Reliable Multicast Transport (RMT) Working Group
Internet-Draft
Expires: 17 March 2006

B. Adamson
NRL
C. Bormann
Universitaet Bremen TZI
M. Handley
UCL
J. Macker
NRL
October 2005

Negative-Acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Building Blocks draft-ietf-rmt-bb-norm-revised-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on March 17, 2006.

Copyright Notice

Abstract

This document discusses the creation of negative-acknowledgment (NACK)-oriented reliable multicast (NORM) protocols. The rationale for NORM goals and assumptions are presented. Technical challenges for NACK-oriented (and in some cases general) reliable multicast protocol operation are identified. These goals and challenges are resolved into a set of functional "building blocks" that address different aspects of NORM protocol operation. It is anticipated that these building blocks will be useful in generating different instantiations of reliable multicast protocols.

Table of Contents

1. Introduction

Reliable multicast transport is a desirable technology for the efficient and reliable distribution of data to a group on the Internet. The complexities of group communication paradigms necessitate different protocol types and instantiations to meet the range of performance and scalability requirements of different potential reliable multicast applications and users [3]. This document addresses the creation of negative-acknowledgment (NACK)-oriented reliable multicast (NORM) protocols. While different protocol instantiations may be required to meet specific application and network architecture demands [4], there are a number of fundamental components that may be common to these different instantiations. This document describes the framework and common "building block" components relevant to multicast protocols based primarily on NACK operation for reliable transport. While this document discusses a large set of reliable multicast components and issues relevant to NORM protocol design, it specifically addresses in detail the following building blocks which are not addressed in other IETF documents:

1) NORM sender transmission strategies,
2) NACK-oriented repair process with timer-based feedback suppression, and
3) Round-trip timing for adapting NORM timers.

The potential relationships to other reliable multicast transport building blocks (Forward Error Correction (FEC), congestion control) and general issues with NORM protocols are also discussed. This document is a product of the IETF RMT WG and follows the guidelines provided in RFC 3269 [5]. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [1].

Statement of Intent

This memo contains part of the definitions necessary to fully specify a Reliable Multicast Transport protocol in accordance with RFC 2357. As per RFC 2357, the use of any reliable multicast protocol in the Internet requiresan adequate congestion control scheme.

While waiting for such a scheme to be available, or for an existing scheme to be proven adequate, the Reliable Multicast Transport working group (RMT) publishes this Request for Comments in the "Experimental" category.

It is the intent of RMT to re-submit this specification as an IETF Proposed Standard as soon as the above condition is met.

2. Rationale

Each potential protocol instantiation using the building blocks presented here (and in other applicable building block documents) will have specific criteria that may influence individual protocol design. To support the development of applicable building blocks, it is useful to identify and summarize driving general protocol design goals and assumptions. These are areas that each protocol instantiation will need to address in detail. Each building block description in this document will include a discussion of the impact of these design criteria. The categories of design criteria considered here include:

1) Delivery Service Model,
2) Group Membership Dynamics,

3) Sender/receiver relationships,
4) Group Size Scalability,
5) Data Delivery Performance,
6) Network Environments, and
7) Router/Intermediate System Interactions.

All of these areas are at least briefly discussed. Additionally, other reliable multicast transport building block documents such as [9] have been created to address areas outside of the scope of this document. NORM protocol instantiations may depend upon these other building blocks as well as the ones presented here. This document focuses on areas that are unique to NORM but may be used in concert with the other building block areas. In some cases, a building block may be able address a wide range of assumptions, while in other cases there will be trade-offs required to meet different application needs or operating environments. Where necessary, building block features are designed to be parametric to meet different requirements. Of course, an underlying goal will be to minimize design complexity and to at least recommend default values for any such parameters that meet a general purpose "bulk data transfer" requirement in a typical Internet environment.

2.1. Delivery Service Model

The implicit goal of a reliable multicast transport protocol is the reliable delivery of data among a group of members communicating using IP multicast datagram service. However, the specific service the application is attempting to provide can impact design decisions. A most basic service model for reliable multicast transport is that of "bulk transfer" which is a primary focus of this and other related RMT working group documents. However, the same principles in protocol design may also be applied to other services models, e.g., more interactive exchanges of small messages such as with white-boarding or text chat. Within these different models there are issues such as the sender's ability to cache transmitted data (or state referencing it) for retransmission or repair. The needs for ordering and/or causality in the sequence of transmissions and receptions among members in the group may be different depending upon data content. The group communication paradigm differs significantly from the point-to-point model in that, depending upon the data content type, some receivers may complete reception of a portion of data content and be able to act upon it before other members have received the content. This may be acceptable (or even desirable) for some applications but not for others. These varying requirements drive the need for a number of different protocol instantiation designs. A significant challenge in developing generally useful building block mechanisms is accommodating even a limited range of these capabilities without defining specific application-level details.

2.2. Group Membership Dynamics

One area where group communication can differ from point-to-point communications is that even if the composition of the group changes, the "thread" of communication can still exist. This contrasts with the point-to-point communication model where, if either of the two parties leave, the communication process (exchange of data) is terminated (or at least paused). Depending upon application goals, senders and receivers participating in a reliable multicast transport "session" may be able to join late, leave, and/or potentially rejoin while the ongoing group communication "thread" still remains functional and useful. Also note that this can impact protocol message content. If "late joiners" are supported, some amount of additional information may be placed in message headers to accommodate this functionality. Alternatively, the information may be sent in its own message (on demand or intermittently) if the impact to the overhead of typical message transmissions is deemed too great. Group dynamics can also impact other protocol mechanisms such as NACK timing, congestion control operation, etc.

2.3. Sender/Receiver Relationships

The relationship of senders and receivers among group members requires consideration. In some applications, there may be a single sender multicasting to a group of receivers. In other cases, there may be more than one sender or the potential for everyone in the group to be a sender _and_ receiver of data may exist.

2.4. Group Size Scalability

Native IP multicast [2] may scale to extremely large group sizes. It may be desirable for some applications to scale along with the multicast infrastructure's ability to scale. In its simplest form, there are limits to the group size to which a NACK-oriented protocol can apply without NACK implosion problems. Research suggests that NORM group sizes on the order of tens of thousands of receivers may operate with modest feedback to the sender using probabilistic, timer-based suppression techniques [7]. However, the potential for router assistance and/or other NACK suppression heuristics may enable these protocols to scale to very large group sizes. In large scale cases, it may be prohibitive for members to maintain state on all other members (in particular, other receivers) in the group. The impact of group size needs to be considered in the development of applicable building blocks.

2.5. Data Delivery Performance

There is a trade-off between scalability and data delivery latency when designing NACK-oriented protocols. If probabilistic, timer-based NACK suppression is to be used, there will be some delays built into the NACK process to allow suppression to occur and for the sender of data to identify appropriate content for efficient repair transmission. For example, backoff timeouts can be used to ensure efficient NACK suppression and repair transmission, but this comes at a cost of increased delivery latency and increased buffering requirements for both senders and receivers. The building blocks SHOULD allow applications to establish bounds for data delivery performance. Note that application designers must be aware of the scalability trade-off that is made when such bounds are applied.

