

INTERNET-DRAFT  
Expires: December 1999

Carsten Bormann, Joerg Ott  
Universitaet Bremen  
Nils Seifert  
Telligence  
June 1999

MTP/SO: Self-Organizing Multicast  
draft-bormann-mtp-so-02.txt

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

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/lid-abstracts.txt>

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

Abstract

MTP/SO is a reliable multicast protocol based on the earlier protocols MTP ([RFC1301](#)) and MTP-2, simplifying the protocol considerably while adding self-organization of the members of the group into a hierarchy of local regions, local retransmissions, local NAK damping, and both global and local forward error correction. MTP/SO retains the coordinated many-to-many multicast model of MTP-2 while improving scalability.

INTERNET-DRAFT

MTP/SO: Self-Organizing Multicast

June 1999

## 1. Introduction

Multiparty cooperative applications have received much attention over the past years, as has the multicasting of datagrams in the Internet. The Internet datagram multicasting mechanism is not reliable, often requiring a higher level protocol to achieve the level of reliability required for an application.

Much of the early work on reliable multicast protocols has assumed relatively stable groups that need to ensure that all messages are eventually received by all members of this well-defined group. Recently, work on loosely coupled teleconferencing has directed attention to a class of multicast applications that scale up to an extent where this assumption is no longer practical. Many other applications in the area of synchronous groupware also do not need the strong property of reliability, but can nonetheless benefit from a multicast protocol providing some weaker form of reliable transport.

An interesting multicast transport protocol with a somewhat relaxed view of reliability is defined in [RFC 1301](#) [1]. MTP can be used with unreliable and not necessarily sequence preserving underlying multicast (or broadcast) network protocols such as IP multicast. MTP provides globally ordered, receiver reliable, rate controlled and atomic transfer of messages to multiple recipients.

A revised version of MTP, the Multicast Transport Protocol MTP-2, has been used for a number of applications for some time [2]. MTP-2 has been designed to avoid some of the practical problems experienced in using MTP and introduces a number of additional facilities that increase its utility. In particular, MTP-2 no longer has a single point of failure.

This document defines Self-Organizing Multicast, MTP/SO. MTP/SO uses MTP-2 as a basis and adds spontaneous self-organization of the members of the group into a hierarchy of local regions. Scalability is increased by providing passive group joining and local retransmission of lost packets, as well as forward error correction (FEC).

## 2. Requirements

Even more so than for unicast protocols, there are difficult trade-offs in designing a multicast protocol. It is unlikely that a single reliable multicast protocol can be applicable to all kinds of multicast applications, from a small set of replicated database systems synchronizing their updates, to distributed interactive simulation systems with hundreds of thousands of processes joining and leaving large numbers of groups with high frequency.

Any design of a protocol that aims to cover a part of the ground must therefore be explicit about the specific requirements the designers had in mind. Concentrating on any single objective is unlikely to

Bormann, Ott, Seifert

[Page 2]

---

INTERNET-DRAFT

MTP/SO: Self-Organizing Multicast

June 1999

yield a generally applicable protocol. In this section, we list what we perceive to be the main requirements that went into the design of MTP/SO, in order of importance.

- o Scalability

While the actual usage pattern of synchronous group communication software is not yet known, it is clear that groups of wildly different sizes will need to be accommodated. A protocol that is not scalable to large groups with a significant rate of membership change will not be a viable multicast platform.

Many existing protocols that focus on reliability require a positive acknowledgment from each recipient to the sender of each message. This does not scale to large groups without elaborate aggregation schemes. Also, group management algorithms that require an acknowledgment from each member to accept a new member are not acceptable in large groups (in particular, building a group creates an  $n$ -square problem).

As a first level of attack, this scaling problem can be circumvented by using negative acknowledgements (NAKs). Unfortunately, this also conflicts with a strict reliability requirement: Not every failure will be immediately detected, since the normal behavior of a recipient, i.e. being silent, cannot be distinguished from a silent failure. There is a trade-off between scalability and the kind of reliability that can be realized.

- o Efficiency

A reliable multicast protocol should be comparable in performance to special protocols specifically designed for an application. Just as

TCP generally is slightly less efficient than a specially designed protocol would be, some more packets and additional per-packet overhead as well as some additional processing time will be tolerable. However, the protocol needs to be in the same class of overhead to be applicable to an application.

#### o Robustness and Reliability

A reliable multicast protocol should obviously be ``reliable'' in some sense. Given the conflict with scalability, we define reliability to mean: A recipient can (within bounded time) find out when it is failing or being partitioned from active senders. A sender is assured (with sufficient probability) that all its messages reach within bounded time all recipients that are not failing or being partitioned.

Obviously, this strict definition of reliability needs to be complemented by some measure of robustness: A protocol that declares failure or creates significant delays in the face of trivial errors may meet this definition but is not useful. In a teleconferencing environment, a desirable robustness property is the ability to

Bormann, Ott, Seifert

[Page 3]

---

INTERNET-DRAFT

MTP/SO: Self-Organizing Multicast

June 1999

continue operating within partitions should the group become partitioned. Ultimately, the applications that use the multicast transport platform should be the ones to decide when the situation has deteriorated to a point where continuing is meaningless.

