A Probabilistic Approach for Estimating the Number of Bad Receivers in a Multicast Group with Application to Hybrid FEC/ARQ-based Reliable Wireless Multicast

Reuven Cohen and Alexander Landau
Dept. of Computer Science
Technion
Israel

ABSTRACT

When a wireless base station needs to transfer a data block to a large multicast group in a reliable way, a hybrid FEC/ARQ-based protocol is usually the best solution. For a large multicast group, having each bad receiver send a NACK to the base station would result in a feedback implosion. In this paper we present an algorithm for estimating the number of bad receivers while limiting the number of NACKs. This algorithm analyzes the NACK transmission times, defines a likelihood function for them, and then uses the Newton-Raphson method to find the number of bad receivers for which this function is maximized. We then analyze the precision of the proposed algorithm, and provide tight upper and lower bounds to the estimation error. These bounds indicate a linear correlation between the estimation error and the number of NACKs. But even more important, they allow us to correct the precision of our estimation, and to bring the error very close to 0. To the best of our knowledge, our paper is the first to explicitly show the correlation between the number of NACKs sent by a group and the ability of the sender to estimate the size of this group precisely.

1. INTRODUCTION

A prominent feature of advanced cellular technologies, like HSDPA, and broadband wireless access technologies like IEEE-802.11 and IEEE-802.16, is the base station’s ability to transmit a single copy of a packet to a group of receivers, a concept known as multicast. In order to ensure some level of reliability, it is often useful for the sender to get negative feedback, NACK, from hosts that have received some of the multicast data incorrectly ("bad receivers"). However, for a large multicast group, having each bad receiver transmit a NACK to the sender would result in a feedback implosion. Designing a protocol for reliable multicast that scales to a large number of receivers requires that a mechanism for feedback implosion avoidance be designed as well.

As stated in RFC-2887 [8], which discusses approaches for reliable multicast, “application requirements for reliable multicast are as broad and varied as the applications themselves.” This RFC also lists a set of requirements that affect the design of a reliable multicast protocol. With regard to feedback implosion avoidance, these requirements include whether the application needs to know that everyone received the data, whether the application needs to constrain differences between receivers, and whether the application needs to be totally reliable.

In a typical FEC-based reliable multicast, the sender creates from each data block $K+n$ packets. A receiver must receive any $K$ of these packets in order to decode the data block. In a hybrid FEC/ARQ-based scheme [1, 2, 7, 10, 15, 17], receivers that have not received enough packets correctly notify the sender, by sending a NACK message, and the sender transmits additional repair packets. The number of such repair rounds might be limited because of possible real-time considerations, because of buffer space limitation at the sender or the receivers, or because of the limited processing capability of the sending node, which might be involved in the transmission of hundreds of multicast data blocks at the same time.

Existing schemes for NACK suppression aim at minimizing the number of NACKs transmitted to the sender [1, 2, 14] while taking into account the differences in the delay from every receiver to the sender. In these schemes, a receiver suppresses the transmission of its NACK if “the accumulated state of NACKs heard from other receivers is equal to or supersedes the repair needs of the local receiver” [1]. In other words, if for a given data block the maximum number of repair packets required by any receiver is $M$, the only information the sender is guaranteed to receive is that at least one receiver needs $M$ repair packets. However, for some applications it would be important for the sender to get a good estimate of the number of receivers that need $M$ repair packets while still avoiding NACK implosion. Moreover, it could be useful for the sender to have an estimate for the exact loss distribution, namely, how many receivers are missing $i$ repair packets, for every $M \geq i \geq 1$. 
The most prominent applications of this type are hybrid FEC/ARQ-based schemes for reliable multicast [1, 7, 10, 17] in advanced cellular networks, like HSDPA, or broadband wireless access network, like IEEE-802.11 and IEEE-802.16. This is for the following reasons:

(R1) In such networks, continuous fading causes strong packet loss correlation over time. Since the number of repair rounds is usually limited, for the reasons given above, the sender needs to send in every round repair packets that will compensate not only for the loss in the previous round, but also for possible loss in the next round. In [17], such packets are referred to as “proactive repairs.” The best way for the sender to estimate the number of proactive repairs is to have a good estimate for the loss distribution in the previous round.

(R2) In such systems, continuous fading might also result in receivers whose PHY conditions are extremely bad. For reasons of cost effectiveness, a sender (base station) with a good estimate for the loss distribution may decide not to take such receivers into account while determining the number of repairs to be transmitted in the next round. The decision to discount such receivers is useful whether or not the sender intends to send proactive repairs.

(R3) In such systems, the concept of adaptive modulation and coding (AMC) plays a critical role in increasing system performance. With AMC, the base station usually uses higher order modulation (16- or 64-QAM) with higher code rate (R = 3/4 turbo code) when transmitting unicast packets to close receivers and lower order modulation (QPSK) and code rate when transmitting unicast packets to distant receivers. In contrast, multicast packets are usually transmitted using low order modulation and coding, because of the very high probability that at least one receiver is not close enough to the base station. However, this naive multicast approach can be enhanced if the sender is able to estimate the loss distribution of multicast packets. For example, the sender could transmit more repair packets using more efficient, high order, modulation, rather than fewer repair packets using less efficient modulation.

In light of the above discussion, and in contrast to previous works on the avoidance of NACK implosion, we propose in this paper a scheme where a limited number of NACKs transmitted by the group of bad receivers allows the sender to derive a good estimate of the size of this group. Such a scheme can be invoked in parallel for every group $G_i$ of receivers that are missing $i$ packets for the multicast data block. As far as we know, our paper is the first to explicitly show the correlation between the number of NACKs sent by a group of bad receivers and the sender’s ability to estimate the size of this group.

In the considered system, the sender (the base station) determines the maximum number of NACK$(i)$s it wants to receive for every number $i$ of missing packets, and the time period $T_i$ during which these NACKs should be transmitted. Each bad receiver uses a backoff timer, taken from a distribution $F$, to determine a time in the interval $[0, T_i]$ when the NACK$(i)$ is to be sent. If the base station receives enough NACK$(i)$s, it immediately sends a STOP$(i)$ message on the downlink, in order to suppress further NACK$(i)$ transmissions.

The most important part of this paper is the development of a statistical analysis algorithm, to be employed by the sender, for estimating the number of bad receivers for every group $G_i$. The estimation is based on the exact times at which the NACKs$(i)$ are received. This algorithm defines the likelihood function for the received NACKs$(i)$, and then uses the Newton-Raphson method to find the number of bad receivers for which this function is maximized. We also show that the distribution function $F$, used by the bad receivers for determining when to transmit their NACKs$(i)$, does not affect the estimation. This implies that a simple uniform distribution over any interval $[0, T]$ works as well as exponential or normal distributions, while reducing the calculation overhead at the sender and at the receivers. Another important contribution of this paper is to provide tight upper and lower bounds on the estimation error. Moreover, we show that when $1 \ll N \ll r$, where $r$ is the number of bad receivers and $N$ is the number of NACKs, this error is approximately $1/(N-1)$. For example, even if the number of bad receivers is very large, e.g., in the order of 10,000, 20-30 NACKs are sufficient to estimate it with an error of less than 5%.