2.6. Network Environments

The Internet Protocol has historically assumed a role of providing service across heterogeneous network topologies. It is desirable that a reliable multicast protocol be capable of effectively operating across a wide range of the networks to which general purpose IP service applies. The bandwidth available on the links between the members of a single group today may vary between low numbers of kbit/s for wireless links and multiple Gbit/s for high speed LAN connections, with varying degrees of contention from other flows. Recently, a number of asymmetric network services including 56K/ADSL modems, CATV Internet service, satellite and other wireless communication services have begun to proliferate. Many of these are inherently broadcast media with potentially large "fan-out" to which IP multicast service is highly applicable. Additionally, policy and/or technical issues may result in topologies where multicast connectivity is limited to a single source multicast (SSM) model from a specific source [8]. Receivers in the group may be restricted to unicast feedback for NACKs and other messages. Consideration must be given, in building block development and protocol design, to the nature of the underlying networks.

2.7. Router/Intermediate System Assistance

While intermediate assistance from devices/systems with direct knowledge of the underlying network topology may be used to leverage the performance and scalability of reliable multicast protocols, there will continue to be a number of instances where this is not available or practical. Any building block components for NACK-oriented reliable multicast SHALL be capable of operating without such assistance. However, it is RECOMMENDED that such protocols also consider utilizing these features when available.

3. Functionality

The previous section has presented the role of protocol building blocks and some of the criteria that may affect NORM building block identification/design. This section describes different building block areas applicable to NORM protocols. Some of these areas are specific to NACK-oriented protocols. Detailed descriptions of such areas are provided. In other cases, the areas (e.g., node identifiers, forward error correction (FEC), etc.) may be applicable to other forms of reliable multicast. In those cases, the discussion below describes requirements placed on those other general building block areas from the standpoint of NACK-oriented reliable multicast. Where applicable, other building block documents are referenced for possible contribution to NORM protocols.

For each building block, a notional "interface description" is provided to illustrate any dependencies of one building block component upon another or upon other protocol parameters. A building block component may require some form of "input" from another building block component or other source to perform its function. Any "inputs" required by a building block component and/or any resultant "output" provided will be defined and described in each building block component's interface description. Note that the set of building blocks presented here do not fully satisfy each other's "input" and "output" needs. In some cases, "inputs" for the building blocks here must come from other building blocks external to this document (e.g., congestion control or FEC). In other cases NORM building block "inputs" must be satisfied by the specific protocol instantiation or implementation (e.g., application data and control).

The following building block components relevant to NORM are identified:

(NORM-Specific)

1) NORM Sender Transmission
2) NORM Repair Process
3) NORM Receiver Join Policies

(General Purpose)

4) Node (member) Identification
5) Data Content Identification
6) Forward Error Correction (FEC)
7) Round-trip Timing Collection
8) Group Size Determination/Estimation
9) Congestion Control Operation
10) Router/Intermediate System Assistance
11) Ancillary Protocol Mechanisms

Figure 1 provides a pictorial overview of these building block areas and some of their relationships. For example, the content of the data messages that a sender initially transmits depends upon the "Node Identification", "Data Content Identification", and "FEC" components while the rate of message transmission will generally depend upon the "Congestion Control" component. Subsequently, the receivers' response to these transmissions (e.g., NACKing for repair) will depend upon the data message content and inputs from other building block components. Finally, the sender's processing of receiver responses will feed back into its transmission strategy.

The components on the left side of this figure are areas that may be applicable beyond NORM. The most significant of these components are discussed in other building block documents such as [9]. A brief description of these areas and their role in the NORM protocol is given below. The components on the right are seen as specific to NORM protocols, most notably the NACK repair process. These areas are discussed in detail below. Some other components (e.g., "Security") impact many aspects of the protocol, and others such as "Router Assistance" may be more transparent to the core protocol processing. The sections below describe the "NORM Sender Transmission", "NORM Repair Process", and "RTT Collection" building blocks in detail. The relationships to and among the other building block areas are also discussed, focusing on issues applicable to NORM protocol design. Where applicable, specific technical recommendations are made for mechanisms that will properly satisfy the goals of NORM transport for the Internet.

```
                                    Application Data and Control
                                                |
                                                V
   .---------------------.            .-----------------------.
   | Node Identification |----------->|  Sender Transmission  |<----.
   `---------------------'       _.-' `-----------------------'     |
   .---------------------.   _.-' .'           | .--------------.   |
   | Data Identification |--'   .''            | |  Join Policy |   |
   `---------------------'    .' '             V `--------------'   |
   .---------------------.  .'  '     .----------------------.      |
.->| Congestion Control  |-'   '      | Receiver NACK        |      |
|  `---------------------'   .'       | Repair Process       |      |
|  .---------------------. .'         | .------------------. |      |
|  |        FEC          |'.          | | NACK Initiation  | |      |
|  `---------------------'` `._       | `------------------' |      |
|  .---------------------. ``. `-._   | .------------------. |      |
`--|    RTT Collection   |._` `    `->| | NACK Content     | |      |
   `---------------------'` `` `      | `------------------' |      |
   .---------------------.  ` ``-`._  | .------------------. |      |
   |    Group Size Est.  |---`-`---`->| | NACK Suppression | |      |
   `---------------------'`.  ` `     | `------------------' |      |
   .---------------------.  `  ` `    `----------------------'      |
   |       Other         |   `  ` `           | +-----------------+ |
   `---------------------'    `  ` `          | |Router Assistance| |
                               `. ` `         V +-----------------+ |
                                 `-` >.-------------------------.   |
                                      | Sender NACK Processing  |___/
                                      | and Repair Response     |
                                      `-------------------------'
                   ^                         ^
                   |                         |
                 .-----------------------------.
                 |         (Security)          |
                 `-----------------------------'
```

Fig. 1 - NORM Building Block Framework

3.1. NORM Sender Transmission

NORM senders will transmit data content to the multicast session. The data content will be application dependent. The sender will transmit data content at a rate, and with message sizes, determined by application and/or network architecture requirements. Any FEC encoding of sender transmissions SHOULD conform with the guidelines of [9]. When congestion control mechanisms are needed (REQUIRED for general Internet operation), the NORM transmission rate SHALL be controlled by the congestion control mechanism. In any case, it is RECOMMENDED that all data transmissions from NORM senders be subject to rate limitations determined by the application or congestion control algorithm. The sender's transmissions SHOULD make good utilization of the available capacity (which may be limited by the application and/or by congestion control). As a result, it is expected there will be overlap and multiplexing of new data content transmission with repair content. Other factors related to application operation may determine sender transmission formats and methods. For example, some consideration needs to be given to the sender's behavior during intermittent idle periods when it has no data to transmit.

In addition to data content, other sender messages or commands may be employed as part of protocol operation. These messages may occur outside of the scope of application data transfer. In NORM protocols, reliability of such protocol messages may be attempted by redundant transmission when positive acknowledgement is prohibitive due to group size scalability concerns. Note that protocol design SHOULD provide mechanisms for dealing with cases where such messages are not received by the group. As an example, a command message might be redundantly transmitted by a sender to indicate that it is temporarily (or permanently) halting transmission. At this time, it may be appropriate for receivers to respond with NACKs for any outstanding repairs they require following the rules of the NORM NACK procedure. For efficiency, the sender should allow sufficient time between the redundant transmissions to receive any NACK-oriented responses from the receivers to this command.

In general, when there is any resultant NACK or other feedback operation, the timing of redundant transmission of control messages issued by a sender and other NORM protocol timeouts should be dependent upon the group greatest round trip timing (`GRTT`) estimate and any expected resultant NACK or other feedback operation. The NORM `GRTT` is an estimate of the worst-case round-trip timing from a sender to any receivers in the group. It is assumed that the `GRTT` interval is a conservative estimate of the maximum span (with respect to delay) of the multicast group across a network topology with respect to given sender. NORM instantiations SHOULD be able to dynamically adapt to a wide range of multicast network topologies.

