P_Mul: 
An Application Protocol for the Transfer of Messages 
over Multicast Subnetworks and under EMCON Restrictions 

(draft-riechmann-multicast-mail-00.txt)

Abstract

P_Mul is a proposal for an application protocol for the transfer of messages using a connectionless transport protocol. The objectives of this protocol are first to take advantage of the bandwidth saving feature of using the multicast service as supported by modern computer networks and second to allow message transfer under EMCON conditions. EMCON (Emission Control) means that, although receiving nodes are able to receive messages, they are not able to acknowledge the receipt of messages.

The protocol described below has already been applied to the transfer of messages between X.400 Message Transfer Agents (MTA). But it is not only restricted to this application and can easily be applied to the transfer of RFC822 messages.
1. Introduction

The objective of this protocol is to take advantage of multicast communication for the transfer of messages between MTAs (Message Transfer Agents) on a single multicast network under normal - which means dialogue oriented - communication conditions and under EMCON conditions. EMCON condition means that a receiving node is able to receive messages, but it cannot - for a relative long time (hours or even days) - acknowledge the received messages.

Figure 1 illustrates a simple multicast scenario, where the same message has to be sent from MTA 1 to MTA 2 and to MTA 3.

```
+-------+
| MTA 1 |
+-------+<---| Router|<
         +-------+ \   +-------+
         | MTA 2 |<---->| MTA 3 |
         +-------+       +-------+
```

Figure 1: Typical MTA Configuration

Using a multicast instead of an unicast communication service in the above MTA configuration only one message transmission from MTA 1 to
the Router is required, instead of two as required with unicast. This saves the transmission of one message and thus network bandwidth utilisation. Depending on the network bandwidth (in some radio networks less than 9.6 Kb/s) this saving can be of vital importance. The saving in bandwidth utilisation becomes even greater with every additional receiving MTA.

As P_Mul employs a connectionless transport protocol to transmit messages, the reliable message transfer is guaranteed even in those cases, when for a certain period of time one or more of the receiving MTAs are not able or allowed to acknowledge completely received messages.

The protocol P_Mul has been applied to the communication between X.400 MTAs; but it is not constrained to X.400 MTA communication, it also can be applied to other kinds of reliable multicast transmission, e.g. file transfer or RFC822 messages.

This protocol specification requires fixed multicast groups and a well known knowledge at each participating node (MTA) about the group memberships to one or more multicast groups of each participating node.

2. Protocol Layering

As mentioned above this protocol has to be understood as an application layer protocol, in the same way as the P1 protocol defined in the X.400 Recommendations [CCITT88]. As in the case of P1, P_Mul will use the lower layer protocols to transmit its PDUs over the multicast network.

Considering the fact that MTAs under EMCON are not allowed to exchange any messages with their partner MTAs, it is not possible to use a reliable transport protocol like "RMP" [RMP95-1, RMP95-2, RMP95-3] or "XTP" [XTP95] for the transfer of messages. Therefore P_Mul will be based on the unreliable connectionless transport protocol "UDP" [UDP80]. A proposal as to which UDP ports shall be used can be found within the Appendix.

In the X.400 context P_Mul is to be implemented within an MTA. A sample implementation has been implemented using the X.400 Message Handling System package available from ISODE Ltd. In this case two channel programs "Multicast_OUT" and "Multicast_IN" have been implemented and integrated. "Multicast_OUT" handles the sending of messages to the multicast connected neighbour MTAs. "Multicast_IN" handles the incoming messages from the multicast connected neighbour MTAs. These channels play the role of the 'Higher Management
Functions’ mentioned in following chapters.

3. Protocol Data Units (PDU)

The number of PDUs used by P_Mul is mainly determined by the fact that this protocol should also be suited for communication under EMCON condition, which is understood as only one sided communication. In this case several nodes are able to receive data and not able or allowed to acknowledge them. Such a restriction can exist for a long period of time (e.g. hours, days). This restriction has resulted in, that P_Mul uses only four different PDUs ("Address_PDU", "Data_PDU", "ACK_PDU" and "Discard_Message_PDU"). Other PDUs for instance as defined for RMP like "Recovery Packet" or "Alarm Packet" and others are not useful in this context.