#### o Ordering

Many applications are simplified considerably when (at least a certain subset of all) messages exchanged in the group arrive in the same order at all recipients, even if originated at different senders. This requirement distinguishes MTP/SO from other multiple sender multicast protocols such as SRM [5] that work best when the shared state of the multicast group is the (commutative and associative) sum of the independent contributions of all participants.

### [3.](#) Overview

This section gives an overview over the protocol functions of MTP/SO. (Note to readers that have seen MTP or MTP-2: This overview is given in terms that are more generic than those used in older protocol definitions. In particular, the terms group, coordinator, sender, and receiver have been substituted for the traditional terms web,

master, producer, and consumer.)

In MTP/SO there are three different roles of members in a group: coordinator, sender and receiver. The coordinator provides the message ordering for all members in a group and oversees the global rate management. Senders send data in messages (each sent as a sequence of one or more data packets) after obtaining a token from the designated coordinator. Receivers receive these messages and request the retransmission of packets that did not arrive.

In MTP/SO, many actions such as retransmitting control packets or requesting retransmissions depend on a time interval that is a parameter to the whole group. This interval is called heartbeat and is measured in microseconds.

### 3.1. Global ordering

The coordinator assigns a global sequence number to each message. In the simplest mode of transmission, before a sender is allowed to start sending a new message, it has to obtain a token from the coordinator. This can be done by transmitting a special request packet to the coordinator or by sending the request along with data packets belonging to other messages. The coordinator answers with a confirm packet, which contains the sequence number for the new message. Senders will then send this sequence number in every data packet belonging to the message. It is the responsibility of the receivers to deliver messages in the correct order to the applications, if sequenced delivery has been specified for a message.

This results in an ordering class called ``global ordering'', which means that even when there are many senders simultaneously sending messages, every receiver will receive the messages in the same order (which comes close to the order in which the tokens were requested).

As the sequencing will quite often result in an additional delay (for example when a short message is preceded by a very long one), applications can assign messages to different streams. A message is delivered irrespective of messages belonging to other streams, even if these carry lower sequence numbers. By using streams, applications can avoid unnecessary delays, simply by assigning independent messages to different streams.

A message that can be processed independent of the ones preceding it

can be marked with a sequencing\_off bit. Messages so marked can be immediately delivered to the application by receivers, even if the stream numbers of preceding messages are still unknown.

Normally the coordinator grants the tokens in the same order the token request packets are received. If there is a need to transmit some messages with a higher priority, applications can assign a priority to every message. This priority is only considered while granting a token (hence only when there are many tokens requested at the same time) and has no effect on the transmission rate of the message once a token has been assigned. As a result, when a sender sends messages with different priorities, it is no longer guaranteed that these are received in the same order they were queued for sending -- if they are in the same stream, they are, however, received in the same order by all receivers (including the sender).

### 3.2. Rate and Load management

Rate management is overseen by the coordinator. A parameter global to the group defines the maximum bandwidth to be used by the group. The coordinator dynamically adjusts a per-message parameter called window to divide up the total rate into the number of tokens currently granted, controlling the inter-packet-interval at which senders pace data packets belonging to one message. So the coordinator can ensure that the maximum throughput defined for the group is not exceeded.

An argument often heard against using a central coordinator is that it might limit scalability by becoming a bottleneck. First, it needs to be noted that in the worst case (all messages are one packet long) the coordinator handles three times as many packets as each other group member that does not send: one (small) token request, one (small) token confirm, and the actual reception of the data packet. There is no scalability problem involved, except the general problem that many active senders will generate many packets (independent of whether coordinated or not).

There remains one problem, however. If more members desire to send than can be granted a token at any one time, a distributed queue

needs to be formed. To be able to sustain large queues of senders, the coordinator maintains a global damping factor  $d$  for token requests. A new value for  $d$  is distributed every heartbeat by the coordinator (as a rounded-down base 2 logarithm). In normal operation,  $d$  is 1. When requesting a token or retransmitting this

request, senders use the current value of  $1/d$  in each heartbeat as the probability for actually sending the request in this heartbeat. Senders echo the damping factor used in each token request actually sent. The coordinator weighs the token requests by their damping factor to allocate tokens. Piggy-backed token requests are considered to have a damping factor of one (no damping is applied to piggy-backed token requests).

The coordinator computes  $d$  as:

$$\max(1, w / (\max(12, k) * 2) - 1)$$

where  $w$  is the sum of the echoed damping factors received in token requests during the last heartbeat and  $k$  is an exponentially weighted moving average of the number of token requests granted in recent heartbeats.

### [3.3.](#) Atomicity

Atomicity (arrival of a message at all members or at none of the members) is a desirable property of a group communication protocol. Unfortunately, full atomicity requires collecting positive acknowledgements from all group members until a message can be acted upon, too heavy-weight for the goals of MTP/SO. Instead, MTP/SO defines a lighter-weight form of atomicity that is still useful for many applications.

At any point in time, each message is assigned a state by the coordinator: pending, accepted, or rejected.

