Internet Engineering Task Force
INTERNET-DRAFT
Expires 15 Sept, 1998
March, 1998

Network Working Group
J. Crowcroft (UCL)
L. Vicisano (UCL)
Z. Wang (Bell Labs)
A. Ghosh (UTS)
M. Fuchs (U. Karlsruhe)
C. Diot (INRIA)
T. Turletti (INRIA)

RMFP: A Reliable Multicast Framing Protocol
<draft-crowcroft-rmfp-02.txt>

Status of this Memo

1. Introduction

There has been considerable interest in reliable multicast, and a number of reliable multicast transport applications and systems have

been built in the past years, e.g. [PGM], [RMDP], [RMTP], [SRM]. A survey of most of current reliable multicast protocols is available in [Diot97].

Reliable multicast transport is considerably more complex than reliable unicast. It is generally difficult to build a generic reliable transport protocol for multicast, much as TCP is a generic transport protocol for unicast, since different applications often have very different reliability requirements and modes of operation.

In this document we propose a framing protocol for reliable multicast transport - Reliable Multicast Framing Protocol (RMFP). RMFP runs over multicast UDP and itself does not provide any reliability (or functionality in a larger extend). Reliability and other protocol functionalities will be defined in specific profiles. The purpose of RMFP is to provide a common framework upon which a set of reliable multicast systems can be built and share similar functionalities where exist.

The philosophy of RMFP is in many respects similar to the one of RTP. However, we believe that using RTP for reliable multicast is not a right approach and will not lead to a clean application design.

This draft is intended to stimulate more discussion on the one issue of a generic framing protocol for reliable multicast.

2. Design Philosophy

This section presents the key mechanisms that have been the foundation for the specification of RMFP.

2.1. Error Control

Since RMFP is a framework for reliable multicast, the error control is the most important issue. RMFP itself provides no error control functionality, this is the task of the protocol profiles. However, since RMFP follows the ALF principle [Clark90], some of the error control functionalities have to be provided by the application.

- RMFP specifies the format of the ADUs. The sequence number field and the FEC and retransmissions flags of the ADU header are primarily provided for the protocol profiles to be used for error control.

- Any protocol profile has to be able to detect the loss of ADUs and to initiate the retransmissions. This includes the transmission of control information from a receiver that suffered a loss to some group member that can perform the retransmission.

The ALF principle introduces ADUs as common unit of transmission for all layers from the transport protocol up to the application. To enable the unordered delivery of ADUs each ADU has an ADU name assigned that identifies the ADU data in the application context independent of the history of received ADUs. This ADU name is of no meaning to the transport protocol. However, the transport protocol uses its own naming concept to perform loss detection and recovery -- the sequence numbers.

The remainder of this section assumes, that only one sender is active in the regarded session. This assumption simplifies the problem in a way, but without limiting generality. If there are several senders in a session, each sender will mark its ADUs with his source ID. Each member of the session has its unique source ID, and all packets can be assigned to their sender. Although the following analysis treats only the case of sessions with a single sender, multiple senders in a session can be regarded as independent from each other, and the discussion corresponds to each sender respectively.

2.1.1. The Automatic Repeat Request (ARQ) mechanism

The ARQ mechanism is one of the two basic approaches to ensure reliable data transmission, and is the most reliable. The other mechanism, Forward Error Correction (FEC), can only reduce the loss-rate. ARQ consists of two components: the loss detection and the loss recovery.

Loss detection:

Even if the application does not need any ordering of the data, the protocol will use some kind of sequence numbers to assign an order to its ADU stream. Normally, this order corresponds to the sending order. Losses are detected by means of gaps in the sequence number space. The actual algorithm can reside at the receiver (receiver-based loss detection) or at the sender (sender-based loss detection). The loss detection algorithm uses some state information, the history of ADUs already received successfully, and computes the necessary information to do the loss recovery, i.e. the sequence numbers corresponding to the lost ADUs.

- The receiver-based algorithm detects the lost ADUs at the receiving protocol instance, that will encode and transmit the sequence numbers of the lost ADUs in control packets. The encoding is mostly done in form of spans to reduce the necessary bandwidth. The addressee of the control packets (in a unicast transmission this is the sender) can then compute the sequence numbers of the lost packets without other information.

- The sender-based algorithm requires that the sender is in charge for the reliability of all the receivers. Consequently, the sender has to keep status information on all of the receivers, see [Levine98].

In the remainder of the report only the receiver-based approach will be considered, since at least for multicast, the sender-based approach has several disadvantages (a comparison of the receiver- and sender-based approach can be found in [Pingali94]):

- To detect the gaps, the history of lost and received ADUs has to be available. If the sender has to do this, the number of receivers would be limited by the senders capacity in keeping this state information.