The rest of this paper is organized as follows. In Section 2 we present related work. In Section 3 we present our estimation algorithm. In Section 4 we analyze the precision of this algorithm, and find tight upper and lower bounds for the error. The most important outcome of this analysis is that it enables to reduce the error introduced by our estimation algorithm, and to bring this error very close to 0. In Section 5 we analyze the effect of NACK loss on the estimation precision, and show that the algorithm performs well also in the presence of such losses. In Section 6 we present an application scenario of a hybrid FEC/ARQ-based reliable multicast protocol, where the sender uses our algorithm in order to determine how to respond to packet losses. Finally, in Section 7 we conclude the paper.
2. RELATED WORK

Not many papers have addressed the problem of estimating the number of bad receivers in a multicast network. In fact, almost all of the relevant works have addressed a slightly different problem: estimating the total number of receivers in a multicast group [4, 3, 14, 13]. While at first glance the two problems look very similar, there are still several significant differences, the most important of which is scalability. Whereas the number of bad receivers for FEC-related purposes should be estimated every transmitted data block, i.e., every 30-50 ms., the size of the multicast group should be estimated much less frequently. Moreover, when the size of the multicast group is estimated, there is a strong correlation between successive measurements. This correlation can be used to reduce the cost of the estimation process.

The authors of [4] discuss the M/M/∞ model for receivers entering and exiting the multicast group. To avoid feedback implosion, not all of the receivers send a message to the sender. Rather, each one sends a message with a predefined probability $p$. The sender uses $p$ and other parameters to estimate the number of receivers. The authors of [3] extend [4] by relaxing several assumptions and using different filters on the received feedbacks. The main differences between our scheme and the one proposed in [3, 4] are, first, that we are not interested in the dynamics of the group, and do not assume any correlation between two successive measurements. Second, in our approach, the base station sets a hard limit $N$ on the number of NACKs it is willing to receive.

The authors of [14] employ a timer-based approach similar to the one used in our scheme. However, they solve the opposite problem of using the size of the group of receivers to estimate the number of feedback messages sent. In their model, the receivers should stop sending feedback messages after the first one is transmitted. Nevertheless, because of the propagation delay, certain receivers might not be aware that a response message has been sent. The paper also estimates the latency incurred because of the timers and proposes a truncated-exponential timer distribution in order to reduce the number of feedbacks. The authors of [13] use the model proposed in [14], and analyze the feedback suppression both analytically and via simulations.

In [6], the authors propose a probabilistic polling model for estimating the size $n$ of a group. In this model, polling takes place over several rounds. In each round $i$, the sender multicasts a polling request. A receiver sends a response message with probability $p_i$. After $k$ rounds, the sender estimates the value of $n$ from the polling probabilities $p_1, p_2, \ldots, p_k$ and from the number of responses $r_1, r_2, \ldots, r_k$ in every round. This paper also shows how to map the models of [5] and [14] to its binomial estimation model.

As in our scheme, the authors of [11] also employ the concept of maximum likelihood, but they do so for the sake of estimating the size of a multicast group and not the number of bad receivers. They consider a sender broadcasting an RFB (Request for Feedback) message to a group of receivers. Each receiver sets a random timer using a known probability distribution function and sends a feedback message when it expires. When the first feedback message arrives at the sender, it broadcasts the next RFB, which also stops the receivers from sending additional responses to the previous RFB. The number of feedback messages received by the sender is therefore proportional to the length of the RTT. The fact that our paper and [11] address different problems is translated into the following differences:

- We overcome feedback implosion by limiting the number of response messages sent by receivers, while [11] limits the time during which these messages are sent.
- Our method is more appropriate for networks with a small RTT (e.g., IEEE 802.16), while [11] better suits networks with a large RTT. In particular, running the algorithm of [11] in a network with a 0 RTT will result in a single NACK being received by the sender.
- In our scheme, each measurement of the size of the bad receiver group is independent of previous measurements, while in [11] the results of previous measurements are taken into account.

RFC 3941 [1] discusses the creation of NACK-oriented reliable multicast (NORM) protocols. It gives a set of “building blocks” that may be used by such protocols for addressing various issues: scalability, reliability, etc. In order to avoid NACK implosion, it proposes that timer-based suppression techniques be used.

In [9], LaMaire and Krishna consider a slotted Aloha shared channel, where contending hosts try to occupy a shared slot with probability $p$. To maximize the throughput, $p$ has to be equal to $1/n$, where $n$ is the number of contending hosts. The authors propose a scheme for estimating $n$. This scheme is derived from the probability of first success given a success $p_f/p_s$. To this end, every time a packet is transmitted, each contending host indicates whether this is the first transmission attempt for this packet or whether it has already experienced a collision. While this paper solves a different problem than ours, it seems at a first glance that the same mechanism can also be used to estimate the number of bad receivers – for example, by having them contend for transmitting their NACKs. However, this is not the case mainly because, in order to be stable, this scheme must allow most of the receivers to transmit their packets, whereas
in our scheme we want only a very small fraction of the 
bad receivers to transmit NACKs. Moreover, this 
scheme does not bound the time period during which 
response messages are sent by the receivers.

3. ESTIMATING THE NUMBER OF BAD RECIPIENTS

As indicated in Section 1, when our scheme is used 
in order to get an estimate for the packet loss distribution, 
it is invoked by the sender (base station) several 
times in parallel, once for each packet received. 
To simplify the notations in the following section, 
we consider one instance of the scheme, and omit any 
reference to i.

Consider a downstream wireless channel, where 
the base station transmits a packet to a group of wireless 
hosts, each of which can be static or mobile. For various 
reasons, some of these receivers might not receive the 
message. We denote the number of such bad receivers 
by r. Our goal is to estimate the value of r while limiting 
the number of NACKs sent by bad receivers. When 
a bad receiver discovers that it has not received 
the message correctly, it invokes the following algorithm:

Algorithm 1. the algorithm invoked by a bad receiver

- Choose a random timer in the range [0, T] using a known probability distribution function F.

- Let n indicate the number of NACKs sent by other bad receivers before the timer expires. If n < N, send a NACK to the base station. Otherwise, do not send.

Following this algorithm, the base station receives 
during [0, T] no more than N NACKs. If this number 
is smaller than N, the base station knows exactly 
how many bad receivers there are. Therefore we will 
focus on the case where the base station receives exactly 
N NACKs.

Let f be the probability density function of F. Let 
X1, . . . , XN be random variables denoting the transmission (and reception) times of the N NACKs. 
Without loss of generality, these random variables are assumed to be ordered in non-decreasing order such that 
X1 ≤ X2 ≤ . . . ≤ XN. Finally, let x1, . . . , xN denote 
the exact values of X1, . . . , XN in a specific experiment.