The state of a message is set to accepted when the coordinator did receive the complete message. As soon as a sender notices one of its messages to be accepted, it sends an acknowledgement of successful transmission to its application. Such an acknowledgement does not mean that every receiver received the message. It only guarantees that at least the coordinator was able to receive it correctly. (It also provides the sequence number assigned to the message so that the application can order its own messages with respect to other messages it may have received).

A message marked as rejected was not completely received (even after requesting retransmissions) by the coordinator. Normally, every receiver will drop such a message and the sender of the message will indicate an unsuccessful-transmission error to its application.

Receivers do not deliver pending or rejected messages to the application. If a specific receiver does not completely receive a message (even after requesting retransmissions) that is finally

marked by the coordinator as accepted, it will signal this as an unsuccessful-reception error to its application.

In summary, it is guaranteed that a message was either delivered correctly to every receiver, that it was delivered to no receiver and the sender is signalled an error, or that any receiver that did not receive the message is signalled an error. (Of course, the protocol works hard to minimize the number of such errors, but the above statements are guarantees of the protocol.)

Atomicity increases the message latency: applications need to wait for the accepted state propagating from the coordinator before they can act on a message. In order to allow every member to quickly learn about the state of messages, every packet contains a copy of the most recent information available about the state of the most recent messages. If application semantics do not require atomicity, unnecessary delay can be avoided by marking a message such that it is delivered to applications even before accepted by the coordinator (`atomicity_off`).

#### 3.4. Retransmission

Receivers request retransmissions of data packets when there is a gap in the sequence numbers of data packets received for a message or if no further data packet has arrived for more than one heartbeat while the message is still incomplete. In case all data packets for a message have been lost, this will be recognized from the message state of packets from following messages or when the coordinator propagates the state of the most recent messages. In any case, the request for retransmission can be generated at the latest after two full heartbeats.

Retransmission requests, or NAKs (negative acknowledgements) for short, are multicast to the group to reduce the implosion problem. Receivers dither the time at which they send NAKs and postpone sending a NAK when they have recently received one or more NAKs that together cover the same set of packets.

In order to answer NAKs, senders keep a copy of every data packet they sent. To limit the number of packets stored, senders are allowed to discard these copies after a defined period of time which is measured in heartbeats and depending on a special factor called retention. After retention+4 heartbeats the copies are no longer available and requests for retransmissions received after that period are ignored. This makes sure packets are available for at least retention retransmissions.

Nonetheless there is a nonzero probability that all retransmissions



(or retransmission requests) related to a packet are lost and some receivers do not receive the message correctly. For example a network partitioning that lasts longer than  $\text{heartbeat} \times \text{retention}$  will result in lost messages.

This sounds undesirable, but it is similar to the retry limit used in positively acknowledged protocols, only that the normally relatively small value of  $\text{heartbeat} \times \text{retention}$  puts a limit to the length of an outage that can be tolerated. We assume that the application protocol will have a way to handle receivers that experience such a long gap in reception, because it already needs a way to treat new members that appear late in the group. Note that for applications where this is undesirable, MTP/SO could be augmented by the equivalent of log servers [3]. In any case, MTP/SO guarantees that when a message was not completely received by every receiver, either the affected receivers or the responsible sender will indicate the error to the application.

### [3.5.](#) Self-organization and Repetitors

Once MTP/SO groups get large, even the handling of NAK-based retransmission traffic becomes a scalability problem. As with many scaling problems, the obvious solution is to introduce some form of hierarchy into the group. This allows at least some of the NAKs and resulting retransmissions to be handled locally within trunks and branches of that hierarchy. As MTP/SO is a many-to-many protocol, it does not make much sense to base the hierarchy on the multicast tree from any specific sender (including the coordinator, which generally is not the sole sender and which may transfer its role to another member during the activity of the group).

Instead, MTP/SO introduces the concept of a regional repetitor\*. Receivers multicast NAKs locally before multicasting them to the entire group. Repetitors that have previously received the requested data, retransmit locally after receiving a local NAK. Repetitors that don't have the data simply send a NAK to the next higher level of hierarchy, up to the whole group (where, finally, the sender replies with another copy of the data).

A prerequisite to this mechanism is a way to do a local multicast (of a NAK as well as of a retransmission). In current IP multicast implementations, one way to define such regions is with TTL threshold scoping; together with an appropriate protocol; administrative scoping provides a similar method [7]. The algorithms described in

the rest of this section work best when such a scoping mechanism is in effect; leaks or other imperfections in the scoping boundaries do not cause catastrophic failures, though. The following discussion assumes three levels of local scopes, e.g., site, country, and continent; the exact choice of number and extent of scopes is a global parameter of the group.

With three local scopes and one global scope, each group member is by

-----  
\* This was called ``repeater'' in earlier presentations of MTP/SO [6]. We are now avoiding this term as it is sometimes used as an alternative term for a transport layer ``bridge'' between disconnected multicast domains.

definition in four scopes, where each local scope is contained by the next higher scope in the hierarchy. Any member that takes on a receiver role can also decide to be a potential repeter for any of the local scopes (e.g. depending on the cost structure of the Internet service or on the availability of local memory space).