Sender Transmission Interface Description

Inputs:

1) Application data and control
2) Sender node identifier
3) Data identifiers
4) Segmentation and FEC parameters
5) Transmission rate
6) Application controls
7) Receiver feedback messages (e.g., NACKs)

Outputs:

1) Controlled transmission of messages with headers uniquely identifying data or repair content within the context of the NORM session.
2) Commands indicating sender's status or other transport control actions to be taken.

3.2. NORM Repair Process

A critical component of NORM protocols is the NACK repair process. This includes the receiver's role in detecting and requesting repair needs, and the sender's response to such requests. There are four primary elements of the NORM repair process:

1) Receiver NACK process initiation,
3) NACK suppression,
2) NACK message content,

4) Sender NACK processing and response.

3.2.1. Receiver NACK Process Initiation

The NORM NACK process (cycle) will be initiated by receivers that detect a need for repair transmissions from a specific sender to achieve reliable reception. When FEC is applied, a receiver should initiate the NACK process only when it is known its repair requirements exceed the amount of pending FEC transmission for a given coding block of data content. This can be determined at the end of the current transmission block (if it is indicated) or upon the start of reception of a subsequent coding block or transmission object. This implies the NORM data content is marked to identify its FEC block number and that ordinal relationship is preserved in order of transmission.

Alternatively, if the sender's transmission advertises the quantity of repair packets it is already planning to send for a block, the receiver may be able to initiate the NACK processor earlier. Allowing receivers to initiate NACK cycles at any time they detect their repair needs have exceeded pending repair transmissions may result in slightly quicker repair cycles. However, it may be useful to limit NACK process initiation to specific events such as at the end-of-transmission of an FEC coding block or upon detection of subsequent coding blocks. This can allow receivers to aggregate NACK content into a smaller number of NACK messages and provide some implicit loose synchronization among the receiver set to help facilitate effective probabilistic suppression of NACK feedback. The receiver MUST maintain a history of data content received from the sender to determine its current repair needs. When FEC is employed, it is expected that the history will correspond to a record of pending or partially-received coding blocks.

For probabilistic, timer-base suppression of feedback, the NACK cycle should begin with receivers observing backoff timeouts. In conjunction with initiating this backoff timeout, it is important that the receivers record the current position in the sender's transmission sequence at which they initiate the NACK cycle. When the suppression backoff timeout expires, the receivers should only consider their repair needs up to this recorded transmission position in making the decision to transmit or suppress a NACK. Without this restriction, suppression is greatly reduced as additional content is received from the sender during the time a NACK message propagates across the network to the sender and other receivers.

Receiver NACK Process Initiation Interface Description

Inputs:

1) Sender data content with sequencing identifiers from sender transmissions.
2) History of content received from sender.

Outputs:

1) NACK process initiation decision
2) Recorded sender transmission sequence position.

3.2.2. NACK Suppression

An effective NORM feedback suppression mechanism is the use of random backoff timeouts prior to NACK transmission by receivers requiring repairs [10]. Upon expiration of the backoff timeout, a receiver will request repairs unless its pending repair needs have been completely superseded by NACK messages heard from other receivers (when receivers are multicasting NACKs) or from some indicator from the sender. When receivers are unicasting NACK messages, the sender may facilitate NACK suppression by forwarding a representation of NACK content it has received to the group at large or provide some other indicator of the repair information it will be subsequently transmitting.

For effective and scalable suppression performance, the backoff timeout periods used by receivers should be independently, randomly picked by receivers with a truncated exponential distribution [6]. This results in the majority of the receiver set holding off transmission of NACK messages under the assumption that the smaller number of "early NACKers" will supersede the repair needs of the remainder of the group. The mean of the distribution should be determined as a function of the current estimate of sender<->group `GRTT` and a group size estimate that is determined by other mechanisms within the protocol or preset by the multicast application.

A simple algorithm can be constructed to generate random backoff timeouts with the appropriate distribution. Additionally, the algorithm may be designed to optimize the backoff distribution given the number of receivers (R) potentially generating feedback. This "optimization" minimizes the number of feedback messages (e.g., NACK) in the worst-case situation where all receivers generate a NACK. The maximum backoff timeout (`T_maxBackoff`) can be set to control reliable delivery latency versus volume of feedback traffic. A larger value of `T_maxBackoff` will result in a lower density of feedback traffic for a given repair cycle. A smaller value of `T_maxBackoff` results in shorter latency which also reduces the buffering requirements of senders and receivers for reliable transport.

Given the receiver group size (`R`), and maximum allowed backoff timeout (`T_maxBackoff`), random backoff timeouts (`t'`) with a truncated exponential distribution can be picked with the following algorithm:

1) Establish an optimal mean (`L`) for the exponential backoff based on the group size:

```
L = ln(R) + 1
```

2) Pick a random number (`x`) from a uniform distribution over a range of:

```
         L                               L                      L
--------------------  to   --------------------  +  ----------
T_maxBackoff*(exp(L)-1)     T_maxBackoff*(exp(L)-1)   T_maxBackoff
```

3) Transform this random variate to generate the desired random backoff time (`t'`) with the following equation:

```
t' = T_maxBackoff/L * ln(x * (exp(L) - 1) * (T_maxBackoff/L))
```

This C language function can be used to generate an appropriate random backoff time interval:

```
double RandomBackoff(double maxTime, double groupSize)
{
     double lambda = log(groupSize) + 1;
     double x = UniformRand(lambda/maxTime) +
                lambda / (maxTime*(exp(lambda)-1));
     return ((maxTime/lambda) *
            log(x*(exp(lambda)-1)*(maxTime/lambda)));
}  // end RandomBackoff()
```

where `UniformRand(double max)` returns random numbers with a uniform distribution from the range of `0..max`. For example, based on the POSIX "`rand()`" function, the following C code can be used:

```
double UniformRand(double max)
{
     return (max * ((double)rand()/(double)RAND_MAX));
}
```

The number of expected NACK messages generated (`N`) within the first round trip time for a single feedback event is approximately:

```
N = exp(1.2 * L / (2*T_maxBackoff/GRTT))
```

Thus the maximum backoff time can be adjusted to tradeoff worst-case NACK feedback volume versus latency. This is derived from [6] and assumes `T_maxBackoff >= GRTT`, and `L` is the mean of the distribution optimized for the given group size as shown in the algorithm above. Note that other mechanisms within the protocol may work to reduce redundant NACK generation further. It is suggested that `T_maxBackoff` be selected as an integer multiple of the sender's current advertised `GRTT` estimate such that:

`T_maxBackoff = K * GRTT` ;where `K >= 1`

For general Internet operation, a default value of `K=4` is RECOMMENDED for operation with multicast (to the group at large) NACK delivery and a value of `K=6` for unicast NACK delivery. Alternate values may be used to for buffer utilization, reliable delivery latency and group size scalability tradeoffs.

Given that (`K*GRTT`) is the maximum backoff time used by the receivers to initiate NACK transmission, other timeout periods related to the NACK repair process can be scaled accordingly. One of those timeouts is the amount of time a receiver should wait after generating a NACK message before allowing itself to initiate another NACK backoff/transmission cycle (`T_rcvrHoldoff`). This delay should be sufficient for the sender to respond to the received NACK with repair messages. An appropriate value depends upon the amount of time for the NACK to reach the sender and the sender to provide a repair response. This MUST include any amount of sender NACK aggregation period during which possible multiple NACKs are accumulated to determine an efficient repair response. These timeouts are further discussed in the section below on "Sender NACK Processing and Repair Response".

