Improving the Throughput of Point-to-Multipoint ARQ Protocols through Destination Set Splitting

Mostafa H. Ammar
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
(404)894-3292
e-mail: ammar@cc.gatech.edu

Li-Ran Wu
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

June 1992
Revised: March 1995

Abstract

Point-to-Multipoint ARQ protocols are required to guarantee the correct delivery and proper sequencing of messages being sent from one source to multiple destinations. A source of inefficiency for such protocols is that the transmitter requires acknowledgements from *all* receivers before it considers that a message has been correctly received. To address this inefficiency, we propose a scheme in which the set of destinations is split into disjoint groups. The transmitter carries a separate conversation with each group. The conversations are time multiplexed over a single channel, which leads to a tradeoff between the increased throughput as a result of the destination set splitting and the wasted bandwidth required for the multiplexing. We consider memoryless, limited-memory and full-memory versions of the Go-Back-N protocol. We evaluate the maximum throughput achievable with our protocols, and address the issue of selecting the grouping of the destinations that maximizes this throughput. Our approach is beneficial when there are a large number of receivers or when a large timeout threshold is required (e.g., for high speed networks or satellite channels). It is also particularly useful when the set of receivers has dissimilar capabilities. The benefits of our approach are demonstrated through a set of numerical examples.

1 Introduction

Point-to-multipoint communication has been receiving increasing attention in current research. This reflects the growing importance of applications that rely on multi-destination communication, such as information distribution via broadcast [1, 2], teleconferencing systems, and distributed and/or replicated database systems [3, 4]. Of particular importance in such systems is the correctness and timeliness of the delivered information. To achieve the first objective, i.e., correctness, several point-to-multipoint automatic repeat request protocols (ARQ) have been proposed and analyzed [5, 6, 7, 8, 9]. These protocols insure that the data sent from the transmitter to the multiple receivers is received correctly and, in some cases, in the same sequence in which it is transmitted. Of additional concern in the design of these ARQ protocols is the efficiency of the data delivery, typically measured by the ARQ protocol's throughput. ARQ protocols are typically at the heart of reliable transport protocols such as TCP [10]. Point-to-multipoint ARQ protocols are also components of many reliable multicast transport protocols (see for example [11]).

A feature common to all point-to-multipoint ARQ protocols is that a message is not considered to have been received correctly, unless the transmitter has collected acknowledgments from *all* destinations [1]. It is this requirement that makes for potential inefficiencies in the operation of point-to-multipoint ARQ protocols. One approach to compensate for this source of inefficiency is to equip the transmitter with memory in which it can remember the acknowledgements it received as a result of the first transmission of a message and consider the message received if only the missing acknowledgements are received in response to subsequent retransmissions. This memory requirement can become extensive in a continuous ARQ protocol which might allow several messages to be outstanding simultaneously. The effect of full, limited and no memory on the efficiency of point-to-multipoint ARQ protocols has been studied in [7].

A second approach to improve the performance of point-to-multipoint ARQ protocols is the Multiple Copy scheme proposed and analyzed in [5]. In that work the transmitter sends multiple copies of the same message all destined to the set of all receivers. This approach is based on similar work for single destination ARQ protocols (see for example [12]).

In this paper we consider a third approach that can best be motivated by considering the case where a particular destination is unusually slow or is experiencing above average error rates in the messages it is receiving or the acknowledgements it is sending. All destinations except the one experiencing problems may get the message on the first transmission. The ARQ protocol will, however, continue to retransmit the message until this one problem destination gets it. If the transmitter is using either a stop-and-wait or a go-back-N ARQ protocol, even one problem destination can have the effect of substantially reducing the throughput because new message transmissions will be held up pending the receipt of the message by this one destination. The rate at which *all* destinations receive new messages will be substantially lowered because of this one bad destination. The situation above will be more likely to occur if the number of destinations is large or if the error rates are high. It might also occur because of the heterogeneity in the capabilities of the destinations. This degradation in performance can happen independently of whether or not the transmitter retains full, limited or no memory of the acknowledgements received.

[1] In this paper we consider the case where the set of destinations is known and remains fixed for the duration of a conversation.

In the typical environment in which point-to-multipoint ARQ protocols are used, the destinations operate independently and are oblivious to the existence and state of other receivers. It is, therefore, perfectly acceptable for the transmitter to split the set of destinations into groups. The transmitter can then carry on independent conversations with each group using an ARQ protocol. This approach has the advantage that if problems are experienced in some destinations, only the destinations which belong to the same group(s) are affected. The major drawback to this approach is that, since these simultaneous conversations are carried out over the same channel, they have to be multiplexed in time. This can lead to a degradation of throughput. The basic objective of this paper is to consider this tradeoff and investigate the benefit of *destination set splitting* on the throughput of a point-to-multipoint ARQ protocol. In this paper we will consider *memoryless*, *limited-memory*, and *full-memory* ARQ protocols. Although we derive our motivation from the situation when the receivers are dissimilar, one of our most surprising results is that destination set splitting can improve the performance of systems where all receivers are statistically identical.

It is perhaps important to contrast our approach with the Multiple Copy approach proposed in [5]. In that work, the transmitter sends multiple copies of the same message, all destined to the *same* set of receivers. In the approach we investigate in this paper, the transmitter also sends multiple copies of the same message. However, each copy is destined to *a different* subset of the set of receivers. It is possible to generalize our protocols in the direction proposed in [5] by having the transmitter send multiple copies to each different receiver group.

In section 2 we describe our assumptions about the network environment and formally describe the operation of the ARQ protocols we are proposing. In section 3 we describe a model for a system using our protocols and investigate their performance and in section 4 we discuss how it may be optimized. Section 5 disusses applying the idea of destination set splitting to the ideal selective repeat point-to-multipoint protocol. Section 6 contains some numerical example and section 7 contains a comparison of the destination set splitting scheme with the Multiple Copy scheme. The paper is concluded in section 8.

2 The Network Environment and The ARQ Protocol

We consider a system with one transmitter and K receivers. The transmitter is connected to the destinations via a subnetwork that can deliver a message to all or a subset of the destinations using the specifications provided by the transmitter. Messages designated for a particular subset of the receivers may or may not be heard by other destinations not in that particular subset. Therefore, we cannot assume that a transmission multicast to a certain subset of receivers can be heard by receivers not in that subset. The subnetwork may loose and/or corrupt messages but can never reorder them. This is typical behavior for satellite links, bus networks, and single-hop packet radio networks. The same behavior can also be enforced in more elaborate subnetworks by procedures at entry and exit points.

Two types of frames can be transmitted over the network: *messages* carrying information from the transmitter to the receivers and *acknowledgements* carrying indications of receipt of messages from the receivers. Errors in the transmission of messages and acknowledgements can be detected and any data received in error is ignored.