For scopes that contain only one member, it does not matter whether a member considers itself to be a repeter for that scope or not. For scopes that contain more than one member, a protocol is needed that makes this fact known and selects one scope member as the repeter. This protocol needs not necessarily ensure that there is exactly one repeter for each scope at any time, as the retransmission protocol still works without a repeter or with more than one repeter per scope, albeit less efficient.

Repeter selection should favor the ``best'' member in the scope, i.e. a member that has particularly good reception from the senders, as it is most likely that this member will have received the data to be able to perform a local retransmission. Each potential repeter therefore maintains a reception quality parameter that, on a first level of approximation, tallies the quotient of the number of recently correctly received packets to the number of packets that should have been received.

Members that consider themselves repeter for a scope periodically multicast a repeter announcement message within the scope, containing the current value of the reception quality parameter. Potential repeters observe these messages. If, within the most local scope, a potential repeter has a considerably better reception quality parameter than the current repeter, it sends a

repetitor announcement at the start of its next heartbeat interval and assumes the role of the repetitor. Only the repetitors of the most local scope compete for the repetitor role of the next higher scope, and so on. (A new repetitor that displaces a member that was repetitor at higher level scopes also announces itself as repetitor at these higher level scopes.)

To better cope with repetitor failure, receivers that are not repetitors send NAKs at the most local scope first and escalate them up the hierarchy if neither a retransmission nor a more global NAK follows within one heartbeat. Repetitors for a set of scopes begin sending NAKs within the next higher scope and then escalate them the same way. Retransmissions always occur at the highest level of scope that the NAKs leading to that retransmission carried (NAKs have a scope field for this purpose).

A repetitor that leaves a group simply sends a repetitor announcement with reception quality zero. A repetitor that crashes stops sending repetitor announcements, causing potential repetitors to start sending repetitor announcements after a time interval that is inversely related to their reception quality parameter.

### [3.6.](#) Coordinator function

As it is responsible for assigning tokens and updating the message state, the coordinator plays a central role in MTP/SO. If the member carrying the coordinator function leaves the group, the coordinator function will be passed to one of the remaining members automatically.

To avoid the coordinator being a single point of failure, MTP/SO provides a coordinator recovery function. This allows the group to elect a new coordinator when the old one crashes or becomes unreachable. The new coordinator will then collect all information needed from the group members so that no information is lost. (This protocol should be, but is not yet, integrated with the repetitor function.)

In order to enhance the performance of MTP/SO it may be useful to actively influence which member performs the coordinator function. For example if only one member will send messages for a longer period of time, the group can migrate the coordinator function to that member, thereby avoiding the overhead caused by requesting and

obtaining tokens (between one and two packets for every message). MTP/SO allows either to request the coordinator function for oneself or the coordinator to pass the coordinator function to another member.

### 3.7. Membership classes

Not all members of the group will be in a position to take over the functions of a coordinator or of a repetitor. We therefore distinguish several ``classes'' of members:

class		description
-----+-----		
1		normal member, potential coordinator and repetitor
2		normal member, potential repetitor
3		normal member
4		unreliable receiver, normal sender
5		unreliable member

Most members of an MTP/SO group will be class 1 members, i.e. they are prepared to take over the coordinator role if this is required in a coordinator recovery\*. Class 2 members do not want to take on this role (for application reasons or for reasons of limited resources), but compete for the repetitor function. Class 3 members take over neither special function, but take part as normal members in the group; in particular, they are allowed to send NAKs.

-----  
\* Instead, a single member can be designated fixed coordinator by assuming class 0. This means that the multicast group shares its fate with the class 0 coordinator.

Class 4 members never send NAKs. Their reception of messages in the group is therefore unreliable. Nonetheless, they can originate messages that are reliably received by the class 3 or higher members of the group.

Class 5 members listen only; the only packet type they can send to the group is unreliable multicast datagrams (not yet described in this version of the draft). When a minimum quality of transmission/reception is defined for the group (see group[info] packets below), members may have to downgrade themselves to class 5 when they find out their own quality has dropped below the acceptable level.

To aid class 4 and class 5 members, and as a general optimization, the multicast group can be configured to immediately add a percentage of redundancy packets to the data packets sent. This allows the receivers to reconstruct missing data packets by interpreting these redundancy packets. Redundancy packets also can be independently added by repetitors based on the local NAK rate.

## [4.](#) Protocol Definition

### [4.1.](#) Notational Conventions

For convenience, the datagrams transmitted by MTP/SO group members are called packets in this document.

MTP/SO packet types are written major[minor], where major is the major type of the packet and minor is the subtype within the major type. E.g., there are data[data] packets as well as data[eom] packets.

### [4.2.](#) Protocol Functions and Packet Types

#### o Heartbeat

All members operate on a time line that is divided into heartbeats. The nominal length of a heartbeat is a global parameter of the group. The actual heartbeat boundaries (or heartbeats for short) are dithered around the nominal value. Most protocol actions are performed at the start of a new heartbeat interval. An exception is the actual transmission of data packets, which is evenly distributed over the heartbeat interval to which the data packets are allocated. Also, token requests (and token request cancellations) can be unicast to the coordinator and be responded to by the coordinator at any time.