There are also secondary measures that can be applied to improve the performance of feedback suppression. For example, the sender's data content transmissions can follow an ordinal sequence of transmission. When repairs for data content occur, the receiver can note that the sender has "rewound" its data content transmission position by observing the data object, FEC block number, and FEC symbol identifiers. Receivers SHOULD limit transmission of NACKs to only when the sender's current transmission position exceeds the point to which the receiver has incomplete reception. This reduces premature requests for repair of data the sender may be planning to provide in response to other receiver requests. This mechanism can be very effective for protocol convergence in high loss conditions when transmissions of NACKs from other receivers (or indicators from the sender) are lost. Another mechanism (particularly applicable when FEC is used) is for the sender to embed an indication of impending repair transmissions in current packets sent. For example, the indication may be as simple as an advertisement of the number of FEC packets to be sent for the current applicable coding block.

Finally, some consideration might be given to using the NACKing history of receivers to weight their selection of NACK backoff timeout intervals. For example, if a receiver has historically been experiencing the greatest degree of loss, it may promote itself to statistically NACK sooner than other receivers. Note this requires there is correlation over successive intervals of time in the loss experienced by a receiver. Such correlation MAY not always be present in multicast networks. This adjustment of backoff timeout selection may require the creation of an "early NACK" slot for these historical NACKers. This additional slot in the NACK backoff window will result in a longer repair cycle process that may not be desirable for some applications. The resolution of these trade-offs may be dependent upon the protocol's target application set or network.

After the random backoff timeout has expired, the receiver will make a decision on whether to generate a NACK repair request or not (i.e., it has been suppressed). The NACK will be suppressed when any of the following conditions has occurred:

1) The accumulated state of NACKs heard from other receivers (or forwarding of this state by the sender) is equal to or supersedes the repair needs of the local receiver. Note that the local receiver should consider its repair needs only up to the sender transmission position recorded at the NACK cycle initiation (when the backoff timer was activated).

2) The sender's data content transmission position "rewinds" to a point ordinally less than that of the lowest sequence position of the local receiver's repair needs. (This detection of sender "rewind" indicates the sender has already responded to other receiver repair needs of which the local receiver may not have been aware). This "rewind" event can occur any time between 1) when the NACK cycle was initiated with the backoff timeout activation and 2) the current moment when the backoff timeout has expired to suppress the NACK. Another NACK cycle must be initiated by the receiver when the sender's transmission sequence position exceeds the receiver's lowest ordinal repair point. Note it is possible that the local receiver may have had its repair needs satisfied as a result of the sender's response to the repair needs of other receivers and no further NACKing is required.

If these conditions have not occurred and the receiver still has pending repair needs, a NACK message is generated and transmitted. The NACK should consist of an accumulation of repair needs from the receiver's lowest ordinal repair point up to the current sender transmission sequence position. A single NACK message should be generated and the NACK message content should be truncated if it exceeds the payload size of single protocol message. When such NACK payload limits occur, the NACK content SHOULD contain requests for the ordinally lowest repair content needed from the sender.

NACK Suppression Interface Description

Inputs:

1) NACK process initiation decision.
2) Recorded sender transmission sequence position.
3) Sender GRTT.
4) Sender group size estimate.
5) Application-defined bound on backoff timeout period.
6) NACKs from other receivers.
7) Pending repair indication from sender (may be forwarded NACKs).
8) Current sender transmission sequence position.

Outputs:

1) Yes/no decision to generate NACK message upon backoff timer expiration.

3.2.3. NACK Content

The content of NACK messages generated by reliable multicast receivers will include information detailing their current repair needs. The specific information depends on the use and type of FEC in the NORM repair process. The identification of repair needs is dependent upon the data content identification (See Section 3.5 below). At the highest level the NACK content will identify the sender to which the NACK is addressed and the data transport object (or stream) within the sender's transmission that needs repair. For the indicated transport entity, the NACK content will then identify the specific FEC coding blocks and/or symbols it requires to reconstruct the complete transmitted data. This content may consist of FEC block erasure counts and/or explicit indication of missing blocks or symbols (segments) of data and FEC content. It should also be noted that NORM can be effectively instantiated without a requirement for reliable NACK delivery using the techniques discussed here.

3.2.3.1. NACK and FEC Repair Strategies

Where FEC-based repair is used, the NACK message content will minimally need to identify the coding block(s) for which repair is needed and a count of erasures (missing packets) for the coding block. An exact count of erasures implies the FEC algorithm is capable of repairing _any_ loss combination within the coding block. This count may need to be adjusted for some FEC algorithms. Considering that multiple repair rounds may be required to successfully complete repair, an erasure count also implies that the quantity of unique FEC parity packets the server has available to transmit is essentially unlimited (i.e., the server will always be able to provide new, unique, previously unsent parity packets in response to any subsequent repair requests for the same coding block). Alternatively, the sender may "round-robin" transmit through its available set of FEC symbols for a given coding block, and eventually affect repair. For a most efficient repair strategy, the NACK content will need to also _explicitly_ identify which symbols (information and/or parity) the receiver requires to successfully reconstruct the content of the coding block. This will be particularly true of small to medium size block FEC codes (e.g., Reed Solomon) that are capable of provided a limited number of parity symbols per FEC coding block.

When FEC is not used as part of the repair process, or the protocol instantiation is required to provide reliability even when the sender has transmitted all available parity for a given coding block (or the sender's ability to buffer transmission history is exceeded by the `delay*bandwidth*loss` characteristics of the network topology), the NACK content will need to contain _explicit_ coding block and/or segment loss information so that the sender can provide appropriate repair packets and/or data retransmissions. Explicit loss information in NACK content may also potentially serve other purposes. For example, it may be useful for decorrelating loss characteristics among a group of receivers to help differentiate candidate congestion control bottlenecks among the receiver set.

When FEC is used and NACK content is designed to contain explicit repair requests, there is a strategy where the receivers can NACK for specific content that will help facilitate NACK suppression and repair efficiency. The assumptions for this strategy are that sender may potentially exhaust its supply of new, unique parity packets available for a given coding block and be required to explicitly retransmit some data or parity symbols to complete reliable transfer. Another assumption is that an FEC algorithm where any parity packet can fill any erasure within the coding block (e.g., Reed Solomon) is used. The goal of this strategy is to make maximum use of the available parity and provide the minimal amount of data and repair transmissions during reliable transfer of data content to the group.

When systematic FEC codes are used, the sender transmits the data content of the coding block (and optionally some quantity of parity packets) in its initial transmission. Note that a systematic FEC coding block is considered to be logically made up of the contiguous set of source data vectors plus parity vectors for the given FEC algorithm used. For example, a systematic coding scheme that provides for 64 data symbols and 32 parity symbols per coding block would contain FEC symbol identifiers in the range of 0 to 95.

Receivers then can construct NACK messages requesting sufficient content to satisfy their repair needs. For example, if the receiver has three erasures in a given received coding block, it will request transmission of the three lowest ordinal parity vectors in the coding block. In our example coding scheme from the previous paragraph, the receiver would explicitly request parity symbols 64 to 66 to fill its three erasures for the coding block. Note that if the receiver's loss for the coding block exceeds the available parity quantity (i.e., greater than 32 missing symbols in our example), the receiver will be required to construct a NACK requesting all (32) of the available parity symbols plus some additional portions of its missing data symbols in order to reconstruct the block. If this is done consistently across the receiver group, the resulting NACKs will comprise a minimal set of sender transmissions to satisfy their repair needs.