We use the maximum likelihood method to estimate 
r. Let R be the random variable denoting the number of 
bad receivers and denote by fX1,X2,...,XN|R=r(x1,x2,...,xN) the 
joint density function of X1, X2, . . . , XN given that 
the number of bad receivers is r. This function is the probability density of the first N order statistics of 
distribution F, for which it is known that [16]:

\[ f_{X_1,X_2,...,X_N}(x_1,x_2,...,x_r) = \]
\[ = P(X_1(x_1,x_1 + dx_1),...,X_r(x_r,x_r + dx_r)) = \]
\[ = r!P(Y_1(x_1,x_1 + dx_1),...,Y_r(x_r,x_r + dx_r)) = \]
\[ = r!(F(x_1 + dx_r) - F(x_1)) \cdots (F(x_r + dx_r) - F(x_r)) = \]
\[ = r!f(x_1) \cdots f(x_r) dx_1 \cdots dx_r, \]

where Y1, . . . , Yr are independent random variables from distribution F. Therefore,

\[ f_{X_1,X_2,...,X_N}(x_1,x_2,...,x_r) = r!f(x_1) \cdots f(x_r). \]

In order to find the joint density of the first N X’s, we 
integrate over xN+1, . . . , xN:

\[ f_{X_1,X_2,...,X_N|R=r}(x_1,x_2,...,x_N) = \]
\[ = \int_{x_N < x_i < x_{i+1} < \cdots < x_r} f_{X_1,X_2,...,X_N}(x_1,x_2,...,x_r) dx_{N+1} ... dx_r = \]
\[ = \int_{x_N < x_i < x_{i+1} < \cdots < x_r} r!f(x_1) \cdots f(x_r) dx_{N+1} ... dx_r = \]
\[ = r! \int_{x_N}^T dx_{N+1} \cdots \int_{x_{i-1}}^T dx_i f(x_1) \cdots f(x_r) = \]
\[ = r! \int_{x_N}^T dx_{N+1} \cdots \int_{x_{i-1}}^T dx_i \prod_{i=1}^{r-1} f(x_i) \cdot (1 - F(x_{i-1})) = \]
\[ = r! \int_{x_N}^T dx_{N+1} \cdots \int_{x_{i-1}}^T dx_i \prod_{i=1}^{r-2} f(x_i) \cdot (1 - F(x_{i-2}))^2 = \]
\[ = \cdots = \frac{r!}{(r-N)!} \prod_{i=1}^N f(x_i) \cdot (1 - F(x_N))^{r-N}. \]  

(1)

Define the likelihood function L(r) to be

\[ L(r) = f_{X_1,X_2,...,X_N|R=r}(x_1,x_2,...,x_N). \]

We now seek for the value of r that maximizes L(r). 
Such an r yields the maximum likelihood for getting 
the considered experiment’s outcome, x1, . . . , xN, and 
is therefore the most probable number of bad receivers. 
We find the maximum of L(r) by differentiation. Since 
L(r) is a product of other functions, it is hard to differen-
tiate it directly. Since ln is a monotonically increasing 
function, L(r) gets its maximum for the same value of 
r as l(r), where

\[ l(r) = \ln L(r) = \]
\[ = \ln \frac{r!}{(r-N)!} + \ln f(x_1) + \cdots + \ln f(x_N) + \]
\[ + (r-N) \ln(1 - F(x_N)) = \]
\[ = \ln(r - N + 1) + \cdots + \ln r + \]
\[ + r \ln(1 - F(x_N)) + \text{const.} \]

(2)

In this equation, const is a constant with respect to r.
We now differentiate \( l(r) \) with respect to \( r \) and get
\[
l'(r) = \frac{1}{r^2} + \frac{1}{r-1} + \ldots + \frac{1}{r-N+1} + \ln(1-F(x_N)).
\]

Thus, in order to find the value of \( r \) which maximizes the likelihood function \( L(r) \), we need to find real values of \( r \) that satisfy the following equation:
\[
\frac{1}{r} + \frac{1}{r-1} + \ldots + \frac{1}{r-N+1} + \ln(1-F(x_N)) = 0. \tag{3}
\]

**Proposition 1.** From the \( N \) possible real solutions of Eq. 3, the one that maximizes \( L(r) \) is the maximum one.

**Proof:** \( L(r) \) and \( l(r) \) get their maximum at the same \( r \). Thus it is enough to show that \( l(r) \) gets its maximum at the maximum solution of Eq. 3.

Since for every \( r \)
\[
l''(r) = -\frac{1}{r^2} - \frac{1}{(r-1)^2} - \ldots - \frac{1}{(r-N+1)^2} < 0, \tag{4}
\]
then any real root of Eq. 3 is a local maximum of \( l(r) \). The global maximum is one of the local maxima, so it remains to find which of the local maxima gives the highest value of \( l \).

Substituting
\[
\ln(1-F(x_N)) = -\left(\frac{1}{r} + \frac{1}{r-1} + \ldots + \frac{1}{r-N+1}\right)
\]
from Eq. 3 into Eq. 2 yields:
\[
l(r^*) = \ln r^* + \ldots + \ln(r^* - N + 1) -
- r^* \left(\frac{1}{r^*} + \frac{1}{r^* - 1} + \ldots + \frac{1}{r^* - N + 1}\right) + \text{const} =
= \ln r^* + \ldots + \ln(r^* - N + 1) -
- 1 - \left(1 + \frac{1}{r^* - 1}\right) - \ldots -
= \left(1 + \frac{N - 1}{r^* - N + 1}\right) + \text{const}.
\]

This is a monotonically increasing function. Therefore, of all the roots \( r^* \) of Eq. 3, the one whose value is maximum will maximize both \( l(r) \) and \( L(r) \).

To conclude, the algorithm executed by the base station for estimating the number of bad receivers \( r \) for every packet is as follows.

**Algorithm 2. (base station algorithm)**

Find the greatest real root of Eq. 3.