#### o Global Ordering

A sender that wants to send a message applies for a token by unicasting a token[request] packet to the coordinator.

Alternatively, the sender can include a token request field in a data packet that is sent under a previously obtained token.

As soon as a token becomes available, the coordinator replies with a token[confirm] containing a new global sequence number, under consideration of the queue of token requests and the priority of the token request. The sender uses this global sequence number as the message number in every data packet pertaining to this message.

#### o Message Acceptance

The coordinator maintains the message acceptance state for recent messages. For the 12 most recent messages, the message acceptance state is disseminated in every packet. Packets sent by the coordinator contain the current message acceptance state; packets sent by other members contain a copy of the most recent message acceptance state available to that sender (for data packets, this is often the state obtained via the token[confirm] packet). As the field that is used to disseminate that state only has 12 entries, the number of messages that can be pending at any point in time is limited.

To ensure that the most recent message acceptance state is always disseminated, the coordinator sends a group[info] packet at least\* in every heartbeat in which no other member is scheduled to send packets based on tokens sent out.

#### o Retransmissions

At each heartbeat, receivers that are missing packets of a message multicast nak[request] packets (see also the discussion of self-organization and repetitors above). A nak[request] contains a list of ranges of sequence numbers for one or more messages. Ranges can be open, i.e. implicitly include all further packets when the ending packet number is not known. A nak[request] that is received by a receiver postpones sending a nak[request] for the set of packets listed in the nak[request]. Empty nak[request] packets are never sent.

### [4.3.](#) Addresses

A MTP/S0 group has one group address and as many member addresses as there are members.

The member address is the combination of a 128-bit IPv6 host address (possibly in IPv4 compatibility format, i.e. with 96 bits of leading zeroes) and a 16-bit UDP port number.

---

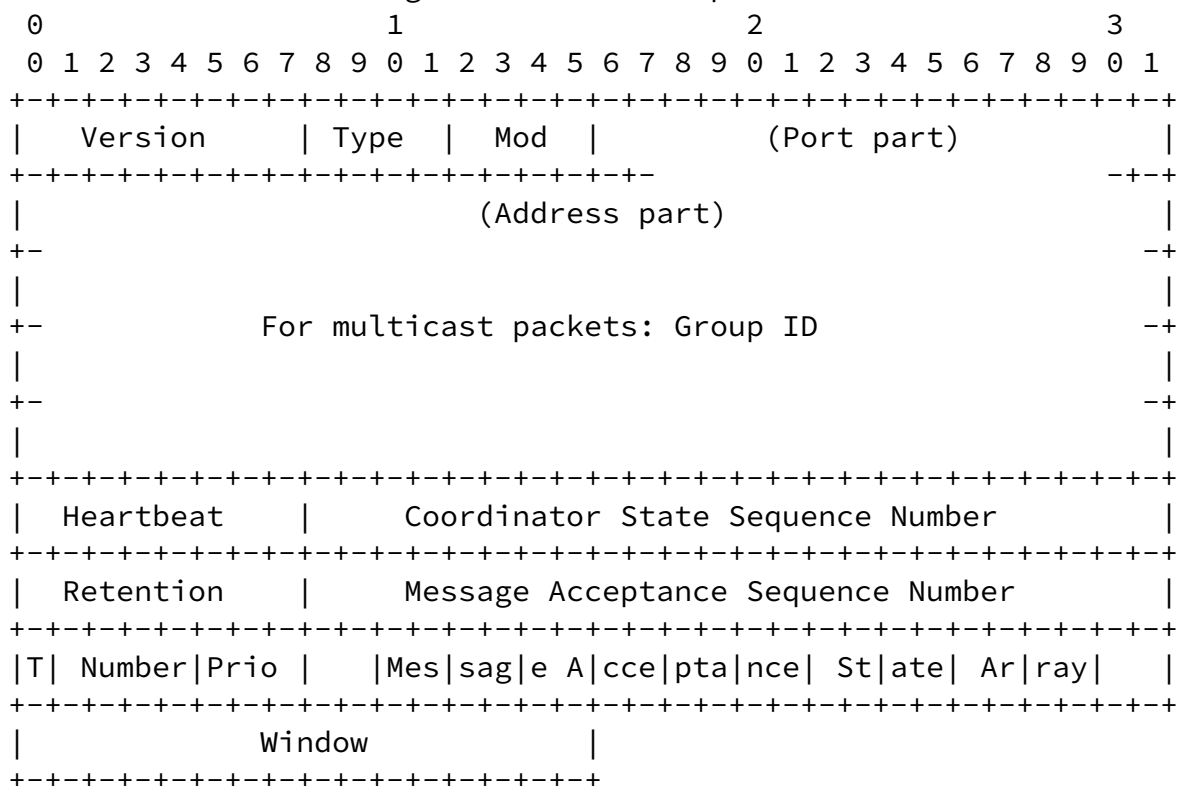
\* In periods of continuous activity, additional group[info] packets are sent at a reduced rate to allow unreliable receivers to join.

