2^{J-j_k+1}\cdot t_k & 0 < j_k \le l \end{cases}$$

with parameters in the range:

(EQ 73)

$$\begin{aligned} s_k &\in \{0,1,\cdots,2^{J-l}-1\} \\ j_k &\in \{0,1,\cdots,l\} \\ t_k &\in \{0,1,\cdots,2^{j_k-1}-1\} \quad j_k > 0 \\ t_k &\in \{0\} \quad j_k = 0 \end{aligned}$$

the representation above is unique which means that:

(EQ 74)

$$(g_1 = g_2) \rightarrow \begin{cases} s_1 = s_2 \\ j_1 = j_2 \\ t_1 = t_2 \end{cases}$$

Proof:

For *s*:

From the binary representation, the coefficients of g_k for $i < J\text{-}l$ are the same as the ones of s_k independent of j_k and t_k:

(EQ 75)

$$\forall_{k=0,1} j_k, t_k \quad b(g_k, i)_{0 \le i < J-l} = b(s_k, i)$$

At the same time, since the binary representation of a number is unique:

(EQ 76)

$$(g_1 = g_2) \rightarrow b(g_1, i)_{0 \le i < J} = b(g_2, i)$$

then:

(EQ 77)

$$\forall_{k=0,1} j_k, t_k \left[b(s_1, i)_{0 \le i < J-l} = b(s_2, i) \right] \rightarrow s_1 = s_2$$

For *j*: (we will show that if $j_1 != j_2$ then $g_1 != g_2$)

Let us assume with no loss of generality that $j_1 > j_2$, from the binary decomposition above we get:

(EQ 78)

$$(j_1 > j_2) \rightarrow (j_1 > 0) \rightarrow$$
$$\rightarrow b(g_1, J - j_1) = 1$$

however for g_2:

(EQ 79)

$$\left(J-l \le J-j_1 < J-j_2\right) \rightarrow b\left(g_2, J-j_1\right)=0$$

which results in:

(EQ 80)

$$\left[b\left(g_1, J-j_1\right) \neq b\left(g_2, J-j_1\right)\right] \rightarrow \left(g_1 \neq g_2\right)$$

so:

(EQ 81)

$$\left(g_1=g_2\right) \rightarrow \left(j_1=j_2\right)$$

For *t*: We have already seen that $j_1=j_2$ so we only need to prove $t_1=t_2$ for $j>1$ (since for $j=0$ and $j=1$ *t* is always 0 so trivially $t_1=t_2$).

From the binary decomposition:

(EQ 82)

$$\forall_{k=0,1} j>1 \; b\left(g_k, i\right)=b\left(t_k, i-J+j-1\right)_{J-j<i<J}$$

at the same time:

(EQ 83)

$$\left(g_1=g_2\right) \rightarrow b\left(g_1, i\right)=b\left(g_2, i\right)_{0 \le i<J}$$

then:

(EQ 84)

$$\forall_{k=0,1} j>1 \left[b(t_1, i-J+j-1) = b(t_2, i-J+j-1) \underset{J-j<i<J}{} \right] \rightarrow$$
$$\rightarrow \left[b(t_1, i) = b(t_2, i) \underset{0 \le i < j-1}{} \right] \rightarrow t_1 = t_2$$

qed

Lemma 4.5 - Modulo Elimination (Generalized G) (Page 106)

In a similar manner as done in Lemma 4.2 - on page 99, we wish to eliminate the modulo in the group index expression. We prove here that when the starting slot is 0, for *G* consecutive packets, the modulo in the group index formula above can be eliminated. Therefore, the group index formula can be rewritten as follows.

(EQ 85)

$$\underset{j=0}{g} = s$$

(EQ 86)

$$\underset{0<j\le l}{g} = s + W2^{J-j} + W2^{J-j+1} \cdot t$$

Proof:

(For j=0) When starting at slot *s=0*, *s* is trivially not smaller than 0. The maximum value that *s* can attain when sending no more then *G* packets is $W2^{J-l}$-1. Clearly the maximum value for this expression is attained when *l=0*. In such a case the maximum s is $W2^{J-0}$-1=*G*-1<*G*.

then:

(EQ 87)

$$\underset{j=0}{g} = \left| s \right|_G = s$$

(For j>0) Provided we start at slot *s=0* and stop at slot $W2^{J-l}$-1 to send *G* packets, from the definitions of *j* and *t*:

(EQ 88)

$$s \in \left\{0,1,\cdots,W\,2^{J-l}-1\right\}$$
$$j \in \left\{1,2,\cdots,l\right\}$$
$$t \in \left\{0,1,\cdots,2^{j-1}-1\right\}$$

g is clearly not smaller than 0.