A practical method for solving Eq. 3 is as follows. Since the \( \ln \) term is constant, the equation has the form
\[
\frac{1}{r^2} + \frac{1}{r-1} + \ldots + \frac{1}{r-N+1} + c = 0.
\]
This function has vertical asymptotes at points \( r = 0, 1, \ldots, N - 1 \). From Eq. 4 it follows that the function decreases monotonically at every interval \((i-1,i), i = 1, \ldots, N - 1\) and thus it has \( N - 1 \) roots at the interval \((0,N-1)\). The function also decreases monotonically at the interval \((N-1,\infty)\), and thus has its greatest root in this interval. This is the root we are seeking. To find it, the sender can employ the Newton-Raphson method. Given an equation \( h(x) = 0 \) where \( h \) is a continuously differentiable function and given a starting point \( x_0 \), near which the equation root is located, the method iteratively finds an approximation for the root with any desirable precision. On the \((n+1)\)-th iteration, \( x_{n+1} = x_n - \frac{h(x_n)}{h'(x_n)} \), where \( h'(x) \) is the derivative of \( h(x) \). The idea is to find the tangent of \( h \) at \( x_n \) and to set \( x_{n+1} \) to the point where the tangent crosses the x-axis, thereby getting closer to the root. In our case, \( h \) is given in Eq. 3 and \( x_0 = N - 1 + \epsilon \), where \( \epsilon \) is small positive number. This starting point was chosen for two reasons. First, there must not be asymptotes between the starting point and the root (therefore \( x_0 > N - 1 \)). Second, there must not be asymptotes between any \( x_n \) and the root, and since the function is monotonic at the interval \((N-1,\infty)\), it is implied that \( x_0 < \text{root} \). The whole process stops when \( |h(x_n)| \) gets sufficiently close to 0. In the simulation results, shown later, we stopped when \( |h(x_n)| < 0.001 \), which usually holds after 9-10 iterations. The value \( x_n \) for the last \((n)\) iteration is taken to be the solution of Eq. 3, namely, the estimated value of \( r \).

**Theorem 1.** The distribution function \( F \) does not affect the estimated value of \( r \) as computed in Eq. 3.

**Proof:** According to Eq. 3, the only way the value of \( r \) might depend on \( F \) is through \( -\ln(1-F(x_N)) \). However, we will show now that for every \( i \), the value of \( -\ln(1-F(x_i)) \) does not depend on \( F \), namely, that the distribution of the random variable \( Y = -\ln(1-F(x_i)) \) does not depend on \( F \) given \( R = r \).

Denote by \( f_{X_i|R=r}(x) \), for \( i = 1, \ldots, N \), the density function of \( X_i \) given that the number of bad receivers is \( r \). The function \( f_{X_i|R=r}(x) \) is actually the probability density of the \( i \)-th order statistic of distribution \( F \), namely:
\[
f_{X_i|R=r}(x) = \frac{d}{dx} F_{X_i|R=r}(x) = \frac{d}{dx} P(X_i \leq x|R = r) =
= \frac{d}{dx} P(\text{at least } i \text{ of the } r \text{ timers expire before } x) =
= \frac{d}{dx} \sum_{j=1}^{r} \binom{r}{j} F(x)^j (1 - F(x))^{r-j} = \ldots =
= r \binom{r-1}{i-1} F(x)^{i-1} (1 - F(x))^{r-i} f(x).
\]

Then, we have
\[
F_{X_i|R=r}(t) = \int_{0}^{t} r \binom{r-1}{i-1} F(x)^{i-1} (1 - F(x))^{r-i} f(x) \, dx.
\]
Substituting $y = F(x)$, so that $dy = F'(x)dx = f(x)dx$, yields
\[
F_{X_t|R=r}(t) = r \binom{r - 1}{i - 1} \int_0^{F(t)} y^{i-1}(1-y)^{r-i} dy =
\]
\[= r \binom{r - 1}{i - 1} \int_0^{F(t)} y^{i-1} \sum_{k=0}^{r-i} (-1)^k \binom{r-i}{k} y^k dy =
\]
\[= r \binom{r - 1}{i - 1} \sum_{k=0}^{r-i} (-1)^k \binom{r-i}{k} \int_0^{F(t)} y^{k+i-1} dy =
\]
\[= r \binom{r - 1}{i - 1} \sum_{k=0}^{r-i} (-1)^k \binom{r-i}{k} \frac{F(t)^{k+i}}{k+i}.
\]
Let $F_{Y|R=r}(z)$ be the distribution function of $Y$ given $R = r$. Hence,
\[F_{Y|R=r}(z) = P(\ln(1 - F(X_t))) \leq z|R=r) =
\]
\[= P(1 - F(X_t) \geq e^{-z}|R=r) =
\]
\[= P(F(X_t) \leq 1 - e^{-z}|R=r) =
\]
\[= P(X_t \leq F^{-1}(1 - e^{-z})|R=r) =
\]
\[= F_{X|X=r}(F^{-1}(1 - e^{-z})) =
\]
\[= r \binom{r - 1}{i - 1} \sum_{k=0}^{r-i} (-1)^k \binom{r-i}{k} (1- e^{-z})^{k+i}.
\]
Thus, given $R = r$, $Y$ does not depend on $F$.

In [14] it was shown that a truncated exponential distribution outperforms a uniform distribution at reducing feedback implosion when the time interval during which NACKs are to be sent is limited. Theorem 1 suggests that this is not the case when setting a hard limit on the number of NACKs. The time interval $[0, T]$ during which the NACKs are transmitted is arbitrary too, since changing $T$ results in changing the support of the distribution.

**Implementation Note:**

1. A problem may arise when implementing Algorithm 1 because NACK messages, usually transmitted on a contention channel, are subject to loss due to collisions or to transmission errors. This problem can be addressed by having the base station send a confirmation for every NACK it receives. A host that does not receive this confirmation will resend the NACK and specify the offset between its current local time and the time of the original NACK. While this approach minimizes the probability for a NACK loss, it does not prevent it altogether, because a receiver might not be able to transmit or retransmit its NACK before the base station “closes the gate” for more NACKs. However, in Section 5 we prove that our scheme can tolerate the loss of several NACKs.

2. Another possible problem is that the transmission of a NACK might be delayed due to possible congestion on the shared uplink channel, or that different hosts might have different delay to the base station. These problems can also be overcome by having the host write inside the NACK the intended transmission time.

We implemented the algorithm proposed in Section 3 for two distribution functions: the uniform distribution $f(x) = \frac{1}{T}$ and the truncated exponential distribution $f(x) = \frac{1}{e^x}, x \in [0, T]$. In the simulation, the receivers draw timers from the above distributions and send a NACK to the sender if fewer than $N$ NACKs have already been sent.

We have simulated 100, 1000 and 10,000 bad receivers. Figure 1 depicts the average error in estimating $r$, namely $|r_{real} - r_{estimated}|/r_{real}$, as a function of $N$ for the uniform distribution and for the truncated exponential distribution. Each point in the graph is the average of 1000 different runs. This figure reveals that there is no noticeable difference between the uniform distribution and the truncated exponential distribution, as predicted by Theorem 1. It is evident that the greater $N$ is, the better the estimation is. This means that the base station may choose in advance which estimation error it can tolerate and then wait for the corresponding number of NACKs. The graphs also show that a modest number of NACKs (e.g., 30) yields an error of only 5% even for a large group of bad receivers, proving that the proposed scheme scales very well.