All PDUs are sent using the UDP protocol and multicast addressing.

3.1. Address_PDU

This PDU and the ACK_PDU (see below) govern the flow control of P_Mul packets. As P_Mul has to observe a maximum PDU_size and as the number of Destination_Entries has no maximum value limitation, it is necessary to be able to split the total addressing information into more than one Address_PDU. To distinguish between the first, middle, or last Address_PDU the MAP field ("More Address_PDUs") is used.
Destinations:

+-------------------------------+-------------------------------+
| Count_of_Destination_Entries  |      Length_of_DES_Key        |
+---------------------------------------------------------------+
|          List_of_Destination_Entries (variable length)        |
|                                                               |
|                                                               |
|                                                               |
+---------------------------------------------------------------+

One Destination_Entry:

+---------------------------------------------------------------+
|                        Destination_ID                         |
+---------------------------------------------------------------+
|                    Message_Sequence_Number                    |
+---------------------------------------------------------------+
|                   DES_Key (variable length)                   |
|                                                               |
|                                                               |
|                                                               |
+---------------------------------------------------------------+

Explanations

Length_of_PDU:
The field (2 octets) indicates the total number of octets within this PDU.

Priority:
This 1-octet field enables the sending MTA to inform the other MTAs that at the sending MTA there exists at least one message with the denoted priority (for further study). The value 0 denotes the highest priority.

MAP: This 2-bit field specifies whether this Address_PDU is a first or a middle or a last one.

The high order bit is set to
‘0’: This is the first one of a set of Address_PDUs.
‘1’: This is NOT the first one of a set of Address_PDUs.
The low order bit is set to
‘0’: This is the last one of a set of Address_PDUs.
‘1’: This is NOT the last one of a set of Address_PDUs.

In the case both bits are set to zero, only one Address_PDU exists.

PDU_Type:
This 6-bit field specifies the type of the actual PDU. PDU_Type x'02' denotes an Address_PDU.

Total_Number_of_PDUs:
These 2 octets hold the total number of Data_PDUs of the message.

Checksum:
The checksum will be built by using one of the currently available checksum algorithms and is calculated over the rest of this PDU. A sample implementation of the Fletcher algorithm can be found in the Appendix.

Source_ID:
This field holds the ID of the sender of this Address_PDU. This ID must be unique for the actual multicast network. As an example the Internet address of the specific multicast interface of the sending MTA node could be used.

Message_ID:
MSID is a unique identifier built within the scope of Source_ID by the transmitter (e.g. seconds since 00:00:00 1.1.1970 - Unix Time -).

Expiry_Time:
This is the expiry time of the message. This value is to be set as a number of seconds since 00:00:00 1.1.1970 - Unix Time -, which is derived from the Latest_Delivery_Time of the corresponding X.400 message and/or by a predefined value. This entry gives the chance for transmitters and receivers of a message to decide, when a message entry is outdated and can be removed from the processing lists.

Destinations:
This is a variable length field containing all receivers, their message sequence information and as an option, if confidentiality has been implemented and is needed, a randomly chosen DES_Key, which is encrypted with the public key of each receiving site.
Count_of_Destination_Entries:
The first 2 octets of this "Destinations" substructure of the PDU denote the number of destination entries. Each is formed by the Destination_ID, a message sequence number and eventually an DES_Key being encoded by the public key owned by this Destination_ID designated receiver.

Length_of_DES_Key:
These 2 octets specify the length of the public key encoded "DES_Key", which is 0, if confidentiality is not used.

List_of_Destination_Entries:
This entry is an array of destination entries. Its dimension is specified within the above mentioned "Count_of_Destination_Entries".

Destination_ID:
This field holds an identifier uniquely identifying a receiving MTA on the actual multicast network. As an example the Internet address of the receiving MTA node could be used.