- Since the sender has to track the history of all ADUs at all receivers, it has to process the control packets from all receivers. With many receivers, the sender will suffer the so-called ack-implosion. This is an overload of the sender by processing the control packets. Some receiver-based protocols use the so-called NACK suppression mechanism to prevent the overload of control packets. A receiver that suffered a loss, does not need to send a control packet with lost ADU information, if another receiver has done so before for the same ADU. If the retransmission for the first request is transmitted, both receivers will receive it.

The following two protocols have been investigated more thoroughly for integration into RMFP as profiles:

The SRM [SRM] protocol uses a receiver-based mechanism with NACK suppression to free the senders completely from management tasks for special error control state information and to avoid the ack-implosion.

The LGC protocol [Hofmann97] uses a combined approach. To prevent the nack-implosion at the sender, LGC builds a tree structure with the

sender as source. The control packets are not sent directly to the sender, but are gathered at the inner nodes (group controllers) of the tree. Thus, the sender and each of the group controllers has only to process the control packets of its children.

Loss Recovery:

For unicast transmission, the sender of the retransmissions is always the original sender. For multicast transmission, receivers that have successfully received a given PDU can also retransmit that PDU to the receivers that have lost the PDU. An example protocol is SRM, where every group member is involved in loss recovery.

2.1.2. Forward Error Control

FEC reduces the loss-rate in sending redundancy information additionally to the useful data. The encoding takes a block of n ADUs and computes a given number k of redundancy packets. The n+k packets form a transmission group. If the packets of a transmission group are sent, it is sufficient to receive any subset of size n of the transmission group to reconstruct the original n ADUs. However, if more then k packets of the transmission group get lost, the losses cannot be repaired. Thus, FEC can only reduce the packet loss-rate. An introduction to FEC can be found in [Rizzo97].

FEC can be combined with ARQ to the so-called hybrid ARQ. This mechanism is especially useful for reliable multicast, since it can effectively reduce the overall loss-rate and thus retransmissions, too. An investigation of hybrid ARQ is presented in [Nonnenmacher97].

There are several possibilities to use FEC in RMFP:

- The usage of FEC within RMFP transparent for the protocol profile, i.e. as some layer under the profile could improve the behavior of all profiles. The effects of such a transparent FEC mechanism have been investigated in [Huitema96] and [Nonnenmacher97].

- FEC can be implemented as a mechanism of a protocol profile.

- The application can implement the FEC mechanism or use some standard module provided by a RMFP implementation, see [Fuchs98].

Sequence numbers are generally ignored when the FEC bit is set. However, specific profiles can use the sequence number field to encode specific protocol information relative to the FEC packet. The

transmission of an FEC packet does not increment the sequence number counter at the source. This insulates the mechanism for detecting normal packet loss from the FEC recovery scheme.

2.1.3. ALF and loss recovery

According to the ALF principle, the application has to handle the data retransmission. In RMFP the protocol profiles have the task to detect the losses and inform the application about the need of retransmissions. The application then provides the retransmission data. However, the protocol profiles use the sequence numbers to identify ADUs, whereas the application requires the ADU name to identify the ADUs. This leads to the need for a mapping between the protocols sequence numbers and the ADU names.

The retransmissions of ADUs can only be performed by group members that have the ADU either sent themselves or received already successfully. Since the complete ADU contains both the sequence number and the ADU name, the mapping information required to provide the retransmission data is already available at the retransmitting group member. The member can map the sequence number to the ADU name and then the ADU name to the retransmission data. Depending on the management of the retransmission data, the mapping may also be performed directly from sequence number to retransmission data.

The RMFP specification doesn't specify, if the mapping from sequence number to ADU name should be performed already at the protocol profile or at the application; this decision is implementation dependent.

2.2. Hierarchical Naming with Objects

Additionally to the sequence number field and the ADU name there is another field in the ADU header to support the mechanisms to identify the data carried in the ADUs: The object ID field. It can be used to optimize the transmission overhead caused by the ADU name.

For example, a file transfer application can put the name of the file into the ADU name field of each ADU. If the file name includes some path name, the file name can become considerably big. This file name, however, doesn't change for all the ADUs belonging to the file; only the byte-offset field varies from ADU to ADU.