In summary, the rule is to request the lower ordinal portion of the parity content for the FEC coding block to satisfy the erasure repair needs on the first NACK cycle. If the available number of parity symbols is insufficient, the receiver will also request the subset of ordinally highest missing data symbols to cover what the parity symbols will not fill. Note this strategy assumes FEC codes such as Reed-Solomon for which a single parity symbol can repair any erased symbol. This strategy would need minor modification to take into account the possibly limited repair capability of other FEC types. On subsequent NACK repair cycles where the receiver may have received some portion of its previously requested repair content, the receiver will use the same strategy, but only NACK for the set of parity and/or data symbols it has not yet received. Optionally, the receivers could also provide a count of erasures as a convenience to the sender or intermediate systems assisting NACK operation.

Other types of FEC schemes may require alteration to the NACK and repair strategy described here. For example, some of the large block or expandable FEC codes described in [14] may be less deterministic with respect to defining optimal repair requests by receivers or repair transmission strategies by senders. For these types of codes, it may be sufficient for receivers to NACK with an estimate of the quantity of additional FEC symbols required to complete reliable reception and for the sender to respond accordingly. This apparent disadvantage as compared to codes such as Reed Solomon may be offset by reduced computational requirements and/or ability to support large coding blocks for increased repair efficiency that these codes can offer.

After receipt and accumulation of NACK messages during the aggregation period, the sender can begin transmission of fresh (previously untransmitted) parity symbols for the coding block based on the highest receiver erasure count _if_ it has a sufficient quantity of parity symbols that were _not_ previously transmitted. Otherwise, the sender MUST resort to transmitting the explicit set of repair vectors requested. With this approach, the sender needs to maintain very little state on requests it has received from the group without need for synchronization of repair requests from the group. Since all receivers use the same consistent algorithm to express their explicit repair needs, NACK suppression among receivers is simplified over the course of multiple repair cycles. The receivers can simply compare NACKs heard from other receivers against their own calculated repair needs to determine whether they should transmit or suppress their pending NACK messages.

3.2.3.2. NACK Content Format

The format of NACK content will depend on the protocol's data service model and the format of data content identification the protocol uses. This NACK format also depends upon the type of FEC encoding (if any) used. Figure 2 illustrates a logical, hierarchical transmission content identification scheme, denoting that the notion of objects (or streams) and/or FEC blocking is optional at the protocol instantiation's discretion. Note that the identification of objects is with respect to a given sender. It is recommended that transport data content identification is done within the context of a sender in a given session. Since the notion of session "streams" and "blocks" is optional, the framework degenerates to that of typical transport data segmentation and reassembly in its simplest form.

```
Session_
        \_
          Sender_
                 \_
                   [Object/Stream(s)]_
                                      \_
                                        [FEC Blocks]_
                                                    \_
                                                      Symbols
```

Fig. 2: NORM Data Content Identification Hierarchy

The format of NACK messages should meet the following goals:

1) Able to identify transport data unit transmissions required to repair a portion of the received content, whether it is an entire missing object/stream (or range), entire FEC coding block(s), or sets of symbols,

2) Be simple to process for NACK aggregation and suppression,

3) Be capable of including NACKs for multiple objects, FEC coding blocks and/or symbols in a single message, and

4) Have a reasonably compact format.

If the NORM transport object/stream is identified with an <*objectId*> and the FEC symbol being transmitted is identified with an <*fecPayloadId*>, the concatenation of <*objectId::fecPayloadId*> comprises a basic transport protocol data unit (TPDU) identifier for symbols from a given source. NACK content can be composed of lists and/or ranges of these TPDU identifiers to build up NACK messages to describe the receivers repair needs. If no hierarchical object delineation or FEC blocking is used, the TPDU is a simple linear representation of the data symbols transmitted by the sender. When the TPDU represents a hierarchy for purposes of object/stream delineation and/or FEC blocking, the NACK content unit may require flags to indicate which portion of the TPDU is applicable. For example, if an entire "object" (or range of objects) is missing in the received data, the receiver will not necessarily know the appropriate range of <*sourceBlockNumbers*> or <*encodingSymbolIds*> for which to request repair and thus requires some mechanism to request repair (or retransmission) of the entire unit represented by an <*objectId*>. The same is true if entire FEC coding blocks represented by one or a range of <*sourceBlockNumbers*> have been lost.

NACK Content Interface Description

Inputs:

1) Sender identification.
2) Sender data identification.
3) Sender FEC Object Transmission Information.
4) Recorded sender transmission sequence position.
5) Current sender transmission sequence position. History of repair needs for this sender.

Outputs:

1) NACK message with repair requests.

3.2.4. Sender Repair Response

Upon reception of a repair request from a receiver in the group, the sender will initiate a repair response procedure. The sender may wish to delay transmission of repair content until it has had sufficient time to accumulate potentially multiple NACKs from the receiver set. This allows the sender to determine the most efficient repair strategy for a given transport stream/object or FEC coding block. Depending upon the approach used, some protocols may find it beneficial for the sender to provide an indicator of pending repair transmissions as part of its current transmitted message content. This can aid some NACK suppression mechanisms. The amount of time to perform this NACK aggregation should be sufficient to allow for the maximum receiver NACK backoff window ("`T_maxBackoff`" from Section 3.2.2) and propagation of NACK messages from the receivers to the sender. Note the maximum transmission delay of a message from a receiver to the sender may be approximately `(1*GRTT)` in the case of very asymmetric network topology with respect to transmission delay. Thus, if the maximum receiver NACK backoff time is `T_maxBackoff = K*GRTT`, the sender NACK aggregation period should be equal to at least:

```
T_sndrAggregate = T_maxBackoff + 1*GRTT = (K+1)*GRTT
```

Immediately after the sender NACK aggregation period, the sender will begin transmitting repair content determined from the aggregate NACK state and continue with any new transmission. Also, at this time, the sender should observe a "holdoff" period where it constrains itself from initiating a new NACK aggregation period to allow propagation of the new transmission sequence position due to the repair response to the receiver group. To allow for worst case asymmetry, this "holdoff" time should be:

```
T_sndrHoldoff = 1*GRTT
```

Recall that the receivers will also employ a "holdoff" timeout after generating a NACK message to allow time for the sender's response. Given a sender `<T_sndrAggregate>` plus `<T_sndrHoldoff>` time of `(K+1)*GRTT`, the receivers should use holdoff timeouts of:

```
T_rcvrHoldoff = T_sndrAggregate + T_sndrHoldoff = (K+2)*GRTT
```

This allows for a worst-case propagation time of the receiver's NACK to the sender, the sender's aggregation time and propagation of the sender's response back to the receiver. Additionally, in the case of unicast feedback from the receiver set, it may be useful for the sender to forward (via multicast) a representation of its aggregated NACK content to the group to allow for NACK suppression when there is not multicast connectivity among the receiver set.

At the expiration of the `<T_sndrAggregate>` timeout, the sender will begin transmitting repair messages according to the accumulated content of NACKs received. There are some guidelines with regards to FEC-based repair and the ordering of the repair response from the sender that can improve reliable multicast efficiency:

1) When FEC is used, it is beneficial that the sender transmit previously untransmitted parity content as repair messages whenever possible. This maximizes the receiving nodes' ability to reconstruct the entire transmitted content from their individual subsets of received messages.