The transmitter receives messages (from a higher layer) which are then given sequence numbers

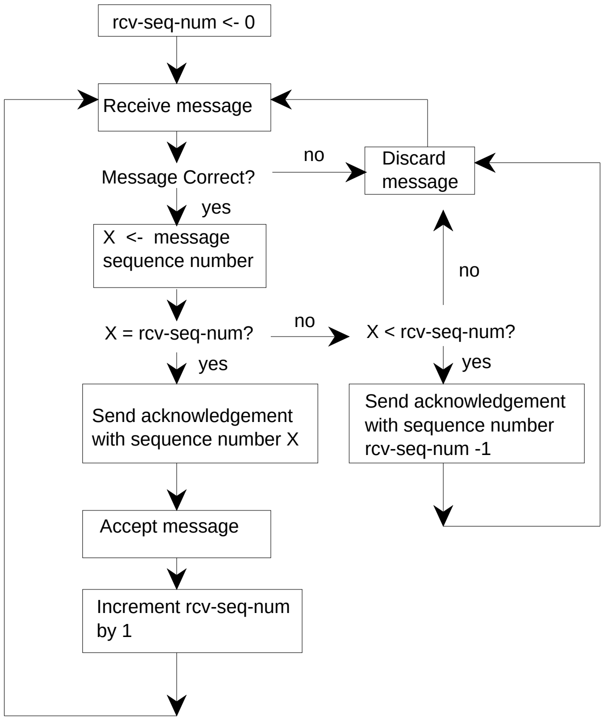

Figure 1: Flow Chart of a Receiver's Operation

starting with 0 and incrementing by 1. The transmitter's objective is to send all the messages it receives from its higher layer to all receivers. We assume that all receivers will accept these messages only if they are received in sequence. Figure 1 shows a flow chart of a receiver's operation. All acknowledgements are assumed to be carried over a separate acknowledgement channel.

The set of receivers is subdivided into g disjoint groups, $(G_1, \ldots, G_g)$. Where $G_i \cap G_j = \phi$ for all $i \neq j$, $G_i \subseteq \{1, 2, \ldots, K\}$ and where $\bigcup_{i=1}^{g} G_i = \{1, 2, \ldots, K\}$. The transmitter carries on g time-multiplexed conversations one with each group using one of the ARQ protocols described below. All g conversations are used to convey the same stream of messages from the transmitter. In other words, all receivers, no matter which group they belong to, will eventually receive the same set of messages. A particular message is considered received in the ith conversation $(i = 1, \ldots, g)$ if the transmitter has collected acknowledgements for that message from all receivers in G_i. The transmitter keeps track of which acknowledgements have been received independently for each of the g conversations.

In this paper we consider the following ARQ protocols: Memoryless Go-Back-N (*m-gbn*), Limited-Memory Go-Back-N (*l-gbn*) and Full-memory Go-Back-N (*f-gbn*). The receivers operate in the manner described above for all protocols. The protocols differ in the way the transmitter operates.

The logical structure for the Go-Back-N transmitter is shown in Figure 2. The transmitter maintains g FCFS message queues one for each receiver subset and inserts a copy of any arriving

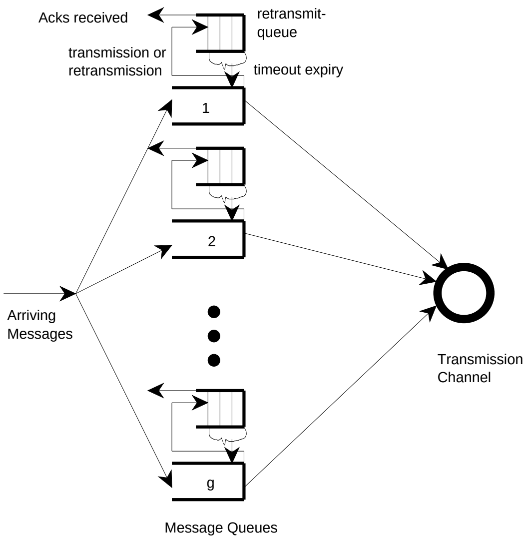

Figure 2: Logical Structure of Go-Back-N Transmitter

message in each of these queues. Transmitted but not-yet acknowledged messages are maintained in a queue called *retrans-q(i)*. When a message is transmitted out of the head of queue i, $i = 1, \ldots, g$, it is copied to the end of *retrans-q(i)*.

The details of the operation of the Go-Back-N protocol are shown in Figure 3. The channel is allocated in a round robin fashion for transmissions for each group. The transmitter processes received acknowledgements and expired timeouts in between consecutive packet transmissions. Each time the channel is allocated to a group, a single message is taken from the head of the message queue of that group (if it is not empty) and multicast to the membership of that group. A transmitted message is then placed at the tail of the *retrans-q* of that group. Whenever a message with sequence number i is multicast to group X, *timeout-counter(X,i)* is started and the *ack-outstanding-list(X,i)* is set to G_X. Messages with an empty *ack-outstanding-list* are cleared from the *retrans-q* only if all messages in the *retrans-q* with lower sequence numbers (if any) also have empty *ack-outstanding-lists*. If *timeout-counter(X,i)* expires, all messages in the *retrans-q* with higher or equal sequence numbers are moved back to the head of the message queue of group X. When this move occurs in the memoryless version of the protocol, the ack-outstanding-lists for all messages moved are reset to G_X. In the limited-memory protocol, the ack-outstanding-list of the message that was at the head of the *retrans-q(X)* [2] is *not* reset and therefore acknowledgements received for that message are remembered. In the full-memory protocol, all ack-outstanding-list's of the messages that were in the *retrans-q(X)* are not reset.

An example of the operation of our Memoryless Go-Back-N protocol is shown in Figure 4.

[2] This is the same as the *First Outstanding Message (FOM)* in [7].