Let us see what is the maximum value of *g* that can be attained given the restricted domain for the parameters *s*,*j* and *t*.

(EQ 89)

$$\begin{aligned}
&\max\left(s+W2^{J-j}+W2^{J-j+1}\cdot t\right)=\\
&=W2^{J-l}-1+W2^{J-j}+W2^{J-j+1}\cdot\left(2^{j-1}-1\right)=\\
&=W2^{J-l}-1+W2^{J-j}+W2^{J}-W2^{J-j+1}=\\
&=W2^{J-l}-1-W2^{J-j}+W2^{J}=G-1+W2^{J-l}-W2^{J-j}
\end{aligned}$$

and since

(EQ 90)

$$j \le l$$

then:

(EQ 91)

$$G-1+W2^{J-l}-W2^{J-j} \le G-1<G$$

therefore:

(EQ 92)

$$\underset{0<j\le l}{g}=\left|s+W2^{J-j}+W2^{J-j+1}\cdot t\right|_G=s+W2^{J-j}+W2^{J-j+1}\cdot t$$

qed

Lemma 4.6 - Uniqueness of the j,s,t representation of g* within a range of G packets at any subscription level l. (Page 110)

If $g^*(s_1,j_1,t_1)=g^*(s_2,j_2,t_2)$ then $s_1=s_2$, $j_1=j_2$ and $t_1=t_2$.

Proof:

For *s*: If $g^*(s_1,j_1,t_1)=g^*(s_2,j_2,t_2)$ then clearly:

(EQ 93)

$$r\left(g^*\left(s_1,j_1,t_1\right)\right)=r\left(g^*\left(s_2,j_2,t_2\right)\right)$$

From the definition above:

(EQ 94)

$$\left[\frac{s_1}{W2^{J-l}}=\frac{s_2}{W2^{J-l}}\right]\rightarrow\left[s_1=s_2\right]$$

For *j*:

Without loss of generality, if j1!=j2 let us assume j1>j2. Then:

(EQ 95)

$$b\left(\left\lfloor g^*\left(s_1,\,j_1,t_1\right)\right\rfloor,J-j_1\right)=1$$

and:

(EQ 96)

$$b\left(\left\lfloor g^*\left(s_2,\,j_2,t_2\right)\right\rfloor,J-j_1\right)=0$$

which means:

(EQ 97)

$$\lfloor g^*(s_2, j_2, t_2) \rfloor \neq \lfloor g^*(s_2, j_2, t_2) \rfloor$$

But for $g^*(s_1,j_1,t_1)=g^*(s_2,j_2,t_2)$ we need:

(EQ 98)

$$\lfloor g^*(s_1, j_1, t_1) \rfloor = \lfloor g^*(s_2, j_2, t_2) \rfloor$$

and therefore:

(EQ 99)

$$j_1 = j_2$$

For *t*: We have already seen that $j_1=j_2$ so we only need to prove $t_1=t_2$ for $j_1=j_2>1$ (since for $j=0$ and $j=1$ *t* is always 0 so trivially $t_1=t_2$).

From the binary representation above:

(EQ 100)

$$\forall j > 1 \;\; b(g^*, i) = \underset{l-j<i<l}{b(t, i-l+j-1)}$$

at the same time:

(EQ 101)

$$\left[g^*(s, j, t_1) = g^*(s, j, t_2) \right] \rightarrow \left[\underset{0 \le i < l}{b(g^*(s, j, t_1), i) = b(g^*(s, j, t_2), i)} \right]$$

then:

(EQ 102)

$$\forall j > 1 \left[b(t_1, i-l+j-1) = \underset{l-j<i<l}{b(t_2, i-l+j-1)} \right] \rightarrow$$
$$\rightarrow \left[\underset{0 \le i < j-1}{b(t_1, i) = b(t_2, i)} \right] \rightarrow t_1 = t_2$$

qed

Packet Interleaving Properties

Lemma A.4 -

(EQ 103)

$$\left\lfloor \frac{a}{b \cdot c} \right\rfloor \le \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor$$

Proof:

(EQ 104)

$$\left\lfloor \frac{a}{b \cdot c} \right\rfloor = \left\lfloor \frac{a/b}{c} \right\rfloor = \left\lfloor \frac{\lfloor a/b \rfloor + a/b - \lfloor a/b \rfloor}{c} \right\rfloor \le \left\lfloor \frac{\lfloor a/b \rfloor + (b-1)/b}{c} \right\rfloor =$$
$$= \left\lfloor \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor + \frac{\lfloor a/b \rfloor}{c} - \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor + \frac{(b-1)/b}{c} \right\rfloor \le \left\lfloor \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor + \frac{(c-1)}{c} + \frac{(b-1)/b}{c} \right\rfloor =$$
$$= \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor + \left\lfloor \frac{c-1+(b-1)/b}{c} \right\rfloor = \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor + 0 = \left\lfloor \frac{\lfloor a/b \rfloor}{c} \right\rfloor$$

qed

Lemma A.5 -

(EQ 105)

$$\left\lfloor \frac{a}{b \cdot c} \right\rfloor \geq \left\lfloor \frac{\left\lfloor a/b \right\rfloor}{c} \right\rfloor$$

Proof:

(EQ 106)

$$\left[\frac{a}{b} \geq \left\lfloor \frac{a}{b} \right\rfloor \right] \rightarrow \left[\frac{a/b}{c} \geq \frac{\left\lfloor a/b \right\rfloor}{c} \right] \rightarrow \left[\left\lfloor \frac{a/b}{c} \right\rfloor \geq \left\lfloor \frac{\left\lfloor a/b \right\rfloor}{c} \right\rfloor \right]$$

qed

Lemma A.6 -

(EQ 107)

$$\left\lfloor \frac{a}{b \cdot c} \right\rfloor = \left\lfloor \frac{\left\lfloor a/b \right\rfloor}{c} \right\rfloor$$

Proof:

Lemma A.4 - on page 207 and Lemma A.5 - on page 208

qed

Lemma 4.7 - Number of superslots for GN packets (Page 115)

The number of superslots required to receive exactly *GN* packets is 2^{M-l}.

Proof:

The number of packets per superslot is the sum of the number of packets at each channel up to channel *l* in one superslot:

(EQ 108)

$$\underbrace{G}_{\text{packets per mini-superslot}} \cdot \left[\overbrace{1}^{\text{mini superslots in channel 0}} + \sum_{j=1}^{l} \underbrace{2^{j-1}}_{\text{mini super slots in channel } j} \right] =$$

$$= G \cdot \left[1 + \frac{2^l - 1}{2 - 1} \right] = G \cdot 2^l$$

clearly then, in 2^{M-l} superslots we have:

(EQ 109)

$$G \cdot 2^l \cdot 2^{M-l} = G \cdot 2^M = G \cdot N$$

packets.

qed

Lemma 4.8 - All packets have the same packet index within a j-mini-superslot. (Page 117)

For any specific channel *j*, *gg* can be eliminated from the packet index formula within a *j*-mini-superslot which means that all packets within the *j*-mini-super slot have the same packet index. What we wish to prove is formally expressed in (Eq. 110 on page 210) and (Eq. 111 on page 210).

(EQ 110)

$$j > 0 \quad gg \in \{0,1,\cdots,G-1\}$$
$$p = \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G \cdot 2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G} \right\rfloor_{2^{j-1}} \right\|_N =$$
$$= \left\| \left\lfloor \frac{2^{j-1}ss + tt}{2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left| 2^{j-1}ss + tt \right|_{2^{j-1}} \right\|_N$$

and:

(EQ 111)

$$j = 0 \quad gg \in \{0,1,\cdots,G-1\}$$
$$p = \left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor \right\|_N = |ss|_N$$

Proof:

For j>0:

From Lemma A.6 - on page 208

(EQ 112)

$$\left\lfloor \frac{a}{b \cdot c} \right\rfloor = \left\lfloor \frac{\lfloor a / b \rfloor}{c} \right\rfloor$$

then:

(EQ 113)

$$p = \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G \cdot 2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left\| \left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G} \right\rfloor \right\|_{2^{j-1}} \right\|_N =$$

$$= \left\| \left\lfloor \frac{\left\lfloor \frac{G\left(2^{j-1}ss + tt\right) + gg}{G} \right\rfloor}{2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left| 2^{j-1}ss + tt \right|_{2^{j-1}} \right|_N =$$

$$= \left\| \left\lfloor \frac{2^{j-1}ss + tt}{2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left| 2^{j-1}ss + tt \right|_{2^{j-1}} \right|_N$$

which is clearly constant within the *j*-mini-superslot as it is not a function of *gg*.

For *j=0*:

(EQ 114)

$$p = \left\| \left\lfloor \frac{G \cdot ss + gg}{G} \right\rfloor \right\|_N = \left\| \left\lfloor ss + \frac{gg}{G} \right\rfloor \right\|_N = \left| ss + \left\lfloor \frac{gg}{G} \right\rfloor \right|_N = \left| ss + 0 \right|_N = \left| ss \right|_N$$

which is clearly constant within the *j*-mini-superslot as it is not a function of *gg*.

qed

Lemma 4.9 - Each packet index is sent exactly G times (and in the same j-mini-superslot) during GN packets starting at a superslot boundary. (Page 117)

We wish to prove that out of *GN* packets sent at level *l*, starting at a superslot boundary, every packet index is sent exactly *G* times and that these *G* times are within the same single *j*-mini-superslot[1].

Proof:

From on page 209, for *j=0*:

1. since the *G* times are within the same mini-superslot then it means they are all in the same channel (since each mini-superslot belongs to one of the channels in the level)

(EQ 115)

$$p = \left| ss \right|_N$$

and for j>0:

(EQ 116)

$$p = \left\| \left\lfloor \frac{2^{j-1} ss + tt}{2^{j-1}} \right\rfloor + 2^{M-j} + 2^{M-j+1} \cdot \left| 2^{j-1} ss + tt \right|_{2^{j-1}} \right|_N =$$
$$= \left| ss + 2^{M-j} + 2^{M-j+1} \cdot tt \right|_N$$

We can now apply the group interleaving theorem (Theorem 4.5 - on page 111) to the packet index of the first packet in every *j*-mini-superslot where *ss* takes the place of *s*, *tt* takes the place of *t*, *M* takes the place of *J* and *N* takes the place of *G*.

With the application of Theorem 4.5 - on page 111 we obtain that the packet indexes for the first packet sent in a *j*-mini-superslot during *G* consecutive *j*-mini-superslots when starting at a superslot boundary are all different. And we already proved in on page 209 that within the j-mini-superslot the packet index is the same.

qed

Lemma 4.10 - Every j-mini-superslot begins at a slot boundary (Page 118)

Proof:

There are $W2^J$ packets per *j*-mini-superslot and 2^{j-1} packets per slot on channel *j*. So the number of slots per *j*-mini-superslot is:

(EQ 117)

$$\frac{W2^J}{2^{j-1}} = W2^{J-j+1}$$

and since:

(EQ 118)

$$0 \le j \le l \le \min(J, N)$$

then the number of slots per j-mini-superslot is an integer.

Therefore since the first j-mini-superslot starts at a slot boundary every subsequent one will do so as well.

qed

Lemma 4.11 - Within a j-mini-superslot there is exactly one packet from every group. (Page 118)

Proof:

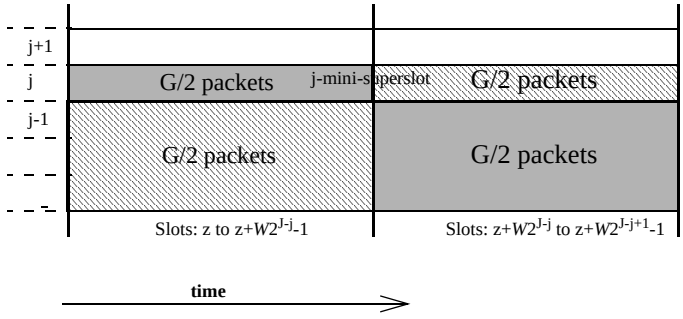
Since a *j*-mini-superslot contains *G* packets, all we need to prove is that all these packets belong to different groups.