The object ID field can reduce the bandwidth required by the ADU name. Each file name used during the transmission is mapped onto a unique object ID. The file name can then be omitted in the ADU name. The

problem with this approach is the transmission of the mapping information of object ID to ADU name that is required at the receivers to process the ADUs. It can be transmitted in one of the ADUs of the file in the ADU name field or separately as session information. In the example, the first approach has the disadvantage, that all ADUs of the file can only be processed, when the ADU with the file name in the ADU name field has been received successfully. The other approach has the disadvantage, that the session packet has to be transmitted reliably, since the ADUs of a file are only useful, if the file name is known.

How the object ID field is used is up to the application. It has to find the optimal way to suit its requirements and to optimize the used bandwidth.

Another issue is the relationship between objects and sequence numbers. Three possibilities are suggested:

1/ The object ID is independent to the sequence number field and is only used by the application. The ADUs are sequenced relative to the start of the session and are not influenced by the object ID. This is suitable for applications that require all ADUs to be received reliably. This is the mechanism defined at the specification the SRM profile.

2/ The sequence numbers are computed relative to the objects and the object IDs are sequenced. If the objects are transmitted one after the other, i.e. the ADUs of several objects are not interleaved, every two ADUs can be compared in respect to their sending order.

   To reorder the ADUs and to detect ADU losses at the receiver, the object IDs and sequence numbers are compared hierarchically: Since the objects are transmitted sequentially, the sending order of two ADUs can be computed out of the object ID, if the object IDs of the ADUs differ. If both ADUs belong to the same object, the sequence number decides about the order. The loss detection is more difficult than with the first sequencing approach:

   - Lost ADUs within an object are detected by gaps in the objects sequence number name-space.

   - Objects lost in total are detected by gaps in the object ID name-space.

- If the first or last ADUs of an object are lost, the start-of-object/end-of-object flags are used to detect the losses.

These mechanisms are sufficient to be able to detect all possible ADU losses, although, in the third case, it is not always possible to determine the number of all lost ADUs and their sequence numbers. The coding of negative acknowledgments for retransmission requests must be performed as spans.

The problematic loss of ADUs around object boundaries (i.e. the loss of ADUs carrying start-of-object/end-of-object flags) imposes the constraint on the transmission order of objects: The transmission of an object must be completed (by an ADU carrying the end-of-object flag) before the first ADU of the next object (i.e. an object with an object ID incremented by one) can be sent. This limits the usability of this approach for applications that want to transmit several objects simultaneously, e.g. a white-board application. Such applications require the next model.

3/ The sequence numbers are computed relative to the objects, i.e. every object has its own sequence number space, but there is no ordering relation between the ADUs of different objects. This requires, however, that all control information has to refer to each object independently, too. In [Fuchs98] a concept is presented that is based on this model of sequencing. It allows the receiver application to decide, which objects have to be received reliably (semantic reliability). Another very general approach of how this can be done is described in [Raman97].

2.3. Late-joining Receivers

An important problem for reliable multicast is the synchronization of late-joining receivers. In general, applications may require to allow receivers to join an ongoing session. Such receivers have to figure out, at which point of the ADU stream they start with the receipt of data.

The following discussion assumes, that the ADUs are sequenced relative to the session and not relative to the objects (see [Fuchs98]), since this is the method used in the current specification of RMFP.

In the rest of the section the term initial sequence number refers to the sequence number of the packet with the lowest sequence number that a receiver processes. A receiver keeps information about the initial

sequence number for each sender independently. Similarly, the highest-sequence-number-sent is the highest sequence number used by the sender. For a receiver, this is actually the highest sequence number seen from a given sender so far.

Several solutions are possible:

- The receiver uses the ADU with the lowest sequence number it receives. It won't ask for retransmissions for any ADU with a lower sequence number.

- The senders transmit synchronization points as session information. Those synchronization points are sequence numbers within their ADU stream, that are determined by the application and are useful in the application context. A joining receiver that receives such information, can ask for retransmission of all ADUs starting at this synchronization point.

It is up to the application to decide, which style of receiver synchronization to use. Consequently, the RMFP supports both. The senders transmit the information of the style to use and if necessary the current synchronization point within the sender report packets.

RMFP defines following behavior at a joining receiver:

1/ The receiver has no information yet. This means that the receiver has not yet received any information about the sequence numbers sent by the sender.

   - ADU received: The sequence number of this ADU is used as an initial sequence number.

   - Highest-sequence-number-sent received: This information is carried e.g. by a SRM heartbeat control packet. The next sequence number is used as the initial sequence number.

   - Synchronization point received: The receiver takes the synchronization point as the first sequence number of the ADU stream from the sender. Since the sender report packet carrying the synchronization information also carries the highest-sequence-number-sent, the receiver can ask for retransmission for all ADUs starting with the synchronization point's sequence number and up to the highest-sequence-number-sent.