The group address is the pair of a 128-bit IPv6 multicast address (again, possibly IPv4 compatible) and a group-ID. The group-ID simply is the member address of the current coordinator.

MTP/SO multicasts always use the UDP destination port number 47112 (to be assigned) and the UDP source port number from the member address. MTP/SO unicasts use UDP source and destination port numbers in the range 47112+1 to 49152-1 (note that the number 49152 marks the end of the medium priority port number space in some current IP multicast router implementations).

#### [4.4.](#) Packet Formats

Figure 1: Standard packet header



The standard packet header contains the following fields:

- o Version

For the current version of MTP/SO, version is always 3.

- o Type, Mod

Packet type and type modifier (subtype).

- o Group ID

For multicast packets, this field gives the member address of the current coordinator. For unicast packets, this field is not used.

Bormann, Ott, Seifert

[Page 13]

---

INTERNET-DRAFT

MTP/SO: Self-Organizing Multicast

June 1999

- o Coordinator State Sequence Number

A sequence number for the version of the coordinator state that is disseminated with this message.

- o Message Acceptance Sequence Number, Message Acceptance State Array

Let  $n$  be the message acceptance sequence number, then message acceptance state array contains the most recent message acceptance states known for messages  $n-1$  to  $n-12$ :

- 0 pending
- 1 accepted
- 2 rejected
- 3 (reserved)

- o T, Number, Prio

If the T bit is set, Number gives the serial number and Prio the priority of a token request piggybacked in this packet.

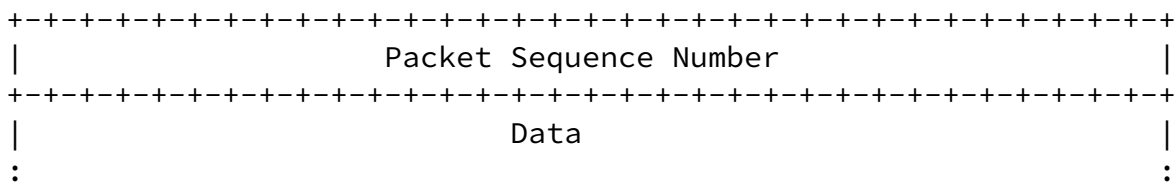
- o Heartbeat, Retention, Window

Current values for these three global parameters of the group. These parameters are given as pseudo-floating-point numbers:

parameter	bits	mantissa (msb)	exponent (lsb)	unit
heartbeat	8	3	5	microseconds (0 to 7
retention	8	4	4	1 (0 to 15
window	16	11	5	microseconds (0 to 2047