Another important property of the proposed scheme is that it actually overestimates the number of bad receivers, i.e., with high probability $r_{estimated} \geq r_{real}$. This is shown in Figure 2, which depicts the estimation error as $(r_{estimated} - r_{real})/r_{real}$, rather than as $|r_{estimated} - r_{real}|/r_{real}$. We can see that for all values of $N$ and for any number of bad receivers, the error is always positive. Moreover, by comparing the graphs in Figure 2 to those in Figure 1, we can see that the curves are almost identical. This implies that $(r_{estimated} - r_{real})/r_{real} \approx |(r_{estimated} - r_{real})|/r_{real}$, or, in other words, that the difference between the actual and the estimated values results from the overestimation.

We prove this property mathematically in Theorem 2 of Section 4, and show how it can be used in order to bring the estimation error very close to 0.

**4. PRECISION ANALYSIS AND OPTIMIZATION**

In this section we analyze the precision of our algorithm. The importance of this section is two-fold. First, we prove that the estimation error is approximately $\frac{1}{N+r}$. Second, we use this analysis in order to
further reduce the error, and to bring it very close to 0.

We have already shown that the timer distribution function $F$ does not affect the result $r$ of the estimation. Therefore, in the following analysis we consider a uniform timer distribution on the interval $[0,1]$. On that interval, $f(x) = 1$ and $F(x) = x$.

The base station estimates the number of bad receivers by finding the maximal value of $r$ that solves the following equation:

$$\frac{1}{r} + \frac{1}{r-1} + \ldots + \frac{1}{r-(N-1)} = -\ln(1-xN). \quad (5)$$

Let this solution be $r = g(xN)$. Let $R$ be a random variable denoting the estimated number of bad receivers, and let its expected value be $E(R)$. Our goal is to approximate the relative error $\frac{E(R)-r}{r}$.

We have seen that

$$f_{X|R=r}(x) = r\frac{r-1}{i-1}F(x)^{i-1}(1-F(x))^{r-i}f(x). \quad (6)$$

Since $f(x) = 1$ and $F(x) = x$, then substituting $i = N$ into Eq. 6, we get

$$f_{X|R=r}(x) = r\left(\frac{r-1}{N-1}\right)x^{N-1}(1-x)^{r-N}. \quad (6)$$

Therefore,

$$E(R) = \int_0^1 g(x)f_{X|R=r}(x)\,dx. \quad (7)$$

We used an iterative method in order to solve $g(x)$. In what follows we seek for upper and lower bounds on this function. From Eq. 5 follows that $-\ln(1-xN) \geq \frac{N}{r}$ and $-\ln(1-xN) \leq \frac{N}{r-(N-1)}$. Therefore,

$$-\frac{N}{\ln(1-xN)} \leq g(xN) \leq -\frac{N}{\ln(1-xN)} + (N-1). \quad (8)$$

To find a lower bound we substitute the left-hand
part of Eq. 8 into Eq. 7 and get
\[ E(R) \geq -rN \left( \frac{r-1}{N-1} \right) \int_0^1 \frac{x^{N-1}(1-x)^{r-N}}{\ln(1-x)} \, dx = \]
\[ = -rN \left( \frac{r-1}{N-1} \right) \int_0^1 \frac{x^{r-N}(1-x)^{N-1}}{\ln x} \, dx. \quad (9) \]
Next, we note that
\[ \int_0^1 \frac{x^n(1-x)^k}{\ln x} \, dx = \sum_{i=0}^k (-1)^i \binom{k}{i} \ln(n+i+1), \quad (10) \]
for \( n \geq 0, k \geq 1 \), and \( \int_0^1 \frac{x^n}{\ln x} \, dx = -\infty \) for \( n \geq 0, k = 0 \).
This means that for \( N = 1 \), \( E(R) = \infty \) and the relative estimation error is also infinite. Thus, from now on we assume that \( N \geq 2 \). In the following equations, a sum or a product whose lower limit is greater than its upper limit is considered to be equal to 0 or to 1 respectively. Substituting Eq. 10 into Eq. 9, yields:
\[ E(R) \geq -rN \left( \frac{r-1}{N-1} \right) \cdot \sum_{i=0}^{N-1} (-1)^i \binom{N-1}{i} \ln(r - (N-1) + i) = \]
\[ = -rN \left( \frac{r-1}{N-1} \right) \prod_{i=0}^{N-2} (r - (N-1) + i). \quad (12) \]
We can expand the right item of this product as follows:
\[ \sum_{i=0}^{N-1} (-1)^i \binom{N-1}{i} \ln(r - (N-1) + i) = \]
\[ = \ln(r - (N-1)) + \sum_{i=1}^{N-2} (-1)^i \left[ \binom{N-2}{i-1} + \binom{N-2}{i} \right] \cdot \ln(r - (N-1) + i) + (-1)^{N-1} \ln r = \]
\[ = \ln(r - (N-1)) + \sum_{i=0}^{N-3} (-1)^{i+1} \binom{N-2}{i} \ln(r - (N-1) + i+1) + \sum_{i=1}^{N-2} (-1)^i \binom{N-2}{i} \ln(r - (N-1) + i) + \]
\[ +(1)^{N-1} \ln r. \quad (13) \]
We now group in Eq. 13 those \( \ln \) terms that are adjacent but differ in sign, and we get
\[ \ln(r - (N-1)) - \]
\[ - \sum_{i=0}^{N-3} (-1)^i \binom{N-2}{i} \ln(r - (N-1) + i) + \]
\[ + \sum_{i=1}^{N-2} (-1)^i \binom{N-2}{i} \ln(r - (N-1) + i) + \]
\[ +(1)^{N-1} \ln r = \]
\[ = \ln \left( 1 + \frac{1}{r - (N-1)} \right) \]
\[ - \sum_{i=0}^{N-3} (-1)^i \binom{N-2}{i} \ln \left( 1 + \frac{1}{r - (N-1) + i} \right) - \]
\[ -(1)^{N-2} \ln \left( 1 + \frac{1}{r - (N-1) + i} \right). \]
Substituting this into Eq. 12 yields
\[ E(R) \geq \frac{rN}{(N-1)^2} \prod_{i=0}^{N-2} (r - (N-1) + i). \quad (14) \]
\[ \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \ln \left( 1 + \frac{1}{r - (N-1) + i} \right) = \]
\[ = \frac{rN}{(N-1)^2} \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \cdot \prod_{j=0}^{N-2} (r - (N-1) + j) \cdot \ln \left( 1 + \frac{1}{r - (N-1) + i} \right). \]
The sequence
\[ \left\{ \left( 1 + \frac{1}{r - (N-1) + i} \right)^{r-(N-1)+i} \right\}_{i=0}^{N-2} \]
is monotonically increasing and upper bounded by \( e \). Denote
\[ p = \ln \left( 1 + \frac{1}{r - (N-1)} \right)^{r-(N-1)}. \]
Then, $0 < p < 1$, and for large enough values of $r - (N - 1)$, $p$ is close to 1. We now get:

$$E(R) \geq \frac{p \cdot r \cdot N}{(N-1)^2}.$$  (15)

$$\sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \prod_{j \neq i} (r - (N - 1) + j).$$

We continue with a series of propositions that will help us to simplify the above expression.