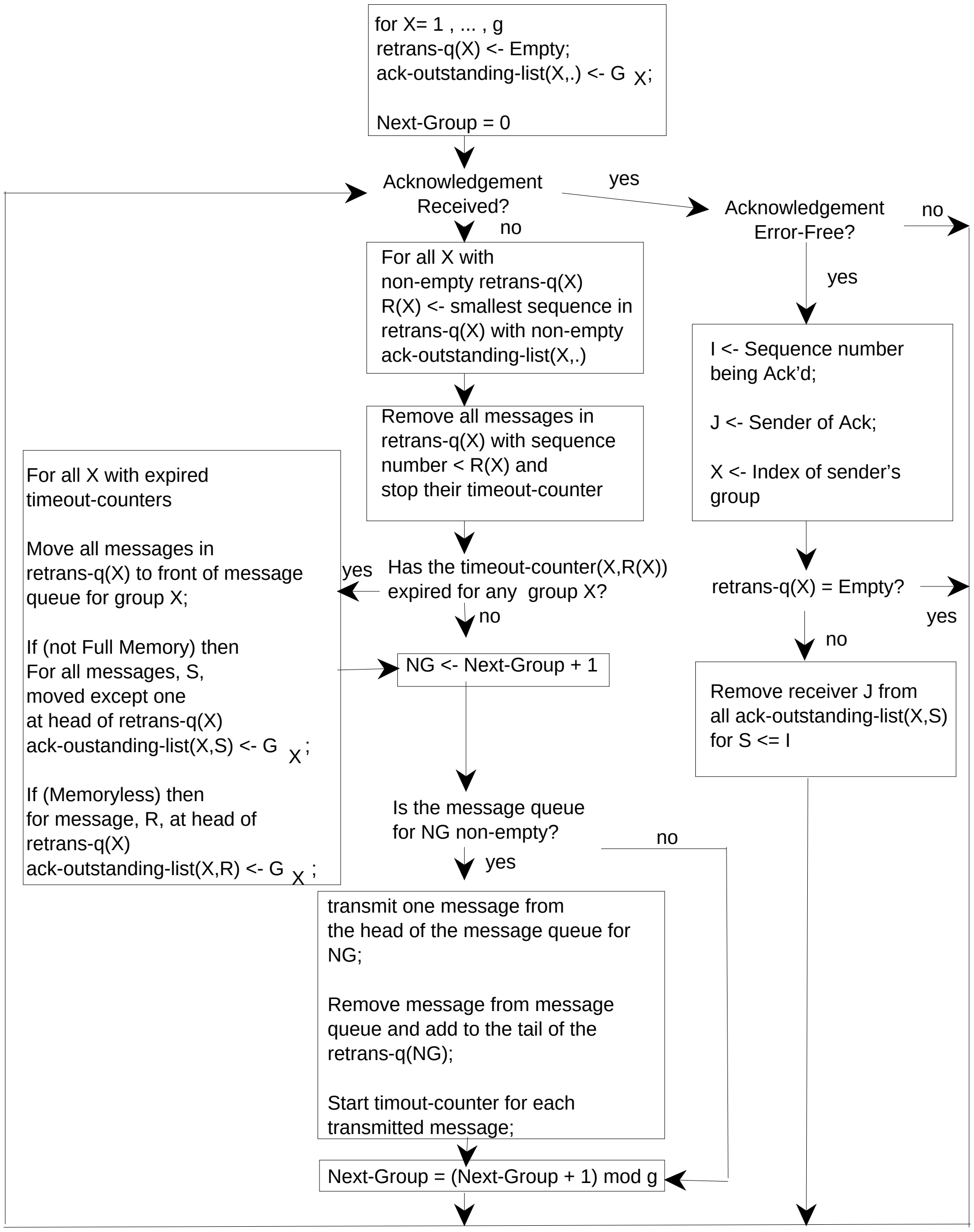

Figure 3: Go-Back-N Protocol Flow Chart

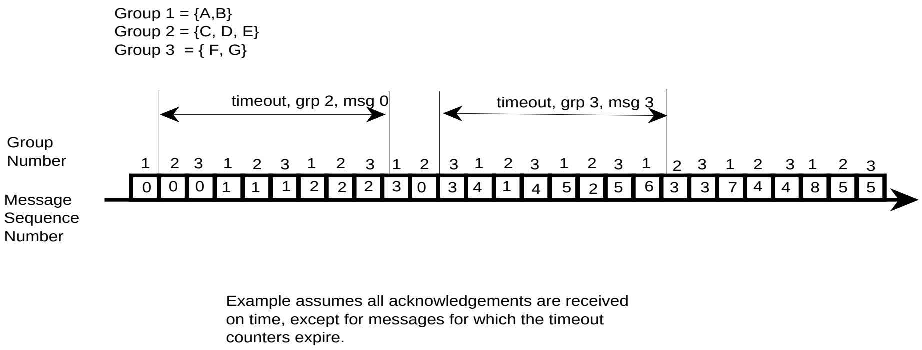

Figure 4: Example of the Operation of a Memoryless Go-Back-N Protocol

3 Performance Evaluation

In this section we investigate the performance of our proposed protocol. Our model and analysis extend the results derived in [7] for our versions of the memoryless and limited-memory ARQ protocols.

3.1 Model

As stated previously, we consider a single transmitter sending messages to K receivers. We assume all messages require the same packet transmission time which we use as our time unit. The timeout threshold is measured from the start of a message's transmission and is denoted by T_o. Note that, although the protocols are defined such that different timeout thresholds may be used for different groups, we assume that the timeout thresholds are the same for all groups. We also assume, as in [7], that acknowledgements are returned the instant prior to the expiry of the timeout counter. Acknowledgements not returned before a timeout counter expires will never be returned.

The set of receivers is subdivided into g groups and we define the vector $\underline{G} = (G_1, ..., G_g)$. The number of receivers in G_i is denoted by m_i and we use $\underline{m}$ to denote the vector $(m_1, m_2, ..., m_g)$.

Two types of errors can occur in the transmission of a message: *group shared* error and *individual receiver* error. If a shared error occurs in a message being transmitted to a group none of the receivers in that group receive the message. If an individual receiver error occurs, only the particular receiver is affected. For a message to be received correctly by a receiver, neither type of error can occur. For a message transmitted to group G_j, the probability of no shared error is denoted by $P_{S,j}$, for $j = 1, \ldots, g$.. The probability of no individual receiver error for receiver i is denoted by P_i for $i = 1, \ldots, K$. We assume that acknowledgements are carried over a separate error-free channel.

We will be concerned with evaluating the normalized maximum throughput of the point-to-multipoint ARQ protocol. This is defined as the average number of correct and new message *receptions* per unit time divided by the total number of receivers, K, under the assumption that the transmitter has an infinite supply of new messages to transmit. In general, this maximum throughput will be a function of the protocol and the grouping being used. We use $\gamma_{protocol}(\underline{G})$ to denote the throughput of a protocol using grouping $\underline{G}$. Our analysis covers Memoryless and

Limited-Memory Go-Back-N protocols. We use simulation to explore the performance of the Full-Memory Go-Back-N protocol.

3.2 Throughput of the Memoryless Go-Back-N Protocol

When our Go-Back-N protocol is used by a saturated transmitter, transmissions will be in cycles. Each cycle is of length g and during each cycle a new or a retransmitted message is sent to each group. The channel will never be idle. A message not received correctly by all receivers upon its first transmission will be retransmitted upon timeout expiry. This will occur $\lceil \frac{T_o}{g} \rceil$ cycles later. The average number of times that a message at the head of the retransmit queue for group G_i has to be retransmitted before it is received correctly is given by $\frac{1}{P_{S,i} \prod_{j \in G_i} P_j} - 1$. After the successful transmission of a message which requires one time unit, the channel has to be allocated to the other $g-1$ groups before it is available to group G_i again.

This leads to the following expression for the maximum throughput of the memoryless go-back-N protocol

$$\gamma_{m-gbn}(\underline{G}) = \sum_{i=1}^{g} \frac{m_i}{gK(1 + \lceil \frac{T_o}{g} \rceil (\frac{1}{P_{S,i} \prod_{j \in G_i} P_j} - 1))} \quad (1)$$

3.3 Throughput of the Limited-Memory Go-Back-N Protocol

Here the average number of times a message has to be transmitted before it is received at least once by all receivers in group G_i is [7]

$$\frac{1}{P_{S,i}} \sum_{x=0}^{\infty} (1 - \prod_{j \in G_i} (1 - (1 - P_j)^x)) \quad (2)$$

Using similar arguments to those used in the derivation of (1), we can show that

$$\gamma_{l-gbn}(\underline{G}) = \sum_{i=1}^{g} \frac{m_i}{gK(1 - \lceil \frac{T_o}{g} \rceil + \lceil \frac{T_o}{g} \rceil \frac{1}{P_{S,i}} (\sum_{x=0}^{\infty} (1 - \prod_{j \in G_i} (1 - (1 - P_j)^x))))} \quad (3)$$

4 Optimization

A question that arises in the design of our protocol is how to pick the vector $\underline{G}$ such that the maximum throughput is maximized. In principle this can be accomplished by enumeration of all the possible groupings for all the different values of $g = 1, \ldots, K$. This, however, is only feasible for small values of K.