2/ The receiver is synchronized without synchronization point received. The receiver is already synchronized due to a received ADU or highest-sequence-number-sent information.

- ADU received: If the ADU's sequence number is lower than the present initial sequence number for that sender, the initial sequence number is set back to the ADU's sequence number and missing packets starting with this sequence number are requested for retransmission.

- Highest-sequence-number-sent received: It should be greater than the already known initial sequence number, which has no impact on the synchronization. If it is not, which could happen in case of out-of-order receipt of control packets, this information is discarded.

- Synchronization point received: If the synchronization point's sequence number is greater than or equal to the initial sequence number, the information is regarded as obsolete. Otherwise, the initial sequence number is set back to the received sequence number and the missing packets are requested for retransmission.

3/ The receiver has already received a synchronization point. This implies, that the synchronization process is already finished. Received synchronization information is not considered anymore at all, and ADUs with lower sequence numbers than the used synchronization point are discarded.

Because of the finite sequence number space, there are problems with the described synchronization algorithm. To ensure proper operation the synchronization process has to be stopped after a defined span of sequence numbers has been seen by a receiver (again independently for each sender). In the implementation the size of the span is a quarter of the sequence number space. At this point the receiver assumes that it is fully synchronized.

2.4. Automatic Profile Configuration

One of the foundations that provide flexibility in RMFP are the different protocol profiles. The protocol profiles have different characteristics and the optimal protocol profile depends on the scenario, i.e. the number of group members, the number of senders etc. If it is clear at the development of an application, that one of the protocol profiles is a good choice for all envisioned scenarios for

the application, the application can always use that profile and every group member always knows this profile, when it joins.

However, for some applications it might prove useful to support several protocol profiles. The information of the profile has to be distributed to all members. RMFP provides a mechanism for joining members to be configured automatically by received sender control packets. However, this mechanism works correctly only if the senders of a session agree about the profile. RMFP provides no mechanism to deal with conflicts, if members of the same group use different profiles.

2.5. External Modules

Some of the standard functionality of other transport protocols have been omitted in RMFP to allow the applications to use the transport functionality in a more flexible way. However, many applications could use the standard functionality. To simplify the use of RMFP it is possible to use some implementations of this functionality as external modules. Some possible modules are the following:

- Retransmission buffer: According to the ALF principle the application is responsible to manage the retransmission data. This brings flexibility, but many application programmers might want to use the classical mechanism, a simple buffer indexed by the sequence numbers.

- Reordering module: ALF explicitly introduces the unordered delivery of received ADUs. Applications, that do not require the flexibility and performance of that mechanism or are not even capable to process the ADUs out-of-order could be implemented simpler, if they could rely on ordered delivery.

3. The RMFP Specification

RMFP specifies three types of packets:

- Data packets (corresponding to ADUs) sent by senders.

- Control packets sent by senders and receivers control.

- Sessions packets that can be defined by the application using the generic session packet header.

The protocol profiles that provide the reliability can define their own control packets. Those profile specifications are not part of the RMFP specification, but are defined separately. Two profile specifications for SRM and LGC can be found in [Fuchs98].

3.1. General Aspects

3.1.1. Network environment

This specification suggests an addressing scheme for the different packet types: For each of the three packet types -- ADU, control and session -- RMFP uses the same IP multicast address, but different UDP ports. Since all packets can be identified due to their type field, they could be well sent on the same IP multicast address/UDP port. However, such an approach can lead to inefficiencies at the buffer management, since the type of a received packet can only be retrieved after the packet has been copied into the application buffers. That's why RMFP relies on UDP to multiplex/demultiplex the three flows.

Some protocol profiles may need to use more addresses and/or ports or cannot even use the global multicast groups in which every group member takes part. However, the profile developers should seek to be as compliant as possible to this suggestion to reduce profile specific differences at the API.

This specification requires the application to provide a single address/port pair for the session, the session address and the session port.