**Proposition 2.** Let $c_k$ be the coefficient of $r^k$ in the polynomial $\prod_{j \neq i} (r - (N - 1) + j)$, for $1 \leq k \leq N - 2$. Then, there exists a polynomial $p(x)$ of degree $N - 2 - k$ such that $c_k = p(i)$.

**Proof:** For $1 \leq k \leq N - 2$,

$$c_k = \sum_{A_1} \text{product of the factors} -[(N - 1) - j] = \sum_{A_2} \text{product of the factors} -[(N - 1) - i] + \sum_{A_3} \text{product of the factors} -[(N - 1) - j] = a_k + \prod_{i \neq j} (N - 1) - i.\quad \text{where:}

1. $A_1$ is the set of all ways to choose $N - 2 - k$ from $\{0, \ldots, N - 2\} \setminus \{i\}$.
2. $A_2$ is the set of all ways to choose $N - 2 - k$ from $\{0, \ldots, N - 2\}$.
3. $A_3$ is the set of all ways to choose $N - 3 - k$ from $\{0, \ldots, N - 2\} \setminus \{i\}$.

$a_k$ is a constant with respect to $i$.

For $k = N - 2$, $c_{N-2} = 1$. Therefore, the proposition holds for $c_{N-2}$ with the constant polynomial $p(x) = 1$.

By reverse induction, suppose that the proposition holds for $k + 1$. Then, there is a polynomial $q(x)$ of degree $N - 3 - k$ such that $c_{k+1} = q(i)$. Define $p(x) = a_k + (N - 1)q(x) - xq(x)$. Then, $p(x)$ is a polynomial of degree $N - 2 - k$, and $p(i) = a_k + (N - 1)q(i) - iq(i) = a_k + \prod_{i \neq j} (N - 1) - i c_{k+1} = c_k$.

**Proposition 3.** For all $n \geq 0$, $\sum_{i=0}^{n} (-1)^i i! = 0$.

**Proof:** The first claim follows immediately from the binomial formula:

$$\sum_{i=0}^{n} \binom{n}{i} (-1)^i = \sum_{i=0}^{n} \binom{n}{i} (-1)^i 1^{n-i} = (1 - 1)^n = 0.$$

The second claim is proved by differentiating $(x - 1)^n$ $k$ times, first as a composite function and then after expansion using the binomial formula.

$$[(x - 1)^n]'(k) = n(n - 1)\ldots(n - (k - 1))(x - 1)^{n-k}$$

$$[(x - 1)^n]'(k) = \left[ \sum_{i=0}^{n} \binom{n}{i} (-1)^{n-i} x^i \right]'(k)$$

$$= (-1)^n \sum_{i=k}^{n} \binom{n}{i} (-1)^i \cdot i(i - 1)\ldots(i - (k - 1)) x^{i-k}.$$  

By substituting $x = 1$, the proof is completed.

**Proposition 4.** Every polynomial $p(x) = \sum_{i=0}^{k} a_i x^i$ of degree $k$ can be written as a linear combination of the polynomials in the set $B_k = \{1, x, x(x-1), x(x-1)(x-2), \ldots, x(x-1)\ldots(x-(k-1))\}$.

**Proof:** By induction on $k$. For $k = 0$, $p(x) = a_0$ is certainly a linear combination of the polynomials in $B_0 = \{1\}$. For a general $k \geq 1$,

$$p(x) = a_k \cdot x(x-1)\ldots(x-(k-1)) + [p(x) - a_k \cdot x(x-1)\ldots(x-(k-1))],$$

where the polynomial in brackets is of degree $k - 1$. Thus, by the induction hypothesis, it can be written as a linear combination of the polynomials in $B_{k-1}$.

**Proposition 5.** For every $k \geq 0$ and $n \geq k + 1$, $\sum_{i=0}^{n} \binom{n}{i} (-1)^i = 0$, where $p_k(x)$ is a polynomial of degree $k$.

**Proof:** By Proposition 4 we can write

$$p_k(i) = b + a_0 i + a_1 i(i-1) + \ldots + a_{k-1} i(i-1)\ldots(i-(k-1)).$$

Then, by Proposition 3,

$$\sum_{i=0}^{n} \binom{n}{i} (-1)^i p_k(i) =$$

$$= b \sum_{i=0}^{n} \binom{n}{i} (-1)^i + a_0 \sum_{i=1}^{n} \binom{n}{i} (-1)^i + \ldots + a_{k-1} \sum_{i=k}^{n} \binom{n}{i} (-1)^i i(i-1)\ldots(i-(k-1)) =$$

$$= 0.$$  

We will now use the above propositions to simplify the expression for $E(R)$ from Eq. 15.
second term can be expanded as follows:

\[ \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \prod_{j=0; j \neq i}^{N-2} (r - (N - 1) + j) = \]

\[ = \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \sum_{k=0}^{N-2} c_k r^k = \]

\[ = \sum_{k=1}^{N-2} r^k \left[ \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} \right] + \]

\[ + r^0 \sum_{i=0}^{N-2} \binom{N-2}{i} (-1)^i c_0. \]

By Proposition 2, \( c_k \) can be written as a polynomial of degree \( N - 2 - k \) for the variable \( i \). Thus, by Proposition 5, the first term vanishes. Note that \( c_0 = (-1)^{N-2} (N-1) \frac{(N-2)!}{(N-1) - 1} \) and so the second term can be expanded as follows:

\[ r^0 \sum_{i=0}^{N-2} \binom{N-2}{i} (-1)^i c_0 = \]

\[ = (-1)^N \sum_{i=0}^{N-2} \binom{N-2}{i} (-1)^i \frac{(N-1)!}{N - 1 - i} = \]

\[ = (-1)^N \frac{(N-1)!}{N - 1} \sum_{i=0}^{N-2} \binom{N-2}{i} (-1)^{N-1} \frac{1}{i + 1} = \]

\[ = (N-1)! \frac{1}{N - 1} \sum_{i=0}^{N-2} (-1)^i \frac{(N-1)!}{i + 1 ! (N-2 - i)!} = \]

\[ = (N-1)! \frac{1}{N - 1} \sum_{i=0}^{N-2} (-1)^i \frac{(N-1)!}{(i+1)! (N-2 - i)!} = \]

\[ = (N-2)! \sum_{i=1}^{N-1} (-1)^i \frac{(N-1)!}{i + 1} = \]

\[ = (N-2)! \sum_{i=1}^{N-1} (-1)^i \frac{(N-1)!}{i + 1} = \]

\[ = -(N-2)! \sum_{i=1}^{N-1} (-1)^i \frac{(N-1)!}{i + 1} = \]

\[ = -(N-2)! \sum_{i=1}^{N-1} (-1)^i \frac{(N-1)!}{i + 1} = \]

\[ = -(N-2)! \left[ \sum_{i=0}^{N-2} (-1)^i \binom{N-1}{i} - 1 \right] = \]

\[ = (N-2)!, \]

where the last equality follows from Proposition 3.