In this section we first consider the special case of systems with homogeneous receivers where $P_i = P_l = P$ for all receivers i and l and $P_{S,j} = P_{S,k} = P_S$ for all groups j and k. Next we consider extending our optimization results to the case of heterogeneous receivers.

4.1 Systems with Homogeneous Receivers

In a system with homogeneous receivers the throughput will only depend on the vector $\underline{m}$ of group sizes and not on the particular group constitution. For such systems we can write

$$\gamma_{m-gbn}(\underline{G}) = \gamma_{m-gbn}(\underline{m}) = \sum_{i=1}^{g} \frac{m_i}{gK(1 + \lceil \frac{T_o}{g} \rceil (\frac{1}{P_S P^{m_i}} - 1))} \tag{4}$$

and

$$\gamma_{l-gbn}(\underline{G}) = \gamma_{l-gbn}(\underline{m}) = \sum_{i=1}^{g} \frac{m_i}{gK(1 - \lceil \frac{T_o}{g} \rceil + \lceil \frac{T_o}{g} \rceil \frac{1}{P_S} (\sum_{x=0}^{\infty}(1 - (1 - (1 - P_j)^x)^{m_i})} \tag{5}$$

For a homogeneous system the optimization problem can be stated as follows:

$$\max_{\underline{m},g} \gamma_{protocol}(\underline{m}) = \max_{\underline{m},g} \sum_{i=1}^{g} \mathcal{F}_{protocol}(m_i) \tag{6}$$

$$such\ that\ \sum_{i=1}^{g} m_i = K; \quad 1 \le m_i \le K - g + 1; \quad m_i, g\ integer$$

where $\mathcal{F}_{m-gbn}(m_i)$ and $\mathcal{F}_{l-gbn}(m_i)$ are the terms inside the summations in (4) and (5), respectively.

We characterize the solution to the optimization problem above, by first relaxing the integer constraint on the m_i's. The following theorem gives a sufficient condition for the optimal value of $\underline{m}$ to be where $m_i = m_j = \frac{K}{g}$.

Theorem 1 For given g, T_o, and P, let $\eta_{protocol}(P, T_o, g)$ be the solution of

$$\frac{\partial^2}{\partial m^2} \mathcal{F}_{protocol}(\underline{m}) = 0 \tag{7}$$

If

$$\eta_{protocol}(P, T_o, g) \ge \frac{K}{g} \tag{8}$$

then the $\max_{\underline{m}} \gamma_{protocol}(\underline{m})$ is achieved for $m_i = m_j = \frac{K}{g}$ for all $i, j = 1, \ldots, g$.

Proof See Appendix A.

The implication of Theorem 1 is the following:

> If the protocol parameters are such that (8) is satisfied for all g then the solution to the optimization problem in (6) with a relaxation of the integer constraint on the m_i's can be found in the set of solutions:
>
> $$\mathcal{M} = \{(m_i = \frac{K}{g}, i = 1, \ldots, g) | g = 1, \ldots, K\} \tag{9}$$

For the limited-memory protocol it can be shown that $\mathcal{F}_{l-gbn}(m_i)$ is a monotonically increasing function of m_i and thus $\eta_{l-gbn}(\bullet) = \infty$. Therefore the sufficient condition in Theorem 1 is always satisfied and thus it is sufficient to consider the solutions in the set $\mathcal{M}$. We actually require a solution with integer m_i's and thus will consider the following set of solutions [3]:

$$\mathcal{M}_I = \{(m_i = \lceil \frac{K}{g} \rceil, i = 1, \ldots, K - \lfloor \frac{K}{g} \rfloor g, m_i = \lfloor \frac{K}{g} \rfloor, i = K - \lfloor \frac{K}{g} \rfloor g + 1, \ldots, g) | g = 1, \ldots, K\} \quad (10)$$

For the memoryless protocol, the situation is somewhat more complex. For a given set of parameters, T_o, P, and K, there exists a value $g_0 = \lceil \frac{K}{\eta_{protocol}(\bullet)} \rceil$ for which the condition in (8) is true if $g \geq g_0$ and does not hold for $g < g_0$. Only if $g_0 \leq 2$ [4] can we assert that the optimum may be found in $\mathcal{M}$ for all values of g. Fortunately, this condition is true for a wide range of realistic values of T_o, P, and K. This can be seen by examining table 1 which shows the values of $\eta_{m-gbn}(P, T_o, g)$ [5].

The results in this section can be summarized as follows:

- For the limited-memory protocol with homogeneous receivers a near-optimal solution to (6) can be found by considering only the K solutions in the set $\mathcal{M}_I$.
- For the memoryless protocols, the set $\mathcal{M}_I$ can be proven to sometimes (when $g_0 \leq 2$) contain a near-optimal solution.

As a result, the set $\mathcal{M}_I$ will, with high likelihood contain a near-optimal grouping. Therefore, we will consider only the groupings identified in $\mathcal{M}_I$ even if the conditions of Theorem 1 are not satisfied.

4.2 Systems with Heterogeneous Receivers

For systems with a large number of heterogeneous receivers where optimization by enumeration is not feasible we propose the heuristic described below. The heuristic starts with solutions based on the homogeneous system solution. These solutions are then refined to obtain a near-optimal solution.

Optimization Heuristic:

(We assume without loss of generality that the K receivers are labelled such that $P_1 \geq P_2 \geq \ldots \geq P_K$. Also the Heuristic uses a REFINE procedure which is described in Figure 5.)

1. For $g = 1, \ldots, K$, Determine $\underline{G}^{(g)} = (G_1^{(g)}, \ldots, G_g^{(g)})$ where $G_i^{(g)} = \{m_{i-1}^{(g)} + 1, \ldots, m_i^{(g)}\}$. and where the $m_i^{(g)}$'s are determined by (10) ($m_0^{(g)} = 0$).

[3] In general the integer solution in (10) might not be the optimal integer solution to (6). We use it as a good near-optimal solution

[4] Note that when $g = 1$ the optimization problem is trivial.

[5] By solving for m in (7) we can show that for the memoryless protocols: $\eta_{m-gbn}(P, g, T_o) = \eta_{m-gbn}(P, \lceil \frac{T_o}{g} \rceil)$. The value of $\eta_{m-gbn}(\bullet)$ can be solved for numerically using (7).