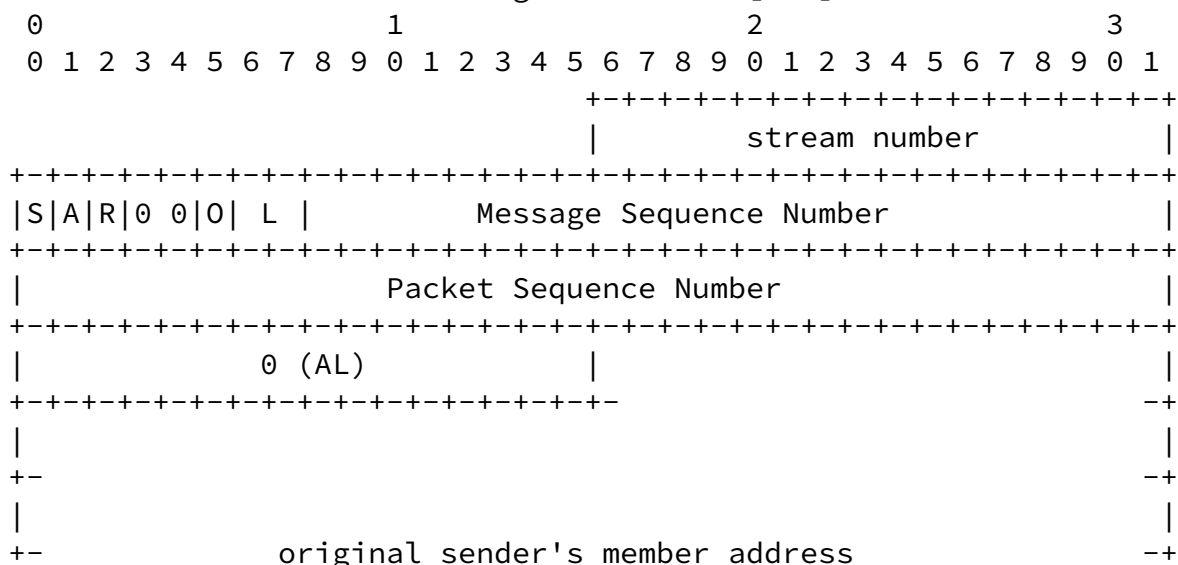


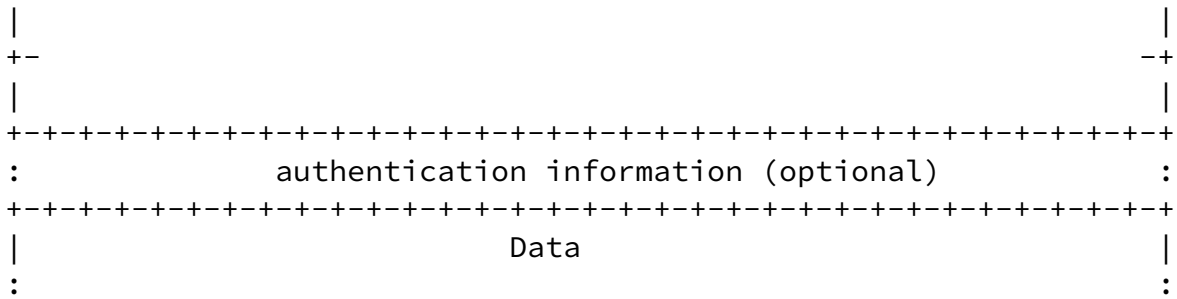




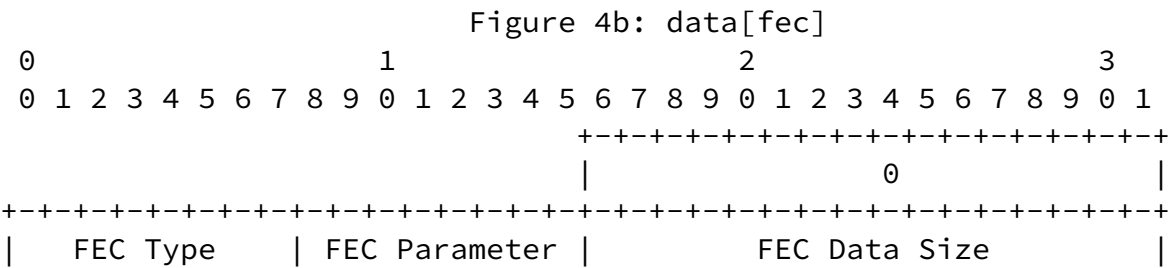
The S bit, if set, indicates that ordered delivery is not required for this message ('`sequencing\_off`). The A bit, if set, indicates that atomic delivery is not required for this message ('`atomicity\_off'). The R bit, if set, indicates that this message is not transmitted reliably, i.e., the producer is not going to answer any nak[request]s. Consumers are expected to wait for any missing packet of this message for one heartbeat and then mark the message as not received. The O bit ('`original') is set only for the first transmission of the data packet by the original sender. It is reset for any kind of retransmission (regardless whether performed by the original sender or not). L ('`level') is a binary number ranging from 0 to 3. Level 0 indicates a global transmission; levels 1 to 3 indicate transmission of the packets at the second most global to most local level scope, resp. (For a retransmission, the transmission level indicates the scope in which this data packet was sent; lower level repetitors can use this information to decide whether they can defer their own retransmissions.)

Figure 4a: data[eom]





To ensure that the original sender of a message becomes known even if the only packets a receiver has received from this message were repetitor retransmissions, the data[eom] packet differs from the other data packets in that it contains a copy of the original sender's member address. (Note that this information is redundant for packets that have the 0-bit set; it is retained in favor of a common packet format for all cases.) With an optional authentication protocol (not specified in this version of the document), authentication information can be given with this last packet of the message; the length in 32-bit words is in AL.



```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
+          FEC Data (including modified header)          +
:
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          0          |          Message Sequence Number          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Packet Sequence Number          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          0          |          Message Sequence Number          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Packet Sequence Number          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:

```

Senders and repetitors can send FEC packets in addition to (or instead of) data packets. A data[*fec*] packet contains forward error correction information computed out of one or more original data packets, including their headers, where the port part of the group address is replaced by a two-byte original packet length field and the rest of the group address is left out; these data packets are each identified by their message sequence number and packet sequence number. FEC data size is the total size of this information. The resulting information (starting at a two-byte boundary) is padded to a four-byte boundary. Only data packets with the same group address can be combined; they are sent with a copy of this group address in the header of the data[*fec*] packet. FEC type and parameter define the exact FEC code used; FEC type 1 is defined as XOR.

[illegible][illegible]

A status request packet can be multicast by a member to request status for messages that already have scrolled off the message acceptance state array in the standard header. A status deny response indicates that the retention time for keeping information about the status of the messages has passed.

The K-bit, if set, indicates that reliable receiver status (membership class 1 to 3) is intended, i.e., that an explicit acknowledgement for this member has to be given within a group[info]. The C-bit, if set, indicates that the transmitter is a potential coordinator (membership class 1); it causes other potential coordinators with a higher member address to back off. The scope field gives the actual scope in which this packet was transmitted (this cannot just be given as a scope level number as the actual scope levels used in this group may not yet be known to the transmitter).

```

0      1      2      3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +---+---+---+---+---+---+---+---+---+---+
                                |                                     Quality   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Activity                |          0           |    dlb    |E| L |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| TTL Scope 0     | TTL Scope 1     | TTL Scope 2     | TTL Scope 3     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Network Packet Size              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   min. Receive Quality       |   min. Send Quality         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Group Name Length        | Group Name ...                 :
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:                               (zero-padded to 4 byte alignment) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:   type      |   length      |             extension            :
:---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:                                                       :
:---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:   type      |   length      |             extension            :
:---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
:
:
```

The group[info] packet is periodically transmitted by the coordinator and by each repeditor to ensure that all group members are aware of the global parameters of the group and of the quality of the current repeditor.