For *j*=0, the proof for the group interleaving property (Theorem 4.5 - on page 111) with *l*=0, proves the Lemma (Theorem 4.5 - on page 111 is applicable since every *j*-mini-superslot starts at a slot boundary as proved in on page 212).

For *j*>0 we split the *j*-mini-super slot in half so that we get two sections with *G/2* packets each[2] (Figure A.1 on page 214). For the first half, the conditions of the group interleaving property (Theorem 4.5 - on page 111) hold (with *l=j* and z="first slot of the j-mini-superslot"). So the packets in the first half all belong to different groups. For the second half, the conditions for the group interleaving (Theorem 4.5 - on page 111) hold as well (with *l=j* and z="first slot of the j-mini-superslot"+2^{J-j}) and therefore the packets in the second half belong to all different groups as desired. We yet need to show that there is no packet in the first half that appears on the second one.

2. If j>0 then *l* is for sure greater than 0. This means J is greater than 0 and then we can divide G by two.

FIGURE A.1 - half a *j*-mini-superslot



For that purpose, we resort to the group interleaving property once again. If we apply the group interleaving theorem (Theorem 4.5 - on page 111) for $l=j-1$ we see that the groups of the G/2 packets sent in channels 0 to $j-1$ from slot $z+W2^{J-j}$ to slot $z+W2^{J-j+1}-1$ ought to be different from those sent in channels 0 to $j-1$ from slot z to slot $z+W2^{J-j}-1$. But these G/2 are in turn different from the G/2 packets in the first half of our j-mini-superslot (which follows from applying the group interleaving theorem with $l=j$). Then, the G/2 packets in the first half of our j-mini-superslot belong to the same groups as the G/2 packets in channels 0 to $j-1$ from slot $z+W2^{J-j}$ to slot $z+W2^{J-j+1}-1$. Finally, when applying the group interleaving theorem (Theorem 4.5 - on page 111) with $l=j$ starting at slot $z+W2^{J-j}$ we reach the conclusion that the packets in the second half of our j-mini-superslot belong to different groups than those in channels 0 to $j-1$. Since these last packets belonged to the same groups as those in the first half of our j-mini-superslot then we proved that packets in the second half of our j-mini-superslot are different from those in the first half.

qed

Higher Subscription Rates

Lemma 4.13 - - Modulo Elimination (l=J+1) (Page 124)

(EQ 119)

$$g \underset{j=J+1}{} = s + W \cdot t$$

Proof:

Provided we start at slot $s=0$ and stop at slot $W-1$ to send G packets, from the definitions of j and t:

(EQ 120)

$$s \in \{0,1,\cdots,W-1\}$$
$$j = J+1$$
$$t \in \{0,1,\cdots,2^{j-1}-1 = 2^J-1\}$$

g is clearly not smaller than 0.

Let us see what is the maximum value of *g* that can be attained given the restricted domain for the parameters *s*,*j* and *t*.

(EQ 121)

$$\max(s+W\cdot t) = W-1+W\cdot(2^J-1) =$$
$$= W-1+W2^J-W = W2^J-1 = G-1 < G$$

qed

Lemma 4.14 - Uniqueness of the j,s,t representation of g* within a range of G packets at channel j=J+1. (Page 125)

If $g^*(s_1,J+1,t_1)=g^*(s_2,J+1,t_2)$ then $s_1=s_2$ and $t_1=t_2$.