P	$\eta_{m-gbn}(P, \lceil \frac{T_o}{g} \rceil)$			
	$\lceil \frac{T_o}{g} \rceil$			
	1	2	4	7
0.5	2.88	2.37	1.64	1.64
0.6	3.91	3.22	2.63	2.28
0.8	8.96	7.39	6.03	5.10
0.9	18.98	15.65	12.79	10.80
0.95	38.99	32.15	26.27	22.18
0.99	198.99	164.11	134.07	113.24
0.999	1998.99	1648.54	1346.80	1137.57
0.9999	19999	16491	13474	11380

Table 1: Table of $\eta_{m-gbn}(P, \lceil \frac{T_o}{g} \rceil)$ Values

2. For $g = 1, \ldots, K$, $\underline{H}^{(g)} = REFINE(\underline{G}^{(g)})$.
3. The near optimal solution is the $\underline{H}^{(g)}$, $g = 1, \ldots, K$ that maximizes the value of $\gamma_{protocol}(\underline{H}^{(g)})$.

The *REFINE* procedure is described in Figure 5.

5 The Ideal Selective Repeat Protocol

We consider the effect of destination set splitting on the performance of the *ideal selective repeat (i-sr)* point-to-multipoint ARQ protocols. This protocol is the one considered in [7, 13]. In this idealized protocol, the transmitter only retransmits messages in error and retains full-memory of acknowledgements received for all messages. Infinite buffer space is required at the receivers to handle packet resequencing. Using similar arguments to those used in the derivation of (1) and (3), we can show that

$$\gamma_{i-sr}(\underline{G}) = \sum_{i=1}^{g} \frac{m_i P_{S,i}}{gK(\sum_{x=0}^{\infty}(1 - \prod_{j \in G_i}(1 - (1 - P_j)^x)))} \tag{11}$$

An upper bound on the performance of our Go-Back-N protocols is given by $\gamma_{i-sr}(\underline{G^*})$ where $\underline{G^*}$ is the grouping that maximizes (11). As before we consider two cases corresponding to homogeneous and heterogeneous receivers. For the homogeneous receivers case we can prove the following:

Lemma 1 For the ideal selective repeat point-to-multipoint ARQ protocol with homogeneous receivers

$$\gamma_{i-sr}(\{1, \ldots, K\}) \geq \gamma_{i-sr}(\underline{G})$$

for any grouping $\underline{G}$.

```
REFINE(G)
/* ℓ is the number of groups in G and G is defined by m_i, i = 1,...,ℓ*/
   G_old = G_new = G; i = 1;
   Do While (m_{i+1} ≠ 0 and i ≠ ℓ)
        Do While (γ(G_new) ≥ γ(G_old) )
             G_old = G_new; G_new = Shift (G, i);
        End While;
        i = i + 1 G_new = G_old ;
   End While;
   Return (G_old);
End REFINE;

Shift (G, i);
   m_i = m_i + 1 m_ℓ = m_ℓ − 1;
   If (m_ℓ = 0) then ℓ = ℓ − 1;
End Shift;
```

Figure 5: The REFINE Procedure

Proof: See Appendix B.

The above lemma indicates that destination set splitting does not improve the performance of the ideal selective repeat protocol for the homogeneous receivers case.

By considering a specific example we show that destination set splitting *can* improve the performance of the ideal selective repeat protocol in the heterogeneous receivers case. We consider a two receiver system where $P_1 \neq P_2$. Assuming no shared error we have:

$$\gamma_{i-sr}(\{1,2\}) = \left[\frac{1}{P_1} + \frac{1}{P_2} - \frac{1}{P_1 + P_2 - P_1 P_2}\right]^{-1}$$

and

$$\gamma_{i-sr}(\{1\},\{2\}) = \frac{P_1 + P_2}{4}$$

Destination set splitting (into two groups) will improve performance in this case if

$$\frac{P_1 + P_2}{4} > \left[\frac{1}{P_1} + \frac{1}{P_2} - \frac{1}{P_1 + P_2 - P_1 P_2}\right]^{-1} \tag{12}$$

Figure 6 shows the values (shaded regions) of P_1 and P_2 for which the two-group operation achieves better maximum throughput than the single group operation. We thus conclude that the single group ideal selective repeat system is not necessarily the “ideal” system when the receivers are of different capabilities.

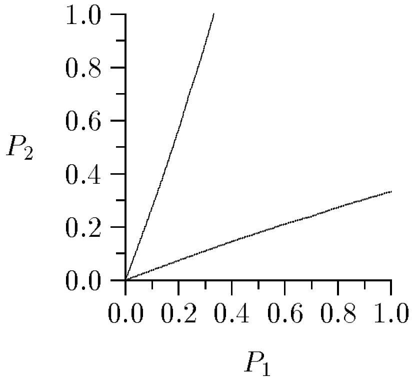

Figure 6: Values of P_1 and P_2 for which Destination Set Splitting Improves the Throughput of the Ideal Selective Repeat Protocol.

6 Numerical Examples

We illustrate the performance benefits of destination set splitting by consideration of some examples. We assume throughout that shared errors do not occur and, therefore, $P_S = 1$.

6.1 Systems with Homogeneous Receivers

Figure 7 illustrates the maximum throughput achievable in the memoryless Go-Back-N protocol with $P = 0.99$ as a function of the number of receivers, K. Cases with timeout threshold $T_o = 5$ and 50 are considered. The low value is typical for traditional terrestrial, low-speed communication. High values of T_o need to be used in long propagation delay systems or for high speed connections. For each value of T_o we show the maximum throughput of memoryless Go-Back-N when a single group is used and when the optimal grouping (chosen from the set $\mathcal{M}_I$) is used. Clearly optimal splitting can achieve throughput improvements, even when the receivers are statistically identical. For this value of P, destination set splitting does not improve the throughput for low values of K. The value of K at which splitting is beneficial is lower for higher values of the timeout threshold. Note that the benefit of destination set splitting is most apparent for high values of K, where the ratio of the maximum throughput with destination set splitting to that without is very high.

We also evaluated the performance of limited memory and full-memory Go-Back-N protocols with and without destination set splitting using simulation for the latter. Although some increase in maximum throughput was possible through the use of destination set splitting, the improvement *for homogeneous receiver systems* was small and was only observed for large values of the timeout threshold.

6.2 Systems with Heterogeneous Receivers

We first consider a system with four receivers. For such a system 15 different groupings are possible. For each protocol and for given values of P_i, $i = 1, \ldots, 4$, and for T_o, it is possible to evaluate the

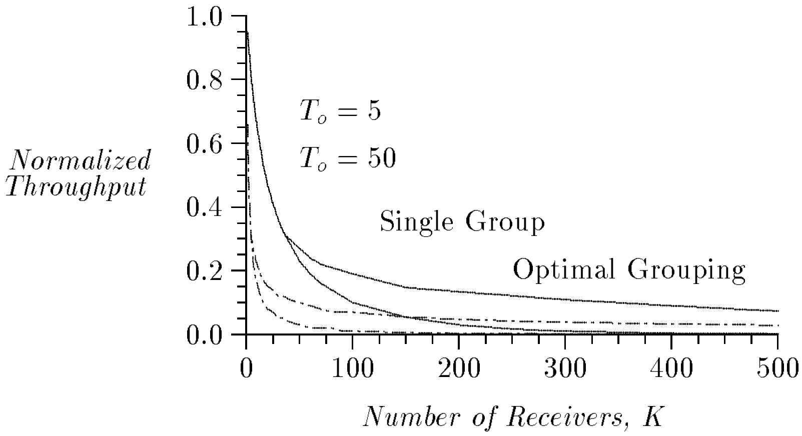

Figure 7: Throughput of the Memoryless Go-Back-N Protocol with and without Destination Set Splitting. Homogeneous, $P_S = 1.0$, $P = 0.99$, $T_o = 5, 50$

maximum throughput for each possible grouping and determine the best one. We experimented with 6 different settings for the P_i's and two values of T_o. The results of our findings are shown in Table 2 for $T_o = 10$ and in Table 3 for $T_o = 100$. In the tables we show the optimal grouping for a given protocol, the maximum throughput achieved by that grouping and the ratio of this maximum throughput to that achieved without destination set splitting, i.e., when all receivers are in the same group.