2) The transmitted object and/or stream data and repair content should be indexed with monotonically increasing sequence numbers (within a reasonably large ordinal space). If the sender observes the discipline of transmitting repair for the earliest content (e.g., ordinally lowest FEC blocks) first, the receivers can use a strategy of withholding repair requests for later content until the sender once again returns to that point in the object/stream transmission sequence. This can increase overall message efficiency among the group and help work to keep repair cycles relatively synchronized without dependence upon strict time synchronization among the sender and receivers. This also helps minimize the buffering requirements of receivers and senders and reduces redundant transmission of data to the group at large.

Sender Repair Response Interface Description

Inputs:

1) Receiver NACK messages
2) Group timing information

Outputs

1) Repair messages (FEC and/or Data content retransmission)
2) Advertisement of current pending repair transmissions when unicast receiver feedback is detected.

3.3. NORM Receiver Join Policies and Procedures

Consideration should be given to the policies and procedures by which new receivers join a group (perhaps where reliable transmission is already in progress) and begin requesting repair. If receiver joins are unconstrained, the dynamics of group membership may impede the application's ability to meet its goals for forward progression of data transmission. Policies limiting the opportunities when receivers begin participating in the NACK process may be used to achieve the desired behavior. For example, it may be beneficial for receivers to attempt reliable reception from a newly-heard sender only upon non-repair transmissions of data in the first FEC block of an object or logical portion of a stream. The sender may also implement policies limiting the receivers from which it will accept NACK requests, but this may be prohibitive for scalability reasons in some situations. Alternatively, it may be desirable to have a looser transport synchronization policy and rely upon session management mechanisms to limit group dynamics that can cause poor performance, in some types of bulk transfer applications (or for potential interactive reliable multicast applications).

Group Join Policy Interface Description

Inputs:

1) Current object/stream data/repair content and sequencing identifiers from sender transmissions.

Outputs:

1) Receiver yes/no decision to begin receiving and NACKing for reliable reception of data

3.4. Reliable Multicast Member Identification

In a NORM protocol (or other multicast protocols) where there is the potential for multiple sources of data, it is necessary to provide some mechanism to uniquely identify the sources (and possibly some or all receivers in some cases) within the group. Identity based on arriving packet source addresses is insufficient for several reasons. These reasons include routing changes for hosts with multiple interfaces that result in different packet source addresses for a given host over time, network address translation (NAT) or firewall devices, or other transport/network bridging approaches. As a result, some type of unique source identifier <*sourceId*> field should be present in packets transmitted by reliable multicast session members.

3.5. Data Content Identification

The data and repair content transmitted by a NORM sender requires some form of identification in the protocol header fields. This identification is required to facilitate the reliable NACK-oriented repair process. These identifiers will also be used in NACK messages generated. This building block document assumes two very general types of data that may comprise bulk transfer session content. One type is static, discrete objects of finite size and the other is continuous non-finite streams. A given application may wish to reliably multicast data content using either one or both of these paradigms. While it may be possible for some applications to further generalize this model and provide mechanisms to encapsulate static objects as content embedded within a stream, there are advantages in many applications to provide distinct support for static bulk objects and messages with the context of a reliable multicast session. These applications may include content caching servers, file transfer, or collaborative tools with bulk content. Applications with requirements for these static object types can then take advantage of transport layer mechanisms (i.e., segmentation/reassembly, caching, integrated forward error correction coding, etc.) rather than being required to provide their own mechanisms for these functions at the application layer.

As noted, some applications may alternatively desire to transmit bulk content in the form of one or more streams of non-finite size. Example streams include continuous quasi-real-time message broadcasts (e.g., stock ticker) or some content types that are part of collaborative tools or other applications. And, as indicated above, some applications may wish to encapsulate other bulk content (e.g., files) into one or more streams within a multicast session.

The components described within this building block document are envisioned to be applicable to both of these models with the potential for a mix of both types within a single multicast session. To support this requirement, the normal data content identification should include a field to uniquely identify the object or stream <*objectId*> within some reasonable temporal or ordinal interval. Note that it is _not_ expected that this data content identification will be globally unique. It is expected that the object/stream identifier will be unique with respect to a given sender within the reliable multicast session and during the time that sender is supporting a specific transport instance of that object or stream.

Since the "bulk" object/stream content usually requires segmentation, some form of segment identification must also be provided. This segment identifier will be relative to any object or stream identifier that has been provided. Thus, in some cases, NORM protocol instantiations may be able to receive transmissions and request repair for multiple streams and one or more sets of static objects in parallel. For protocol instantiations employing FEC the segment identification portion of the data content identifier may consist of a logical concatenation of a coding block identifier <*sourceBlockNumber*> and an identifier for the specific data or parity symbol <*encodingSymbolId*> of the code block. The FEC Basic Schemes document [11] and descriptions of additional FEC schemes that may be documented later provide a standard message format for identifying FEC transmission content. NORM protocol instantiations using FEC SHOULD follow such guidelines.

Additionally, flags to determine the usage of the content identifier fields (e.g., stream vs. object) may be applicable. Flags may also serve other purposes in data content identification. It is expected that any flags defined will be dependent upon individual protocol instantiations.

In summary, the following data content identification fields may be required for NORM protocol data content messages:

1) Source node identifier (<*sourceId*>)
2) Object/Stream identifier (<*objectId*>), if applicable.
3) FEC Block identifier (<*sourceBlockNumber*>), if applicable.
4) FEC Symbol identifier (<*encodingSymbolId*>)
5) Flags to differentiate interpretation of identifier fields or identifier structure that implicitly indicates usage.
6) Additional FEC transmission content fields per FEC Building Block

These fields have been identified because any generated NACK messages will use these identifiers in requesting repair or retransmission of data. NORM protocols that use these data content fields should also be compatible with support for intermediate system assistance to reliable multicast transport operation when available.

3.6. Forward Error Correction (FEC)

Multiple forward error correction (FEC) approaches have been identified that can provide great performance enhancements to the repair process of NACK-oriented and other reliable multicast protocols [12], [13], [14]. NORM protocols can reap additional benefits since FEC-based repair does not _generally_ require explicit knowledge of repair content within the bounds of its coding block size (in symbols). In NORM, parity repair packets generated will generally be transmitted only in response to NACK repair requests from receiving nodes. However, there are benefits in some network environments for transmitting some predetermined quantity of FEC repair packets multiplexed with the regular data symbol transmissions [15]. This can reduce the amount of NACK traffic generated with relatively little overhead cost when group sizes are very large or the network connectivity has a large `delay*bandwidth` product with some nominal level of expected packet loss. While the application of FEC is not unique to NORM, these sorts of requirements may dictate the types of algorithms and protocol approaches that are applicable.

A specific issue related to the use of FEC with NORM is the mechanism used to identify the portion(s) of transmitted data content to which specific FEC packets are applicable. It is expected that FEC algorithms will be based on generating a set of parity repair packets for a corresponding block of transmitted data packets. Since data content packets are uniquely identified by the concatenation of <*sourceId::objectId::sourceBlockNumber::encodingSymbolId*> during transport, it is expected that FEC packets will be identified in a similar manner. The FEC Building Block document [9] provides detailed recommendations concerning application of FEC and standard formats for related reliable multicast protocol messages.