The other fields of the packet give global group parameters that usually are constant: The E-Bit ('`elect'') is set for group[info] packets originated by the coordinator in case it is willing to transfer the coordinator function to a higher quality member; it requests other potential coordinators to announce their quality (if better) via group[info]. L gives the scope level, and, indirectly, the source of the group[info]: level 0 packets are originated by the coordinator or by other potential coordinators (the latter if the source address is not equal to the coordinator part of the group address), level 1 to 3 packets are originated by repetitors of the respective level. Analogously, the TTL fields provide the TTL scopes of the levels: TTL 0 is the scope of the entire group, TTL 1 to TTL 3

[Page 20]

June 1999

At the end of the fixed part of group[info] packets, extensions can be added. Their type is identified by a one-byte type code their length given by a one-byte length field, giving the number of 32-bit words beyond the initial one in this extension.

+++++-----					
:	1		4		(Port part)
+++++-----					
:	(Address part)				:
+--					--+
:	Acknowledged				:
+--	Member-Address				--+
:					:



```

+-                                         +-
:                                         :
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Type 1 group[info] extensions are used to carry an acknowledgement for a group[seek] requests by a member that needs to achieve reliable reception status quickly (K-bit in group[seek] set).

4.5. Summary of packet types

packet type	type[code]	multi/uni	sent by	see Figure
-----				
data[data]	0[0]	m	C,R,s	4
data[eom]	0[1]	m	C,R,s	4a
data[ceom]	0[3]	m	C,R,s	4*)
data[fec]	0[4]	m	C,R,s	4b
nak[request]	1[0]	mu	r	5
group[info]	2[0]	m	C,R	9
group[seek]	2[1]	m	C,R,s,r	8
token[request]	4[0]	u	s	2
token[confirm]	4[1]	u	C	3
token[cancel]	4[2]	u	s	3
status[request]	5[0]	m	C,R,s,r	6
status[info]	5[1]	m	C,R	7
coord[suspected]	6[0]	m	R,s,r	*)
coord[inforeq]	6[4]	u	p	*)
coord[info]	6[5]	u	C	*)

INTERNET-DRAFT	MTP/S0: Self-Organizing Multicast	June 1999
coord[statusreq]	6[6]	m
coord[status]	6[7]	m
coord[give]	6[8]	u
coord[accept]	6[9]	u

multi/uni: m is multicast, u is unicast.

sent by: C is coordinator (p is potential coordinator), R is repetitor, s is sender, r is receiver.

\*) Not yet described in the present version of the document.

5. References

- [1] S. Armstrong, A. Freier, K. Marzullo: ``Multicast Transport Protocol'', [RFC 1301](#), February 1992.
- [2] C. Bormann, J. Ott, H.-C. Gehrcke, T. Kerschhat and N. Seifert: ``MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport'', International Conference on Computer Communications and Networks (ICCCN 94), 1994 (available from <ftp://ftp.cs.tu-berlin.de/pub/local/kbs/mtp/doc/sero.ps>).
- [3] Holbrook, H.W., Singhal, S.K., and Cheriton, D.R., Log-based Receiver-Reliable Multicast for Distributed Interactive Simulation. SIGCOMM '95, Cambridge, MA, August, 1995.
- [4] N. Seifert, C. Bormann, J. Ott: MTP/SO: Self-Organizing Multicast, First Multicast-Workshop, GI/TU Braunschweig, May 1999.
- [5] S. Floyd, V. Jacobson, S. McCanne: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, SIGCOMM '95, Cambridge, MA, August, 1995.
- [6] C. Bormann, J. Ott, N. Seifert: MTP/SO: Receiver-Reliable Coordinated Many-to-Many Multicast, Presentation at the SIGCOMM 96 Workshop on Matters Mbone (``SIG-Bone''), Palo Alto; <http://www.cs.ucl.ac.uk/staff/jon/sigbone.html>, 27-August-1996.
- [7] R. Kermode: Scoped Address Discovery Protocol (SADP), November 1998, Internet-draft [draft-kermode-sadp-00.txt](#).

## 6. Authors' addresses

Carsten Bormann, Joerg Ott  
Universitaet Bremen FB3 TZI

Postfach 330440  
D-28334 Bremen, GERMANY  
cabo, jo@tzi.org  
phone +49.421.218-7024, 201-7028  
fax +49.421.218-7000

Nils Seifert, Joerg Ott  
Telligence GmbH  
Gustav-Meyer-Allee 25, Haus 12  
D-13355 Berlin, GERMANY  
se, jo@telligence.de  
phone +49.30.46307-551, -550  
fax +49.30.46307-579