For the cases we considered, significant throughput improvements can be achieved (even for this small number of receivers). In one instance the throughput is improved by more than 10 times by using the appropriate destination grouping. As before the throughput improvement is more pronounced when T_o is large. Note that, unlike the homogeneous receiver case, the limited-memory protocol's performance is significantly affected by destination set splitting.

The optimal groupings obtained can sometimes be intuitively derived. For example when the P_i's are (0.99,0.99,0.99,0.75) the optimal grouping when $T_o = 10$ separates the "bad" receiver into its own group. However, when $T_0 = 100$, one of the "good" receivers is also separated out, a non-intuitive optimal grouping.

Next we consider three examples for systems with non-identical receivers and a large number of nodes.

- **Example 1:** $P_1 = 0.9999$ for $i = 1, \ldots, K-1$, and $P_K = 0.75$.
- **Example 2:** $P_i = 0.9999$ for $i = 1, \ldots, \lfloor \frac{K}{4} \rfloor + \lfloor \frac{K}{2} \rfloor$, $P_i = 0.75$ for $i = \lfloor \frac{K}{4} \rfloor + \lfloor \frac{K}{2} \rfloor + 1, \ldots, K$.

Example 1 represents a set of receivers with only one "bad" receiver, i.e., a receiver with a low probability of correct reception. Example 2 represents a set of receivers where approximately 3/4 of the receivers are "good" and approximately 1/4 are "bad".

We investigate the effect of destination set splitting for these three examples by applying the heuristic we developed in the previous section. In Tables 4,5 and 6 we consider the maximum

(P_1, P_2, P_3, P_4)	m-gbn, $T_o = 10$			l-gbn, $T_o = 10$		
	Optimal Grouping	Optimal Throughput	Ratio to Single Group	Optimal Grouping	Optimal Throughput	Ratio to Single Group
0.99, 0.95, 0.90, 0.85	(1,2)(3,4)	0.29	1.4	(1,2)(3,4)	0.30	1.2
0.99, 0.89, 0.79, 0.69	(1,2)(3,4)	0.20	2.4	(1,2)(3,4)	0.21	1.7
0.99, 0.99, 0.90, 0.90	(1,2)(3,4)	0.34	1.2	(1,2)(3,4)	0.35	1.1
0.99, 0.99, 0.75, 0.75	(1,2)(3,4)	0.28	2.5	(1,2)(3,4)	0.29	2.1
0.99, 0.99, 0.99, 0.90	(1,2,3,4)	0.41	1	(1,2,3,4)	0.42	1
0.99, 0.99, 0.99, 0.75	(1,2,3)(4)	0.37	1.7	(1,2,3)(4)	0.37	1.7

Table 2: Four-Receiver Heterogeneous Systems and their Optimal Groupings. $T_o = 10$, and $P_S = 1.0$ for all groups.

(P_1, P_2, P_3, P_4)	m-gbn, $T_o = 100$			l-gbn, $T_o = 100$		
	Optimal Grouping	Optimal Throughput	Ratio to Single Group	Optimal Grouping	Optimal Throughput	Ratio to Single Group
0.99, 0.95, 0.90, 0.85	(1)(2)(3,4)	0.11	4.3	(1)(2)(3,4)	0.11	3.6
0.99, 0.89, 0.79, 0.69	(1)(2)(3,4)	0.08	9.1	(1)(2,3,4)	0.09	6.8
0.99, 0.99, 0.90, 0.90	(1,2)(3,4)	0.14	3.9	(1,2)(3,4)	0.15	3.5
0.99, 0.99, 0.75, 0.75	(1,2)(3,4)	0.13	10.8	(1,2)(3,4)	0.13	8.2
0.99, 0.99, 0.99, 0.90	(1,2)(3)(4)	0.18	2.8	(1,2)(3)(4)	0.18	2.6
0.99, 0.99, 0.99, 0.75	(1,2)(3)(4)	0.17	6.4	(1,2)(3)(4)	0.17	6.2

Table 3: Four-Receiver Heterogeneous Systems and their Optimal Groupings. $T_o = 100$, and $P_S = 1.0$ for all groups.

	Example 1, $T_0 = 10$		Example 2, $T_0 = 10$		Example 1, $T_0 = 100$		Example 2,$T_0 = 100$	
K	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping
5	0.230	0.437	0.114	0.340	0.029	0.397	0.012	0.300
10	0.230	0.467	0.068	0.368	0.029	0433	0.007	0.340
30	0.229	0.482	0.011	0.366	0.028	0.423	0.001	0.330
50	0.227	0.482	0.002	0.364	0.028	0.393	0.0002	0.312

Table 4: Maximum Throughput for Eaxmples 1 and 2 with and without Destination Set Splitting using Memoryless Go-Back-N, $T_o = 10, 100$, and $P_S = 1.0$ for all groups.

	Example 1, $T_0 = 10$		Example 2, $T_0 = 10$		Example 1, $T_0 = 100$		Example 2,$T_0 = 100$	
K	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping
5	0.231	0.437	0.143	0.350	0.029	0.398	0.016	0.302
10	0.230	0.466	0.109	0.378	0.029	0.433	0.012	0.341
30	0.229	0.483	0.064	0.379	0.029	0.423	0.007	0.332
50	0.228	0.482	0.053	0.378	0.029	0.394	0.005	0.314

Table 5: Maximum Throughput for Examples 1 and 2 with and without Destination Set Splitting using Limited-Memory Go-Back-N, $T_o = 10, 100$, and $P_S = 1.0$ for all groups.

throughput for memoryless, limited- and full-memory Go-Back-N protocols for Examples 1 and 2 above. We show, for two values of the timeout threshold, the maximum throughput achievable when all receivers are in a single group and when a near-optimal grouping obtained by our heuristic is used. In both examples this near-optimal grouping consists of two groups, one with all the “good” receivers and the other with all the “bad” receivers.

The use of limited or full memory does not substantially affect the performance in Example 1. While splitting off the “bad” receiver into its own group almost doubles the maximum throughput. We note that in Example 1 receiver K is a “bottleneck” to the throughput of the system and that this bottleneck seems to dictate the throughput of the system regardless of the number of receivers or the type ef memory used. The effect of this bottleneck can only be dealt with through destination set splitting. For Example 2, because almost one quarter of the receivers are bottleneck receivers, more protocol memory has a substantial effect when a single group is used. However, splitting off the “bad” receivers has a more significant effect. Finally, we note that in this heterogeneous environment, a large timeout value results in a significant reduction in the throughput when all receivers are in a single group. When destination set splitting is used this effect is not as pronounced.