3.7. Round-trip Timing Collection

The measurement of packet propagation round-trip time (RTT) among members of the group is required to support timer-based NACK suppression algorithms, timing of sender commands or certain repair functions, and congestion control operation. The nature of the round-trip information collected is dependent upon the type of interaction among the members of the group. In the case where only "one-to-many" transmission is required, it may be that only the sender require RTT knowledge of the greatest RTT (GRTT) among the receiver set and/or RTT knowledge of only a portion of the group. Here, the GRTT information might be collected in a reasonably scalable manner. For congestion control operation, it is possible that RTT information may be required by each receiver in the group.

In this case, an alternative RTT collection scheme may be utilized where receivers collect individual RTT measurements with respect to the sender and advertise them to the group or sender. Where it is likely that exchange of reliable multicast data will occur among the group on a "many-to-many" basis, there are alternative measurement techniques that might be employed for increased efficiency [16]. And in some cases, there might be absolute time synchronization among hosts that may simplify RTT measurement. There are trade-offs in multicast congestion control design that require further consideration before a universal recommendation on RTT (or GRTT) measurement can be specified. Regardless of how the RTT information is collected (and more specifically GRTT) with respect to congestion control or other requirements, the sender will need to advertise its current GRTT estimate to the group for various timeouts used by receivers.

3.7.1. One-to-Many Sender GRTT Measurement

The goal of this form of RTT measurement is for the sender to learn the GRTT among the receivers who are actively participating in NORM operation. The set of receivers participating in this process may be the entire group or some subset of the group determined from another mechanism within the protocol instantiation. An approach to collect this GRTT information follows.

The sender periodically polls the group with a message (independent or "piggy-backed" with other transmissions) containing a `<sendTime>` timestamp relative to an internal clock at the sender. Upon reception of this message, the receivers will record this `<sendTime>` timestamp and the time (referenced to their own clocks) at which it was received `<recvTime>`. When the receiver provides feedback to the sender (either explicitly or as part of other feedback messages depending upon protocol instantiation specification), it will construct a "response" using the formula:

```
grttResponse = sendTime + (currentTime - recvTime)
```

where the `<sendTime>` is the timestamp from the last probe message received from the source and the (`<currentTime> - <recvTime>`) is the amount of time differential since that request was received until the receiver generated the response.

The sender processes each receiver response by calculating a current RTT measurement for the receiver from whom the response was received using the following formula:

```
RTT_rcvr = currentTime - grttResponse
```

During the each periodic GRTT probing interval, the source keeps the peak round trip timing measurement (`RTT_peak`) from the set of responses it has received. A conservative estimate of `GRTT` is kept to maximize the efficiency of redundant NACK suppression and repair aggregation. The update to the source's ongoing estimate of `GRTT` is done observing the following rules:

1) If a receiver's response round trip time (`RTT_rcvr`) is greater than the current `GRTT` estimate, the `GRTT` is immediately updated to this new peak value:

```
GRTT = RTT_rcvr
```

2) At the end of the response collection period (i.e., the GRTT probe interval), if the recorded "peak" response RTT_peak) is less than the current GRTT estimate, the GRTT is updated to:

```
GRTT = MAX(0.9*GRTT, RTT_peak)
```

3) If no feedback is received, the sender GRTT estimate remains unchanged.

4) At the end of the response collection period, the peak tracking value (RTT_peak) is reset to ZERO for subsequent peak detection.

The GRTT collection period (i.e., period of probe transmission) could be fixed at a value on the order of that expected for group membership and/or network topology dynamics. For robustness, more rapid probing could be used at protocol startup before settling to a less frequent, steady-state interval. Optionally, an algorithm may be developed to adjust the GRTT collection period dynamically in response to the current GRTT estimate (or variations in it) and to an estimation of packet loss. The overhead of probing messages could then be reduced when the GRTT estimate is stable and unchanging, but be adjusted to track more dynamically during periods of variation with correspondingly shorter GRTT collection periods. GRTT collection may also be coupled with collection of other information for congestion control purposes.

In summary, although NORM repair cycle timeouts are based on GRTT, it should be noted that convergent operation of the protocol does not _strictly_ depend on highly accurate GRTT estimation. The current mechanism has proved sufficient in simulations and in the environments where NORM-like protocols have been deployed to date. The estimate provided by the algorithm tracks the peak envelope of actual GRTT (including operating system effect as well as network delays) even in relatively high loss connectivity. The steady-state probing/update interval may potentially be varied to accommodate different levels of expected network dynamics in different environments.

3.7.2. One-to-Many Receiver RTT Measurement

In this approach, receivers send messages with timestamps to the sender. To control the volume of these receiver-generated messages, a suppression mechanism similar to that described for NACK suppression my be used. The "age" of receivers' RTT measurement should be kept by receivers and used as a metric in competing for feedback opportunities in the suppression scheme. For example, receiver who have not made any RTT measurement or whose RTT measurement has aged most should have precedence over other receivers. In turn the sender may have limited capacity to provide an "echo" of the receiver timestamps back to the group, and it could use this RTT "age" metric to determine which receivers get precedence. The sender can determine the GRTT as described in 3.7.1 if it provides sender timestamps to the group. Alternatively, receivers who note their RTT is greater than the sender GRTT can compete in the feedback opportunity/suppression scheme to provide the sender and group with this information.

3.7.3. Many-to-Many RTT Measurement

For reliable multicast sessions that involve multiple senders, it may be useful to have RTT measurements occur on a true "many-to-many" basis rather than have each sender independently tracking RTT. Some protocol efficiency can be gained when receivers can infer an approximation of their RTT with respect to a sender based on RTT information they have on another sender and that other sender's RTT with respect to the new sender of interest. For example, for receiver "a" and sender's "b" and "c", it is likely that:

```
RTT(a<->b) <= RTT(a<->c)) + RTT(b<->c)
```

Further refinement of this estimate can be conducted if RTT information is available to a node concerning its own RTT to a small subset of other group members and RTT information among those other group members it learns during protocol operation.

3.7.4. Sender GRTT Advertisement

To facilitate deterministic NORM protocol operation, the sender should robustly advertise its current estimation of GRTT to the receiver set. Common, robust knowledge of the sender's current operating GRTT estimate among the group will allow the protocol to progress in its most efficient manner. The sender's GRTT estimate can be robustly advertised to the group by simply embedding the estimate into all pertinent messages transmitted by the sender. The overhead of this can be made quite small by quantizing (compressing) the GRTT estimate to a single byte of information. The following C-language functions allows this to be done over a wide range (RTT_MIN through RTT_MAX) of GRTT values while maintaining a greater range of precision for small GRTT values and less precision for large values. Values of 1.0e-06 seconds and 1000 seconds are RECOMMENDED for RTT_MIN and RTT_MAX respectively. NORM applications may wish to place an additional, smaller upper limit on the GRTT advertised by senders to meet application data delivery latency constraints at the expense of greater feedback volume in some network environments.

```
unsigned char QuantizeGrtt(double grtt)
{
    if (grtt > RTT_MAX)
        grtt = RTT_MAX;
    else if (grtt < RTT_MIN)
        grtt = RTT_MIN;
    if (grtt < (33*RTT_MIN))
        return ((unsigned char)(grtt / RTT_MIN) - 1);
    else
        return ((unsigned char)(ceil(255.0 -
                               (13.0 * log(RTT_MAX/grtt)))));
}

double UnquantizeRtt(unsigned char qrtt)
{
    return ((qrtt <= 31) ?
             (((double)(qrtt+1))*(double)RTT_MIN) :
             (RTT_MAX/exp(((double)(255-qrtt))/(double)13.0)));
}
```

Note that this function is useful for quantizing GRTT times in the range of 1 microsecond to 1000 seconds. Of course, NORM protocol implementations may wish to further constrain advertised GRTT estimates (e.g., limit the maximum value) for practical reasons.