Therefore, from Eq. 15, it follows that

\[ E(R) \geq \frac{p \cdot r \cdot N}{(N-1)!} (N-2)! = p \cdot r \frac{N}{N-1} = \]

\[ = p \cdot r \left( 1 + \frac{1}{N-1} \right). \]  \hspace{1cm} (16)

This completes the lower bound analysis for \( E(R) \). To find an upper bound for \( E(R) \), we employ similar techniques. In the following equations, a number above a relation symbol denotes the number of an equivalent equation for the lower bound:

\[ E(R) = \int_0^1 g(x) f_{X|\cap=r^{-1}}(x) dx \leq \]

\[ \leq -rN \int_0^1 \frac{x^{N-1}(1-x)^{r-N}}{\ln(1-x)} dx + \]

\[ + (N-1) \int_0^1 f_{X|\cap=r^{-1}}(x) dx = \]

\[ = -rN \int_0^1 \frac{x^{r-N}(1-x)^{N-1}}{\ln x} dx + \]

\[ + (N-1) \int_0^1 f_{X|\cap=r^{-1}}(x) dx = \]

\[ = -rN \int_0^1 \frac{x^{r-N}(1-x)^{N-1}}{\ln x} dx + \]

\[ + (N-1) \int_0^1 f_{X|\cap=r^{-1}}(x) dx = \]

\[ = -rN \int_0^1 \frac{x^{r-N}(1-x)^{N-1}}{\ln x} dx + \]

\[ + (N-1) \int_0^1 f_{X|\cap=r^{-1}}(x) dx = \]

Denote \( q = \ln \left( 1 + \frac{1}{x} \right) \). Then, \( 0 < q < 1 \), and for large enough values of \( r - 1 \), \( q \) is close to 1. Then,

\[ E(R) \leq \frac{q \cdot r \cdot N}{(N-1)!} \sum_{i=0}^{N-2} (-1)^i \binom{N-2}{i} = \]

\[ \cdot \left[ \prod_{j=0; j \neq i}^{N-2} (r - (N - 1) + j) \right] + (N-1) \] \hspace{1cm} (16)

\[ = q \cdot r \frac{N}{N-1} + (N-1) = \]

\[ = q \cdot r \left( 1 + \frac{1}{N-1} \right) + (N-1). \]  \hspace{1cm} (17)

This completes the upper bound analysis. From Eq.
Using the above analysis, we now show how to reduce the estimation error with no further cost. Let \( r_{\text{real}} \) be the real number of bad receivers and \( r_{\text{estim}} \) be the estimated number using our estimation algorithm. From the above analysis, we know that for the case when NACKs do not get lost,

\[
r_{\text{real}} \approx \frac{r_{\text{estim}}}{1 + \frac{1}{N-1}}.
\]  

(19)

Figure 4 shows the result of this process for 1,000 and 10,000 bad receivers. The upper curve (“Before”) is drawn without applying “error cancellation”, whereas the lower curve (“After”) is drawn with it. It is evident that this process significantly reduces the error and is especially efficient when the number of NACKs is small.

5. THE EFFECT OF NACK LOSS

In the analysis so far we have assumed that NACKs are not lost. We have also presented (see “Implementation Notes” in Section 3) a simple handshake-based protocol, to be executed between each NACK sender and the base station, thus guaranteeing the retransmission of lost NACKs by their senders. However, this mechanism does not entirely prevent NACK loss. For example, a NACK sender may not be able to get transmission rights on the shared upstream channel before the deadline for sending NACKs is reached. Such a deadline is necessary, of course, even if the base station is willing to wait a while for the retransmissions to be received after the formal window of \([0, T]\). In this section we study the effect of NACK losses on our estimation scheme.

Suppose that by the time the base station receives the \( N \)th NACK, \( S \) NACKs have been lost. Therefore, the NACK considered by the base station to be the \( N \)th is actually the \( N + S \)th. This influences Eq. 5, which the base station solves to find \( r \). This equation now should read:

\[
\frac{1}{r} + \frac{1}{r-1} + \ldots + \frac{1}{r-(N-1)} = - \ln(1-x_{N+S}).
\]  

(20)

Going through the analysis in Section 4 and replacing occurrences of \( N \) with \( N + S \) when \( N \) indicates the index of the \( N \)th NACK (as opposed to places where it indicates the number of NACKs), we get:

\[
-p \frac{S-1}{N+S-1} - (1-p) \leq \frac{E(R) - r}{r} \leq -q \frac{S-1}{N+S-1} - (1-q) + \frac{N-1}{r}.
\]  

(21)

In this equation, \( p = \ln \left( 1 + \frac{1}{r-(N+S-1)} \right)^{r-(N+S-1)} \), and \( q = \ln \left( 1 + \frac{1}{r} \right)^{r-1} \), and see excellent agreement between the two curves.

Using the above analysis, we now show how to reduce the estimation error with no further cost. Let \( r_{\text{real}} \) be the real number of bad receivers and \( r_{\text{estim}} \) be the estimated number using our estimation algorithm. From the above analysis, we know that for the case when
bad receivers, respectively. In each figure we consider 1 or 5 NACK losses, and show the error using both simulation and Eq. 22.

A few conclusions can be drawn from the graphs. First, comparing Figure 5(a) to Figure 5(b) reveals no major difference. This implies that, as before, the error is independent of $r$. Second, as expected (see Eq. 22), the error now is negative. Thus, in the presence of NACK loss, our method underestimates the number of bad receivers, instead of overestimating it. Finally, we can see also here very good agreement between the simulation and the analysis.

An interesting observation is that when $S = 1$, the error almost vanishes. This is also implied by Eq. 22, which predicts a 0 error in this situation. This fact has a rather simple explanation. As noted above, NACK losses cause our method to underestimate the error, while with no NACK losses the error is overestimated. When $S = 1$, the two counter-forces cancel each other, resulting in an error very close to 0.

We can apply the same process used in the end of Section 4 to reduce the effect of NACK loss on the precision of our algorithm. In this case, Eq. 19 is replaced by the following equation:

$$ r_{real} \approx r_{estim} \frac{2}{N+S-1} $$  

Figure 6 presents the error for $S = 10$ lost NACKs before and after applying the error cancellation process. Note, however, that in order to implement this process when $S > 0$, the base station needs to know the value of $S$, which is not always possible.