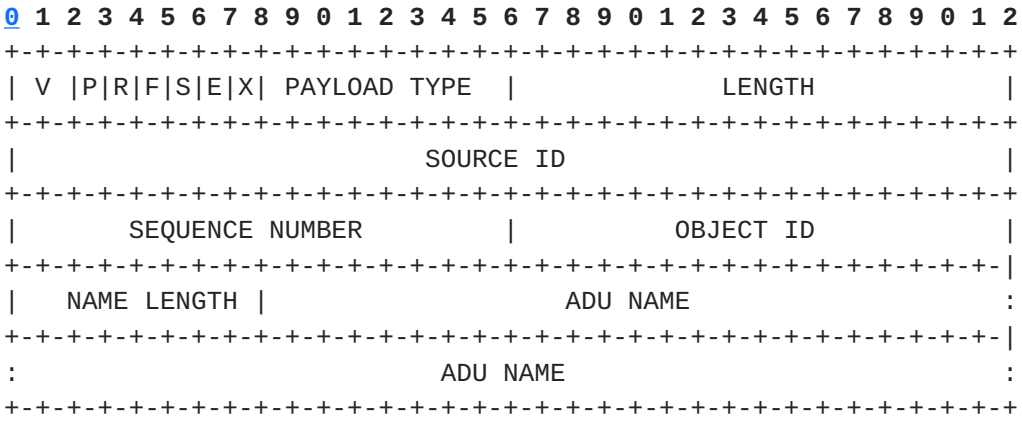
- The data flow (all the ADUs) is assigned to the session address/session port.
- The control flow (sender and receiver report packets as well as the profiles' control packets) is assigned to the session address/session port + 1.
- The session flow (all application defined session packets) is assigned to the session address/session port + 2.

3.1.2. System environment

To avoid problems with alignment, all packet fields are naturally aligned, e.g. all two-octet sized fields are placed on even addresses. The packets themselves are assumed to be four-octet aligned.

3.2. RMFP ADU Format

The RMFP ADUs have the following header format:



The intention in designing this format was to include enough information to be sufficient for the different protocol profiles, but to keep the overhead small.

Version(V): 2 bits
This field identifies the version of RMFP.

Padding(P): 1 bit
If the padding bit is set, the packet contains one or more additional padding bytes at the end, which are not part of the payload. The last octet of the padding contains a count of how many bytes should be ignored. The padding bytes keep all the ADUs four-byte aligned.

Retransmission (R): 1 bit
Set to 1 if the ADU is being retransmitted.

Forward Error Correction (F): 1 bit
Set to 1 if FEC is used.

Start of Object (S): 1 bit
Set to 1 if the ADU is the first one of an object.

End of Object (E): 1 bit
Set to 1 if the ADU is the last one of an object.

Exceptional Handling (X): 1 bit
This bit is free for use by the application. It is not processed at

RMFP or any profile and is intended to allow the application to mark ADUs that should be treated in a unusual way.

Payload Type: 8 bits
This field is intended to serve the application in a similar way as the payload type field in RTP [Schulzrinne95] does. The application can use this field to indicate the type of the payload. Some values of this field are used to indicate control or session packets used by RMFP and the profiles and may not be used for application purposes. Following values are so far defined:

- o 201: Sender report packets.
- o 202: Receiver report packets.
- o 203: Session packets.
- o 205: SRM control packets.
- o 206: LGC control packets.

The application can use the other values freely, however, it is possible that other values above 200 may be used by other profiles, or added functionality in future versions of RMFP.

Length: 16 bits
This field identifies the length of the packet in 32 bits minus one, including the header and any padding. To count in multiples of four bytes avoids an alignment check. This algorithm has been introduced by RTP.

It can be used to combine several ADUs into one UDP packet. In a compound UDP packet only the length fields allow the detection of the ADU boundaries.

When several ADUs (original and retransmitted) are concatenated within one UDP packet, the original ADUs should all be placed at the beginning of the UDP packet so that receivers that do not encounter losses can just drop the tail of the retransmitted ADUs without processing it.

Source ID: 32 bits
This field identifies the source. The source IDs are generated randomly similar to the SSRC field in RTP to avoid collisions between several members.

Sequence Number: 16 bits
The sequence number is an ADU counter. It is incremented by one for each ADU sent. It can be used to detect ADU losses and calculate loss rates. The exact semantics of the sequence number is determined by the protocol profile. It is possible to count the sequence number starting with the first ADU sent and incrementing it for each ADU throughout the session. Another possibility would be to use the sequence number object-relative, i.e. each object has its own counter assigned starting at zero for its first ADU.

Object ID: 16 bits
This field identifies the object carried in the packet.

Name Length: 8 bits
This field specifies the length in bytes of the following ADU Name. zero is a valid value, indicating that no explicit ADU name is available.