3.8. Group Size Determination/Estimation

When NORM protocol operation includes mechanisms that excite feedback from the group at large (e.g., congestion control), it may be possible to roughly estimate the group size based on the number of feedback messages received with respect to the distribution of the probabilistic suppression mechanism used. Note the timer-based suppression mechanism described in this document does not require a very accurate estimate of group size to perform adequately. Thus, a rough estimate, particularly if conservatively managed, may suffice. Group size may also be determined administratively. In absence of a group size determination mechanism a default group size value of 10,000 is RECOMMENDED for reasonable management of feedback given the scalability of expected NORM usage.

3.9. Congestion Control Operation

Congestion control that fairly shares available network capacity with other reliable multicast and TCP instantiations is REQUIRED for general Internet operation. The TCP-Friendly Multicast Congestion Control (TFMCC) [17] or Pragmatic General Multicast Congestion Control (PGMCC) techniques [18] may be applied to NORM operation to meet this requirement.

3.10. Router/Intermediate System Assistance

NACK-oriented protocols may benefit from general purpose router assistance. In particular, additional NACK suppression where routers or intermediate systems can aggregate NACK content (or filter duplicate NACK content) from receivers as it is relayed toward the sender could enhance NORM group size scalability. For NORM protocols using FEC, it is possible that intermediate systems may be able to filter FEC repair messages to provide an intelligent "subcast" of repair content to different legs of the multicast topology depending on the repair needs learned from previous receiver NACKs. Both of these types of assist functions would require router interpretation of transport data unit content identifiers and flags.

3.11. NORM Applicability

The NORM building block applies to protocols wishing to employ negative acknowledgement to achieve reliable data transfer. Properly designed negative-acknowledgement (NACK)-oriented reliable multicast (NORM) protocols offer scalability advantages for applications and/or network topologies where, for various reasons, it is prohibitive to construct a higher order delivery infrastructure above the basic Layer 3 IP multicast service (e.g., unicast or hybrid unicast/multicast data distribution trees). Additionally, the scalability property of NACK-oriented protocols [19], [20] is applicable where broad "fan-out" is expected for a single network hop (e.g., cable-TV data delivery, satellite, or other broadcast communication services). Furthermore, the simplicity of a protocol based on "flat" group-wide multicast distribution may offer advantages for a broad range of distributed services or dynamic networks and applications. NORM protocols can make use of reciprocal (among senders and receivers) multicast communication under the Any-Source Multicast (ASM) model defined in RFC 1112 [2], and are capable of scalable operation in asymmetric topologies such as Single-Source Multicast (SSM) [8] where there may only be unicast routing service from the receivers to the sender(s).

NORM operation is compatible with transport layer forward error correction coding techniques as described in [14] and congestion control mechanisms such as those described in [17] and [18]. A principal limitation of NORM operation involves group size scalability when network capacity for receiver feedback is very limited. NORM operation is also governed by implementation buffering constraints. Buffering greater than that required for typical point-to-point reliable transport (e.g., TCP) is recommended to allow for disparity in the receiver group connectivity and to allow for the feedback delays required to attain group size scalability.

4. Security Considerations

NORM protocols are expected to be subject to the same sort of security vulnerabilities as other IP and IP multicast protocols. NORM is compatible with IP security (IPsec) authentication mechanisms [21] that are RECOMMENDED for protection against session intrusion and denial of service attacks. A particular threat for NACK based protocols is that of NACK replay attacks that would prevent a NORM sender from making forward progress in transmission. Any standard IPsec mechanisms that can provide protection against such replay attacks are RECOMMENDED for use. Additionally, NORM protocol instantiations SHOULD consider providing support for their own NACK replay attack protection when network layer mechanisms are not available. The IETF Multicast Security (msec) Working Group is also developing solutions which may be applicable to NORM in the future.

5. Acknowledgements *(and these are not Negative)*

The authors would like to thank Rick Jones, and Joerg Widmer for their valuable comments on this document. The authors would also like to thank the RMT working group chairs, Roger Kermode and Lorenzo Vicisano, for their support in development of this specification, and Sally Floyd for her early inputs into this document.

6. References

6.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, August 1989.

6.2. Informative References

[3] Mankin, A., Romanow, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, June 1998.

[4] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols". In Proc. ACM SIGCOMM, pages 201--208, September 1990.

[5] Kermode, R. and L. Vicisano, "Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents", RFC 3269, April 2002.

[6] Nonnenmacher, J. and E. Biersack, "Optimal Multicast Feedback," in IEEE Infocom , San Francisco, California, p. 964, March/April 1998.

[7] Macker, J., and R. Adamson, "Quantitative Prediction of Nack Oriented Reliable Multicast (NORM) Feedback", Proc. IEEE MILCOM 2002, October 2002.

[8] Holbrook, H., "A Channel Model for Multicast", Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, California, August 2001.

[9] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002.

[10] Floyd, S., Jacobson, V., McCanne, S., Liu, C., and L. Zhang. "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", Proc. ACM SIGCOMM, August 1995.

[11] Watson, M., "Basic Forward Error Correction (FEC) Schemes", Internet-Draft draft-ietf-rmt-bb-fec-basic-schemes-revised-00, July 2005.

[12] Metzner, J., "An Improved Broadcast Retransmission Protocol", IEEE Transactions on Communications, Vol. Com-32, No.6, June 1984.

[13] Macker, J., "Reliable Multicast Transport and Integrated Erasure-based Forward Error Correction", Proc. IEEE MILCOM 97, October 1997.

[14] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast", RFC 3453, December 2002.

[15] Gossink, D., and J. Macker, "Reliable Multicast and Integrated Parity Retransmission with Channel Estimation", IEEE GLOBECOM 98'.

[16] Ozdemir, V., Muthukrishnan, S., and I. Rhee, "Scalable, Low-Overhead Network Delay Estimation", NCSU/AT&T White Paper, February 1999.

[17] Widmer J., and M. Handley, "Extending Equation-Based Congestion Control to Multicast Applications", Proc ACM SIGCOMM 2001, San Diego, August 2001.

[18] Rizzo, L., "pgmcc: A TCP-Friendly Single-Rate Multicast Congestion Control Scheme", Proc ACM SIGCOMM 2000, Stockholm, August 2000.

[19] Pingali, S., Towsley, D., and J. Kurose, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols". In Proc. INFOCOM, San Francisco, CA, October 1993.

[20] Levine, B., and J.J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Protocols", Proc. International Conference on Network Protocols (ICNP-96), Columbus, Ohio, Oct 29--Nov 1, 1996.

[21] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

7. Authors' Addresses

Brian Adamson
Naval Research Laboratory
Washington, DC 20375

EMail: adamson@itd.nrl.navy.mil

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28334 Bremen, Germany

EMail: cabo@tzi.org

Mark Handley
Department of Computer Science
University College London
Gower Street
London
WC1E 6BT
UK

EMail: M.Handley@cs.ucl.ac.uk

Joe Macker
Naval Research Laboratory
Washington, DC 20375

EMail: macker@itd.nrl.navy.mil

8. Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.