	Example 1, $T_0 = 10$		Example 2, $T_0 = 10$		Example 1, $T_0 = 100$		Example 2,$T_0 = 100$	
K	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping	Single Group	Near-Optimal Grouping
5	0.230	0.437	0.203	0.362	0.029	0.400	0.028	0.306
10	0.232	0.467	0.192	0.392	0.029	0.432	0.028	0.345
30	0.230	0.482	0.169	0.395	0.028	0.428	0.027	0.338
50	0.228	0.482	0.160	0.392	0.029	0.408	0.027	0.325

Table 6: Maximum Throughput for Examples 1 and 2 with and without Destination Set Splitting using Full-Memory Go-Back-N, $T_o = 10, 100$, and $P_S = 1.0$ for all groups.

7 Comparison with the Multiple Copy Scheme

We now undertake a comparison of the destination set splitting (DSS) scheme with the Multiple Copy (MC) [5, 12] scheme. Recall that in the Multiple Copy scheme the source sends multiple copies of each message with each copy destined to the entire destination group.

There are two main points of comparison: 1) the number of returned acknowledgements per message and 2) the maximum achievable throughput. The MC scheme has the potential of generating many unnecessary returned acknowledgements. Consider for example a DSS scheme maintaining ℓ groups and a DC scheme that sends ℓ copies of each message. In both scenarios, the source is expending the same amount of effort. Assuming all transmissions are successful, each receiver will each send ℓ acknowledgements per message in the MC scheme and 1 acknowledgement per message in the DSS scheme. This is clearly an advantage of our DSS scheme over the MC scheme.

In order to compare the maximum achievable throughput for both schemes, we first observe that the two schemes can be combined. That is one can run multiple conversations using destination set splitting as described in this paper and then send multiple copies within each conversation. *Assuming no shared errors,* and under the same assumptions used for the derivation of equations (1) and (3), the throughput of systems using a combined MC and DSS scheme with c_i copies being sent to members of group i is given by

$$\gamma_{m-gbn}(\underline{G}) = \sum_{i=1}^{g} \frac{m_i}{LK(1 + \lceil \frac{T_o}{L} \rceil (\frac{1}{\prod_{j \in G_i} P_{ji}} - 1))} \tag{13}$$

$$\gamma_{l-gbn}(\underline{G}) = \sum_{i=1}^{g} \frac{m_i}{LK(1 - \lceil \frac{T_o}{L} \rceil + \lceil \frac{T_o}{L} \rceil (\sum_{x=0}^{\infty}(1 - \prod_{j \in G_i}(1 - (1 - P_{ji})^x))))} \tag{14}$$

where $L = \sum_{i=1}^{g} c_i$ and $P_{ji} = 1 - (1 - P_j)^{c_i}$.

Using the expressions above it can be shown that *for the homogeneous receivers case*, the pure MC (i.e., with no destination set splitting) scheme can provide higher maximum throughput than our pure DSS (i.e., with single copy to each group) scheme. This is, of course, at the expense

K	Single Grp	Pure DSS	Pure MC	
				copies
5	0.029	0.398	0.228	4
10	0.029	0.433	0.228	4
30	0.029	0.423	0.228	4
50	0.029	0.394	0.228	4

Table 7: Comparison of Maximum Throughput for DSS and MC Schemes, Example 1, Limited-Memory Go-Back-N

K	Single Grp	Pure DSS	Pure MC		Combined DSS/MC	
				copies		copies
5	0.016	0.126	0.209	4	0.204	2,1
10	0.012	0.082	0.193	4	0.231	2,1
30	0.007	0.035	0.173	5	0.230	2,1
50	0.005	0.024	0.159	5	0.220	2,1

Table 8: Comparison of Maximum Throughput for DSS, MC and Combined Schemes, Example 3, Limited-Memory Go-Back-N

of more processing by the receivers and more returned acknowledgements. No advantage can be gained in the homogeneous case from combining the destination set splitting with the MC scheme.

For the heterogeneous receivers case, DSS and combined DSS/MC can give higher throughputs than a pure MC scheme. This is illustrated in tables 7 and 8 which show results for the limited-memory protocol. Table 8 shows for Example 1 (shown in Section 6.2) the maximum achievable throughput for pure DSS and pure MC schemes. This is compared to the single group, single copy scheme. The number of copies required to achieve the best MC throughput is also shown. The pure DSS scheme is clearly superior in this case. It also results in fewer acknowledgements returned per message. For this particular example there is no advantage that can be gained from combining the two schemes.

Table 8 shows results for the following example:
Example 3: $P_i = 0.99$ for $i = 1, \ldots, \lfloor \frac{K}{4} \rfloor + \lfloor \frac{K}{2} \rfloor$, $P_i = 0.75$ for $i = \lfloor \frac{K}{4} \rfloor + \lfloor \frac{K}{2} \rfloor + 1, \ldots, K$.
For this case the best throughput is achieved by a combined DSS/MC scheme. In such a scheme, the "good" receivers are split off from the "bad" receivers. Two copies of each message are transmitted to the "good" receiver group and one copy of each message to the "bad" receiver group.

8 Concluding Remarks

We have considered an approach to improving the performance of point-to-multipoint ARQ protocols. The approach involves splitting the destination set into disjoint groups and having the source carry out independent, time-multiplexed conversations with each group. The performance of memoryless, limited-, and full-memory versions of the Go-Back-N protocol was investigated and the question of how to pick an optimal grouping of nodes was addressed. For systems with homogeneous receivers we found that an equal size grouping is almost always optimal or near-optimal. In the homogeneous case, the benefit of destination set splitting is not significant when full-memory is used. For systems with heterogeneous receivers destination set splitting seems to be an effective way to deal with "bottleneck" receivers and can provide a significant improvement in throughput even when full-memory is used. Compared to the Multiple Copy scheme, the destination set splitting technique results in fewer returned acknowledgements for the same number of source transmissions. Our splitting scheme can also result in higher maximum throughput either in its pure form or when combined with the Multiple Copy scheme.

We have recently investigated the use of the destination set splitting technique to improve the performance of window-based congestion control in a multipoint conversation [15]. We are also currently investigating the feasibility of using a splitting technique to deal with heterogeneity in receivers in a point-to-multipoint video distribution scenario.

Appendix A : Proof of Theorem 1

Consider a point $\underline{m}^* = (m_1^*, \ldots, m_g^*)$ that satisfies for a given g, $\sum_{i=1}^{g} m_i^* = K$ and satisfies

$$\nabla(-\gamma_{protocol}(\underline{m}^*)) + \lambda\nabla(\sum_{i=1}^{g} m_i^* - K) = 0 \tag{15}$$

For such a point to be a maximum of $\gamma_{protocol}(\underline{m})$, it is sufficient that the matrix

$$\mathcal{L}(\underline{m}^*) = \nabla^2(-\gamma_{protocol}(\underline{m}^*)) + \lambda\nabla^2(\sum_{i=1}^{g} m_i^* - K) \tag{16}$$

be positive definite (see [14, p307]).