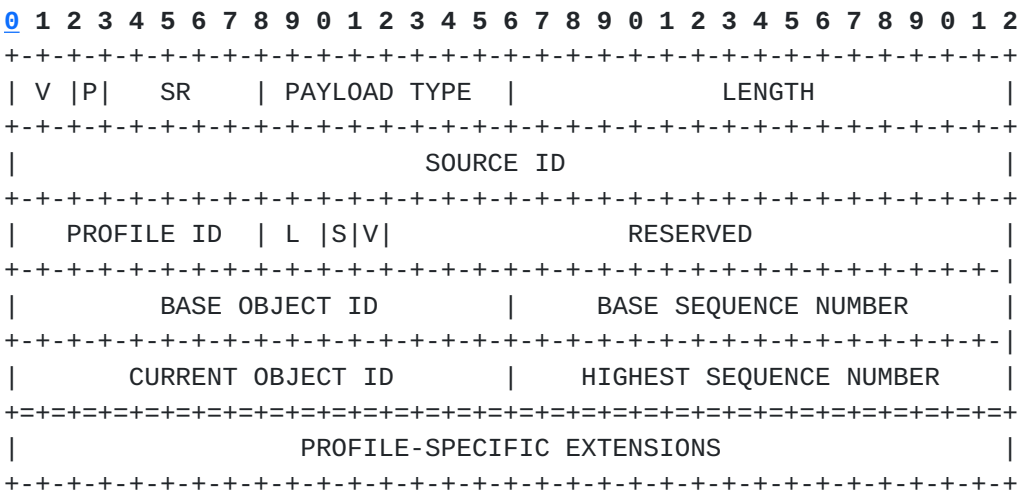
ADU Name: variable length
The ADU name is used by the application to identify an ADU in the application context. The contents of this field are completely transparent to RMFP and the protocol profiles. The length of the ADU name can be between 0 and 255 bytes. There can be unused bytes to ensure proper alignment (32bit) within the ADU header. This field can contain the information to identify both the object and the position within the object of the ADU, e.g. the filename and the byte-offset for ADUs in a file transfer application. However, the application can also use the object IDs and sequence numbers to identify objects and ADUs.

3.3. RMFP Control Packet Format

RMFP control packets include sender report packets and receiver report packets. Those packets can be used by the senders and receivers respectively to transmit session information.

3.3.1. Sender Report packet

Sender report packets are sent periodically by the sender and contain information about the current sending state. They can help to configure new joining receivers and provide information to detect tail losses. The structure of the header is shown in the following figure:



Version(V): 2 bits
  This field identifies the version of RMFP.

Padding(P): 1 bit
  If the padding bit is set, the packet contains one or more additional padding bytes at the end which are not part of the payload. The last octet of the padding contains a count of how many bytes should be ignored. Since the actual header is already aligned, the padding flag is only necessary, if an application specific extension is included in the packet.

SR Type: 5 bits
  This field has no interpretation by RMFP and can be used by the application, e.g. to transmit extra information like an end of transmission indication. It might also be used to denote the type of the application specific extension.

Payload Type: 8 bits
  This field is set to 201 for sender report packets

Header Length: 16 bits
  This field specifies the length of the packet in multiples of 32 bits minus one.

Source ID: 32 bits
  This field identifies the sender.

Profile: 8 bits
  This field indicates the type of the protocol profile used. It is used together with the LSV, first object ID and lowest sequence number

fields to configure late joining receivers. A receiver that wants to join a session and does not know a-priori which protocol profile is used, can wait for receipt of a sender report packet and configure its protocol profile according to this field.

Lowest Sequence Valid (LSV): 2 bits
These bits define the interpretation of lowest sequence number field:

- o 00: The sequence number of the first ADU sent by the sender in this session.
- o 01: The sequence number of some position in the transmission that can be used to synchronize.
- o 10: No valid information. The sender provides no special help to synchronize. The new receiver should synchronize its join on the first ADU it receives.
- o 11: Reserved.

If the lowest sequence number fields is valid, a late-joining receiver can ask for retransmission back to the indicated sequence number. The sender can choose the value of this field appropriately to mark some logical boundary in the ADU stream.

First Object ID: 16 bits
The object ID for late-joining receivers to synchronize.