6. AN APPLICATION SCENARIO

In this section we shows how the proposed scheme for estimating the number of bad receivers can be used in the context of a hybrid FEC/ARQ reliable multicast protocol. In particular, we show how the number of NACKs used by the sender for estimating the number of bad receivers affects the performance of the protocol.
Figure 6: Estimation error vs. $N$ for $S = 10$, before and after “error cancellation”

FEC (Forward Error Correction) codes are known to be an efficient method for increasing the reliability of a lossy channel, especially for multicast applications. We consider two special classes of FEC codes: block FEC codes and expandable FEC codes [12]. Block FEC codes create from each data block $n$ symbols. A receiver must receive at least $K$ of these symbols in order to decode the data block correctly. One option for using such a code is to have the sender send all of the $n$ symbols, each in a different packet. Consequently, each receiver that does not lose more than $n - K$ packets (symbols) will be able to decode the original data block. Another option is to combine FEC with ARQ (Automatic Repeat Request). Namely, the sender multicasts only $L$ symbols, where $K \leq L \leq n$. A receiver that does not receive at least $K$ packets informs the sender, which may then broadcast some, or even all, of the remaining $n - L$ symbols. An expandable FEC encoder can generate as many unique encoding symbols as requested, such that any $K$ of them is sufficient to reconstruct the original data block.

There are several ways for a hybrid FEC/ARQ scheme to take advantage of the estimation algorithm we proposed. We now describe one of them, which can be implemented using either a block FEC code or an expandable FEC code. The encoder first generates from each data block $L$ symbols, which are broadcast in $L$ independent packets. In order to decode the data block correctly, a receiver must receive at least $K$ of these packets. After transmitting the first $L$ packets, the sender estimates the number of receivers that are missing $i$ packets, for every $i = 1, \ldots, K$. Next, it must determine how many additional packets (symbols) should be transmitted in order to allow additional hosts to pick up sufficient ($K$) symbols for the considered block. To simplify the discussion, we assume that there are only two rounds of transmission. However, the scheme can be generalized to address more rounds as well.

Let $G_i$ be the group of receivers that are missing $i$ packets, for every $i = 1, \ldots, K$. This implies that each member of $G_i$ has received only $K - i$ of the $L$ transmitted packets. Let $R_i = |G_i|$, i.e., the actual number of hosts that are missing exactly $i$ packets. The base station uses our algorithm to estimate the value of $R_i$. To this end, the algorithm is executed in parallel $K$ times, once for every group $G_i$. Practically, this means that a receiver needs to indicate the number $i$ of missing packets in the NACK message it sends, and that the base station needs to broadcast a STOP$(i)$ message on the downlink when it receives sufficient NACKs for $G_i$. Let $\hat{r}_i$ be the estimation of the value of $R_i$.

Since the sender sends $L$ packets in the first round and each receiver in $G_i$ receives only $K - i$ of these packets, each receiver observes a temporary packet loss rate of $p_i = (L - K + i)/L$. Packet loss in a wireless channel is known to be bursty, and it is usually modeled with a good approximation by a low order Markovian chain. Due to the very short time between the two transmission rounds, we assume that each receiver remains in the same state. Therefore, the packet loss probability for each host during the first round is equal to that in the second round.

Let $L'$ be the number of packets sent by the base station during the second round. The probability that a receiver in $G_i$ will receive $i$ of these packets successfully, and thus be able to decode the whole block, is:

$$\text{Prob}(L') = \sum_{j=1}^{L'} \binom{L'}{j} (p_i)^{L' - j} (1 - p_i)^j.$$
ter the first transmission round, but are able to decode it after the second round, is therefore $\text{Ret}(L') = \sum_{i=1}^{K} \text{Prob}_i(L') \cdot R_i$. We also define a function $f(L') = \text{Ret}(L') / \sum_{i=1}^{K} R_i$, which indicates the percentage of these receivers from the total number of bad receivers. We view both $\text{Ret}(L')$ and $f(L')$ as our profit functions.

We simulated a multicast session consisting of 10,000 receivers. In our simulations, the number of packets required for constructing a data block is $K = 10$, while the number of packets transmitted by the sender is $L = 30$. Each receiver may be in a good or bad state of a two-state Markov chain. The probability for being in a good state is 0.75, while that of being in a bad state is 0.25. The packet loss probability in the good state is $10^{-4}$, and in the bad state it is 0.1.

Figure 7(a) shows the value of $\text{Ret}(L')$ as a function of $L'$. This figure contains 4 different curves. However, since all these curves are almost identical, only one curve is visible. The curve labeled “Real” indicates the correct value of $\text{Ret}(L')$ as a function of $L'$. The other 3 curves show the value of $\text{Ret}(L')$ when the sender uses our algorithm to estimate the value of $R_i$ with $N = 5, 20$ and 50 NACKs. It is evident that the profit increases linearly with $L'$, until $L' \approx 25-30$, after which the marginal profit rapidly disappears.

Next, consider Figure 7(b), which shows the value of $f(L')$ as a function of $L'$. Again, only one curve is visible in this graph, because all four curves are identical. For example, if the sender wants about 50% of the receivers that have not received enough packets correctly in the first round to be able to decode the data block after the second, then $L' = 10$ additional packets have to be transmitted. The sender will be able to compute this value of $L'$ correctly even if it uses only $N = 5$ NACKs for estimating the value of $R_i$ for every $i$.

We simulated the same system with other parameters and obtained results similar to those shown in Figure 7. For example, in Figure 8 the number of packets required for constructing a data block is $K = 10$, while the number of packets transmitted by the sender is $L = 15$. The probability for being in a good state is 0.5. The packet loss probability in the good state is $10^{-3}$, while in the bad state it is 0.3.

7. CONCLUSIONS

This paper is the first to explicitly show the correlation between the number of NACKs sent by a group of bad receivers and the ability of the sender to estimate the size of this group precisely. We considered a system where a sender transmits a block of data, consisting of several packets, to a large group of receivers. Upon transmitting these packets, the sender determines the maximum number of NACKs it wants to receive for that block, and the time period $T$ during which these NACKs should be transmitted. Each bad receiver uses a backoff timer, taken from a distribution $F$, to determine when in the interval $[0, T]$ the NACK will be sent. If the base station receives enough NACKs, it immediately sends a STOP message on the downlink, in order to suppress the transmission of additional NACKs. We developed a statistical analysis algorithm for estimating the number of bad receivers, which is based on the times the NACKs are received. The algorithm defines the likelihood function for the received NACKs, and then uses the Newton-Raphson method to find the number of bad receivers for which this function is maximized. We analyzed the error of our algorithm, and showed that when $1 \ll N \ll r$, where $r$ is the number of bad receivers and $N$ is the number of NACKs, this error is positive and approximately equal to $1/(N - 1)$. We used this important result to correct the estimation of our algorithm, and to bring it very close to 0. We also analyzed the estimation error due to NACK losses, and showed that the algorithm performs well also in the presence of such losses. Finally, we have shown the applicability of the proposed scheme for a hybrid FEC/ARQ reliable multicast, where the multicast sender uses the proposed scheme in order to determine the number of parity repair packets to be sent for each data block.

8. REFERENCES