The condition in (15) translates into the following set of equations:

$$\frac{\partial}{\partial m_i}\mathcal{F}_{protocol}(m_i^*) = \lambda \qquad i = 1, \ldots, g \tag{17}$$

The matrix $\mathcal{L}$ is given by

$$\mathcal{L}(\underline{m}^*) = [\frac{\partial^2}{\partial m_1^2}(-\mathcal{F}_{protocol}(m_1^*)), \ldots, \frac{\partial^2}{\partial m_g^2}(-\mathcal{F}_{protocol}(m_g^*))] \quad \mathcal{I} \tag{18}$$

where $\mathcal{I}$ is a $g \times g$ identity matrix. The matrix $\mathcal{L}$ is positive definite if and only if its eigenvalues are > 0. In other words we need to have $\frac{\partial^2}{\partial m_i^2}(-\mathcal{F}_{protocol}(m_i^*)) > 0$. Or alternatively we must have

$$\frac{\partial^2}{\partial m_i^2}\mathcal{F}_{protocol}(m_i^*) < 0 \tag{19}$$

Let us consider the point $m_i^* = K/g$ for $i = 1, \ldots, g$. This point satisfies the equations in (17). A sufficient condition for this point to represent a maximum is for it to satisfy (19).

We define $\eta_{protocol}(P, T_o, g)$ as the solution of $\frac{\partial^2}{\partial m_1^2}\mathcal{F}_{protocol}(m_1^*) = 0$. It can be shown that $\frac{\partial^2}{\partial m_1^2}\mathcal{F}_{protocol}(m_1^*)$ is monotonically increasing in the range $[1, \eta_{protocol}(P, T_o, g)]$.

Thus the condition in (19) is satisfied, and by implication the point $m_i^* = K/g$ for $i = 1, \ldots, g$ is a maximum, when $\eta_{protocol}(P, T_o, g) > K/g$ which is the sufficient condition we are seeking and the theorem is proved.

Appendix B : Proof of Lemma 1

We first observe that by virtue of Theorem 1 we can show that the optimal solution (to the relaxed optimization) of (6) can be found in the set $\mathcal{M}$. Therefore, for any given number of groups g we have that

$$\gamma_{i-sr}(\underline{m} = (m_1, \ldots, m_g)) \leq \frac{P_S}{g(\sum_{x=0}^{\infty}(1 - (1 - (1 - P)^x)^{\frac{K}{g}})}$$

In the single group case we have

$$\gamma_{i-sr}(\{1, \ldots, K\}) = \frac{P_S}{\sum_{x=0}^{\infty}(1 - (1 - (1 - P)^x)^K)}$$

Therefore in order to demonstrate that $\gamma_{i-sr}(\{1, \ldots, K\}) \geq \gamma_{i-sr}(\underline{m})$, it is sufficient to show that

$$g(\sum_{x=0}^{\infty}(1 - (1 - (1 - P_j)^x)^{\frac{K}{g}}) \geq \sum_{x=0}^{\infty}(1 - (1 - (1 - P)^x)^K)$$

For the above to be true it is sufficient that $\mathcal{A}(g) = g((1 - (1 - (1 - P)^x)^{\frac{K}{g}})$ be a monotonically increasing function of g for all values of x.

Taking the derivative of $\mathcal{A}(g)$ we get

$$\frac{\partial}{\partial g}\mathcal{A}(g) = 1 - \gamma + \gamma \ln \gamma \tag{20}$$

where

$$\gamma = (1 - (1 - P)^x)^{K/g}$$

Using the Taylor series expansion for $\ln \gamma$ we get

$$\frac{\partial}{\partial g}\mathcal{A}(g) = (1 - \gamma)^2(1 - (\gamma \sum_{i=0}^{\infty} \frac{(1 - \gamma)^i}{(i + 2)})) \tag{21}$$

But we have that

$$\frac{1}{\gamma} = \sum_{i=0}^{\infty}(1 - \gamma)^i \geq \sum_{i=0}^{\infty} \frac{(1 - \gamma)^i}{i + 2}$$

Therefore

$$\frac{\partial}{\partial g}\mathcal{A}(g) \geq (1 - \gamma)^2(1 - \gamma\frac{1}{\gamma}) = 0 \tag{22}$$

Therefore $\mathcal{A}(g)$ is an increasing function of g and the lemma is proven.

References

[1] D. Gifford, "Polychannel systems for mass digital communication," *Communications of the ACM*, vol. 33, pp. 141–151, February 1990.

[2] J. W. Wong and M. H. Ammar, "Analysis of broadcast delivery in a videotex system," *IEEE Transactions on Computer*, vol. 34, pp. 863–866, September 1985.

[3] D. Gifford, "Weighted voting for replicated data," in *Proceedings of 7th Symposium on Operating Systems*, pp. 150–162, ACM, 1979.

[4] M. Ahamad and M. H. Ammar, "Performance characterization of quorum-consensus algorithms for replicated data," *IEEE Transactions on Software Engineering*, vol. 15, pp. 492–496, April 1989.

[5] J. L. Wang and J. A. Silvester, "Optimal adaptive ARQ protocols for point-to-multipoint communication," in *Proceedings of INFOCOM '88*, pp. 704–713, IEEE Computer Society, 1988.

[6] J. S. J. Daka and A. G. Waters, "A high performance broadcast file protocol," in *Proceedings of SIGCOMM '88 Synposium*, pp. 274–281, ACM, 1988.

[7] I. S. Gopal and J. M. Jaffe, "Point-to-multipoint communication over broadcast," *IEEE Transactions on Communications*, vol. 32, pp. 1034–1044, September 1984.

[8] D. Towsley, "An analysis of a point-to-multipoint channel using a go-back-n error control protocol," *IEEE Transactions on Communications*, vol. 33, pp. 282–285, March 1985.

[9] S. B. Calo and M. C. Easton, "A broadcast protocol for file transfers to multiple sites," *IEEE Transactions on Communications*, vol. 29, pp. 1701–1707, November 1981.

[10] A. Tannenbaum, *Computer Networks,* 2nd Ed., Addison Wesley, 1988.

[11] J. Crowcroft and K. Paliwoda, "A multicast transport protocol," Proceedings of ACM SIGCOMM, pp. 247-256, 1988.

[12] Y. T. Yukuo Hayashida, Shumji Fujii, "Delay performance of a continuous ARQ system with copy-transmissions," in *Proceedings of INFOCOM '88*, pp. 714–721, IEEE Computer Society, 1988.

[13] K. K. Sabnani, *Multidestination Protocols for Satellite Broadcast Channels.* PhD thesis, Columbia University, 1982.

[14] D. G. Luenberger, *Linear and NonLinear Programming.* Addison-Wesley, 2nd edition ed., 1984.

[15] S. Y. Cheung and M. H. Ammar, "Using destination set grouping to improve the performance of window-controlled multipoint connections," College of Computing, Georgia Institute of Technology, Technical Report GIT-CC-94-32, August 1994.