Proof:

For *s*: If $g^*(s_1,J+1,t_1)=g^*(s_2,J+1,t_2)$ then clearly:

(EQ 122)

$$r(g^*(s_1,J+1,t_1)) = r(g^*(s_2,J+1,t_2))$$

From the definition above:

(EQ 123)

$$\left[\frac{s_1}{W} = \frac{s_2}{W}\right] \rightarrow [s_1 = s_2]$$

For *t*:

(EQ 124)

$$\begin{aligned}
&\left[g^*(s, J+1, t_1) = g^*(s, J+1, t_2)\right] \rightarrow \\
&\rightarrow \left[\left\lfloor g^*(s, J+1, t_1)\right\rfloor = \left\lfloor g^*(s, J+1, t_2)\right\rfloor\right] \rightarrow \\
&\rightarrow t_1 = t_2
\end{aligned}$$

qed

APPENDIX B

Simulation Issues

B.1 - Tree Generation Algorithm

Node Data Structure

```
1   typedef struct stNode
2   {
3       struct stNode *pUp;
4       struct stNode *pDown;
5       WORD wMaxCap;                   //max rate available (slowest link upstream)
6       WORD wRcvCap;                   //rate received
7       WORD wRcvCapCh[MAX_CHANNELS];//for every channel rate currently being received
8       BYTE cLinks;
9       DWORD dwDwRecv;
10      BOOL bCons;
11  }
12  NODE;
```

Generate Tree

This function is the entry point for the tree generation algorithm. It allocates the memory space for the tree database as a contiguous area.

```
13  void GenerateTree(MCTREE *pMCTree, BYTE cMaxDepth,BYTE cMaxLnk,WORD wMaxCap)
14  {
15
16      pRemNodes=(NODE *)malloc(MAX_NODES * sizeof(NODE));
17      dwRemNodes=MAX_NODES;
18
19      pMCTree->bGen=TRUE;
20      pMCTree->cDepth=cMaxDepth;
21      pMCTree->pRoot=GetNodes(1);
22      GenerateNode(pMCTree->pRoot,(NODE *)0,wMaxCap,0,cMaxDepth,cMaxLnk,wMaxCap);
23
24      MyPrintf("Total Receivers:%09d \r\n",(pMCTree->pRoot)->dwDwRecv);
25  }
```

GetNodes

```
26  static NODE *GetNodes(BYTE cNodes)
27  {
28      NODE *pDownNodes;
29
30      if (cNodes < dwRemNodes)
31      {
32              pDownNodes = pRemNodes;
33              pRemNodes += cNodes;
34              dwRemNodes -= cNodes;
35      }
36      else
37      {
38              PrintError(FUNCTION_NAME,"No more nodes available");
39              pDownNodes = (NODE *) 0;
40      }
41
42      return pDownNodes;
43  }
```

GenerateNode

```
44  static void GenerateNode(NODE *pNode, NODE *pParent, WORD wParCap, BYTE cDepth, BYTE cMaxDepth, BYTE cMaxLnk,
WORD wMaxCap)
45  {
46      BYTE n;
47      double r;
48
49      pNode->dwDwRecv=0;
50      pNode->pUp = pParent;
51      r=((double)rand()/RAND_MAX);
52      pNode->wMaxCap = min(wParCap,(WORD)(r * wMaxCap)+1);
53      r=((double)rand()/RAND_MAX);
54      pNode->cLinks = (cDepth < cMaxDepth) ? (BYTE)(r * cMaxLnk) : 0;
55
56
57      pNode->pDown = GetNodes(pNode->cLinks);
58      //REV need true error handling for the case when GetDownNodes completes in error
59      //because no more nodes are available
60      if(pNode->pDown == 0) pNode->cLinks=0;
61
62      //MyPrintf("L:%02d D:%02d C:%05d \r\n",pNode->cLinks,cDepth,pNode->wMaxCap);
63
64      if(pNode->cLinks == 0)
65              pNode->dwDwRecv=1;
66
67      for(n=0;n<pNode->cLinks;n++)
68      {
69              GenerateNode((pNode->pDown)+n,pNode,pNode->wMaxCap,cDepth+1,cMaxDepth,cMaxLnk,wMaxCap);
70              pNode->dwDwRecv += ((pNode->pDown)+n)->dwDwRecv;
71      }
72  }
73
```

B.2 - Implementation Issues

$$F_{i,j} \equiv \frac{(i+j)!}{i!\cdot j!}$$

$$F_{0,j} = F_{j,0} = 1$$

$$F_{i+1,j} = \frac{(i+1+j)!}{(i+1)!\cdot j!} = \frac{(i+1+j)(i+j)!}{(i+1)\cdot i!\cdot j!} = \frac{(i+1+j)}{(i+1)} F_{i,j}$$