Message_Sequence_Number:
This entry holds a message sequence number, which is unique for the sender/receiver pair denoted by Source_ID and Destination_ID. This sequence number will be generated by transmitter consecutively with no leaks and is used by each receiver to detect message loss.

DES_Key:
In case of confidential transfer of Data_PDUs the data portions will be encoded by a symmetric encryption algorithm. The symmetric key, which is the same for all receiving MTAs, will be asymmetrically encoded by the public key of each receiving MTA and transmitted within this field. Each receiving MTA can decode its "own" key information by using its private key. The key for the symmetric encryption should be valid for the total transmission of one message and will be generated randomly.

3.2. Data_PDU

This PDU holds essential information such as the unique identifier of the message, the position of this Data_PDU within the ordered set of all Data_PDUs and one fragment of the total message.
### Explanations

**Length_of_PDU:**
- The field (2 octets) indicates the total number of octets within this PDU.

**Priority:**
- This 1-octet field enables the sending MTA to inform the other MTAs that at the sending MTA there exists at least one message with the denoted priority (for further study). The value 0 denotes the highest priority.

**PDU_Type:**
- This 6-bit field specifies the type of the actual PDU. PDU_Type x'00' denotes a Data_PDU.

**Number_of_PDU:**
- This value offers the information where to place this message fragment into the whole message.

**Checksum:**
- The checksum will be built by using one of the currently available checksum algorithms and is calculated over the rest of this PDU. A sample implementation of the Fletcher algorithm can be found in the Appendix.

**Source_ID:**
- This field holds the ID of the sender of this Data_PDU and is equivalent to the Source_ID within the corresponding Address_PDU. This ID must be unique for the actual multicast
network. As an example the Internet address of the specific multicast interface of the sending MTA node could be used.

Message_ID:
MSID is a unique identifier built within the scope of Source_ID by Multicast_OUT (e.g. seconds since 00:00:00 1.1.1970 - Unix Time).

Fragment
This portion of the Data_PDU holds the message or only fragments of it.

3.3. Discard_Message_PDU

This PDU is used to inform the receiving MTAs, that the transfer of a specific message has been terminated and no further PDU of this message will be transmitted. Such a situation can occur because of different reasons like hardware error or message outdating. Those PDUs already received shall be forgotten by the receiving MTAs. In this case the sending MTA shall generate a Non-Delivery Report.

```
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
+-------------------------------+---------------+---+-----------+
|         Length_of_PDU         |   Priority    |   | PDU_Type  |
+-------------------------------+---------------+---+-----------+
|           not_used            |            Checksum           |
+-------------------------------+-------------------------------+
|                           Source_ID                           |
+---------------------------------------------------------------+
|                       Message_ID (MSID)                       |
+---------------------------------------------------------------+
```

Explanations

Length_of_PDU:
The field (2 octets) indicates the total number of octets within this PDU.

Priority:
This 1-octet field enables the sending MTA to inform the other MTAs that at the sending MTA there exists at least one message with the denoted priority (for further study). The value 0 denotes the highest priority.

PDU_Type:
This 6-bit field specifies the type of the actual PDU. PDU_Type x’03’ denotes a Discard_Message_PDU.

not_used:
This field is currently not used (reserved for future use).

Checksum:
The checksum will be built by using one of the currently available checksum algorithms and is calculated over the rest of this PDU. A sample implementation of the Fletcher algorithm can be found in the Appendix.

Source_ID:
This field holds the ID of the sender of this Discard_Message_PDU and is equivalent to the Source_ID within the corresponding Address_PDU. This ID must be unique for the actual multicast network. As an example the Internet address of the specific multicast interface of the sending MTA node could be used.

Message_ID:
MSID is a unique identifier built within the scope of Source_ID by the transmitter (e.g. seconds since 00:00:00 1.1.1970 - Unix Time -).