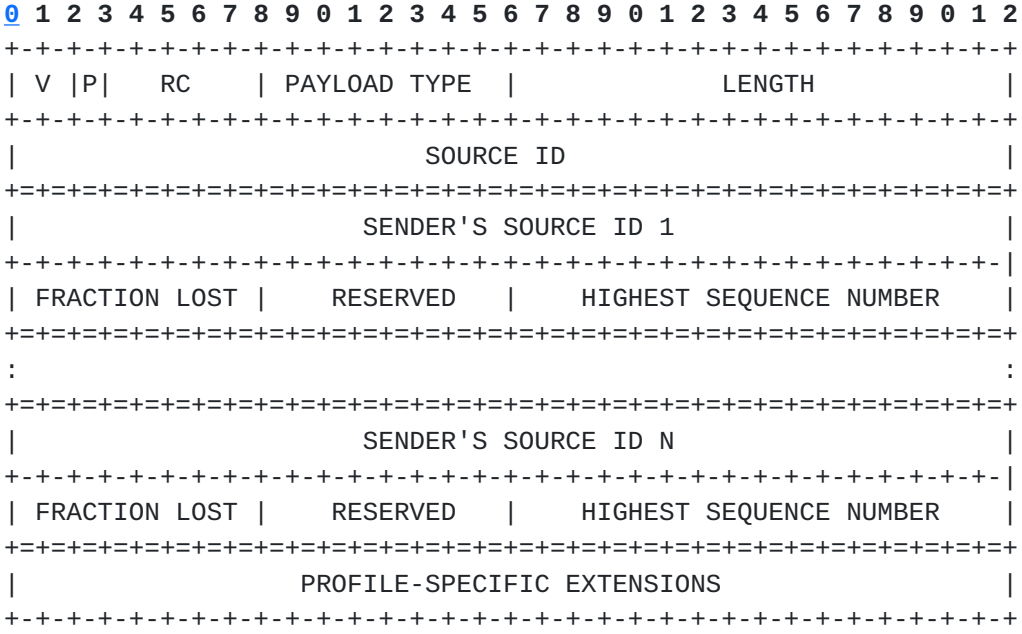
Lowest Sequence Number: 16 bits
This field encodes the sequence number for late-joining receivers to synchronize.

Current Object ID: 16 bits
This field and the highest sequence number field are used to indicate the current state of the sender. The receivers can use this information to detect tail-losses.

Highest Sequence Number: 16 bits
This field comes together with the current object ID and is the sequence number of the last ADU sent. It is used to detect tail-losses.

3.3.2. Receiver Report packet

Receiver report packets are sent periodically by the receivers to give feedback on congestion and packet losses. They contain some receive statistics for each sender. The format of this packet type is shown in the following figure.



The fields have the following meaning:

Version(V): 2 bits
  This field identifies the version of RMFP.

Padding(P): 1 bit
  The padding bit is used to force alignment of the packet. It is used in the same way as in the sender report packet.

Report Block Count(RC): 5 bits
  The RC denotes, how many report blocks are contained in this packet. Each report block consists of a sender's source ID, a fraction lost field and a highest sequence number field.

Payload Type: 8 bits
  Set to 202 for receiver report packets.

Header Length: 16 bits
  This field specifies the length of the packet in multiples of 32

bits minus one.

Source ID: 32 bits
Identifies the sender of this packet (a receiver).

Senders Source ID X: 32 bits
This field identifies the sender X corresponding to the following fraction lost and highest sequence number information.

Fraction Lost: 8 bits
The fraction of packets lost since last receiver report, expressed as a fixed point number with the binary point at the left edge of the field. Fraction lost is the loss rate seen by the receiver in respect to the sender identified by the previous sender's source ID field. The information may be used for congestion control or error recovery (FEC) by the sender.

Highest Sequence Number: 16 bits
Indicates the highest sequence number received from the corresponding sender so far.

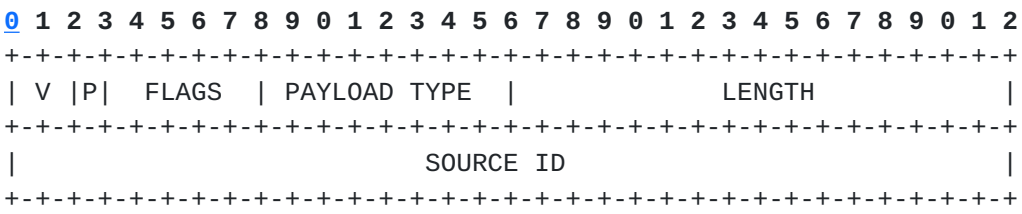
3.4. RMFP Session Packet Format

The session packets are used to enable group members to easily exchange session information. RMFP defines a very light-weight approach, that merely supports the sending and receiving of unreliable data, that is marked as session information. Thus the RMFP just defines the protocol header and provides the transmission and receipt of such packets. There are no special packets defined for some specific use, this is up to the application. Session packets can be used e.g. to support the following functions:

- - Remote configuration: A sender can transmit configuration parameters to configure other members. This mechanism is only used to transmit parameters. The application has the responsibility to use the parameters to configure the protocol.

- - Support at joining a session: A member joining a session has to be informed about the current state of the session. For small groups, it could use a special session packet to issue some status request packet, and the senders can answer to that packet with some status reply session packets.

The RMFP session packet has the following format:



Version(V): 2 bits
The version of RMFP.

Padding(P): 1 bit
The padding bit is used to force alignment of the packet. It is used in the same way as in the sender report packet.

FLAGS: 5 bits
The usage of this field is defined by the application. It could be used e.g. to identify different types of session packets.

Payload Type: 8 bits
This field is set to 203 for RMFP session packets.

Length: 16 bits
This field specifies the length of the packet in multiples of 32 bits minus one, including the header and any padding.

Source ID: 32 bits
This field identifies the sender of the session packet. It is calculated like the length field of the ADU.

Addresses of Authors

J. Crowcroft, L. Vicisano
{j.crowcroft,l.vicisano}@cs.ucl.ac.uk
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
UK

Zheng Wang
zhwang@dnrc.bell-labs.com
Bell Labs Lucent Tech.
101 Crawfords Corner Road
Holmdel NJ
USA

Atanu Ghosh
atanu@socs.uts.EDU.AU
School of Computing Sciences
University of Technology
Sydney
PO Box 123 , Broadway
NSW 2007
Australia

Michael Fuchs
Michael.Fuchs@telematik.informatik.uni-karlsruhe.de
Institute of Telematics
University Karlsruhe
Germany

Christophe Diot, Thierry Turletti
{cdiot,turletti}@sophia.inria.fr
INRIA Sophia Antipolis
2004 route des Lucioles
BP 93, 06902 Sophia Antipolis
France

References

[Clark90]
D. Clark, D. Tennenhouse, Architectural Considerations for a New Generation of Protocols, Proceedings of ACM SIGCOMM '90, Sept. 1990, pp. 201-208.

[Diot97]
C. Diot, W. Dabbous, and J. Crowcroft, Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms, IEEE/JSAC, Vol. 15, No. 3, pp. 277-290, April 1997.

[SRM]
S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang, A Reliable Multicast Framework for Light-weight Session and Application Level framing, IEEE/ACM Transactions on Networking, Dec. 1997, Vol. 5, No 6, pp. 784-803.

[Fuchs98]
M. Fuchs, C. Diot, T. Turletti, and M. Hofmann, A Framework for reliable Multicast in the Internet, INRIA Research report No 3363, Feb. 1998. See also the RMFP Home page at URL ``www.inria.fr/rodeo/rmfp/''.

[Hofmann97]
M. Hofmann, Enabling group communication in global network, Proceedings of Global Networking '97, Calgary, Canada, June 1997.

[Huitema96]
C. Huitema, The case for packet level FEC, Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)}. INRIA, Sophia Antipolis, France, Oct. 1996.

[Levine98]
B.N. Levine, and J.J. Garcia-Luna-Aceves, A Comparison of Reliable Multicast Protocols, to appear in ACM Multimedia Systems Journal, August 1998.

[Nonnenmacher97]
J. Nonnenmacher, E. Biersack, and D. Towsley, Parity-Based Loss recovery for Reliable Multicast Transmission, Proceedings of ACM SIGCOMM '97, Sept. 1997.

[Pingali94]
S. Pingali, D. Towsley, and J. Kurose. A Comparison of Sender-

Initiated and Receiver-Initiated Reliable Multicast Protocols, Proceedings of SIGMETRICS'94, 1994.

[PGM]
T. Speakman, S. Farinacci, S. Lin, and A. Tweedly, Pretty Good Multicast (PGM) Transport Protocol Specification, Internet Draft, draft-speakman-pgm-spec-00.txt, January 1998.

[Raman97]
S. Raman, and S.R. McCanne, General Data Naming and scalable State Announcements for Reliable Multicast, Technical report, Computer Science Division (EECS), University of California, June 1997.

[Rizzo97]
L. Rizzo, On the feasibility of software FEC, Technical report, Universita di Pisa, January 1997.

[RMTP]
J.C. Lin, and S. Paul, RMTP: A Reliable Multicast Transport Protocol, Proceedings of IEEE INFOCOM '96, pp. 1414-1424.

[RMDP]
L. Rizzo, and L. Vicisano, A Reliable Multicast data Distribution Protocol based on software FEC techniques, Proceedings of the 4th IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97), Sani Beach, Chalkidiki, Greece June 23-25, 1997.

[Schulzrinne95]
H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 1889, November 1995.

[TIBnet]
TIBCO, TIBNnet White Paper, "http://www.tibco.com/products/tibwhite.html"

[Vicisano98]
L. Vicisano, l. Rizzo, and J. Crowcroft, TCP-like congestion control for layered multicast data transfer, to appear in Infocom'98.

Table of Contents