3.4. ACK_PDU

This PDU is generated by a receiving MTA identified by the Source_ID_of_ACK_Sender and is used to inform the sending MTA about the receive status of one or more messages. This information is composed within one or more entries of the List_of_ACK_Info_Entries. Each of these entries holds a message identifier (Source_ID and Message_ID) and a List_of_Missing_Data_PDUs, which may contain a list of those Data_PDUs not yet received. If this list is empty, the message identified by Source_ID and Message_ID has been completely received by this receiving MTA.
Explanations

Length_of_PDU:
The field (2 octets) indicates the total number of octets within this PDU.

Priority:
This 1-octet field enables the sending MTA to inform the other MTAs that at the sending MTA there exists at least one message with the denoted priority (for further study). The value 0 denotes the highest priority.

PDU_Type:
This 6-bit field specifies the type of the actual PDU. PDU_Type x’01’ denotes an ACK_PDU.

not_used:
This field is currently not used (reserved for future use).

Checksum:
The checksum will be built by using one of the currently available checksum algorithms and is calculated over the rest of this PDU. A sample implementation of the Fletcher algorithm can be found in the Appendix.

Source_ID_of_ACK_Sender:
This field holds the ID of the sender of this ACK_PDU and is equivalent to one of the Destination_IDs described for the Address_PDU. This ID must be unique for the actual multicast network. As an example the Internet address of the specific multicast interface of the ACK_PDU sending MTA node could be used.

Dest_ACK_Info:
This is a variable field containing ACK_Info_Entries for one or more message sending nodes on the multicast network. Each ACK_Info_Entry consists of the corresponding Source_ID of the message sending node, the message_ID and the list of those Data_PDUs not yet received by the ACK_PDU generating receiver.

Count_of_ACK_Info_Entries:
This field contains the number of ACK_Info_Entries. As long as there is only one active sender, we have only one ACK_Info_Entry, but it is assumed, that we can have more than one message sending node on the same multicast network.

Length_of_ACK_Info_Entry:
This field holds the octet length of each ACK_Info_Entry.
Having this field, the List_of_Missing_Data_PDUs is adaptable (in length) to the quality of the transmission (see "List_of_Missing_Data_PDUs" below). The maximum number of entries within this list is defined as

\[
M = \frac{\text{Length_of_ACK_Info_Entry} - 8}{2}.
\]

**List_of_ACK_Info_Entries**

This entry is an array of ACK_Info_Entries. Its dimension is specified within "Count_of_ACK_Info_Entries".

**Source_ID:**

This field holds the identifier of the message sending node. It is equivalent to "Source_ID" within the description of the Address_PDU, Data_PDU or Discard_Message_PDU.

**Message_ID:**

MSID is a unique identifier built by the sender (e.g. seconds since 00:00:00 1.1.1970 - Unix Time -). It is equivalent to "Message_ID" of the description of the Address_PDU.

**List_of_Missing_Data_PDUs:**

This field contains the list of the order numbers of those Data_PDUs definitely sent by the sender but not yet received at the receiver generating the current ACK_PDU. The maximum number of entries in this list is defined by M (see "Length_of_ACK_Info_Entry" above).

If the list holds less than M entries, then the first free entry has to be set to 0 to mark the end of the list. If all Data_PDUs have been acknowledged, the first entry has to be set to 0.

Whereas normally an ACK_PDU will only signal the receiving status ("ACK") of one MSID of Destination_ID, especially at the end of an EMCON period, one ACK_PDU may contain more "ACKs" to one or more MSIDs of one or more Destination_IDs.

### 4. Messaging Procedures

The following describes the messaging procedures to be employed between a transmitting MTA and one or more receiving MTAs either in the non-EMCON or EMCON mode of operation using the P_Mul multicast protocol. The following procedures shall be implemented for every message transmitted. The procedures are divided into three main areas, namely:

* Transmission and Re-transmission
* Reception
* Acknowledgement

4.1. Transmission and Re-transmission of a Message

An MTA shall initiate the transmission of a message by firstly transmitting an Address_PDU containing a list of all the MTAs that are to receive the message. The transmitting MTA shall be made aware, by a 'Higher Management Function', which of the receiving MTAs are in the non-EMCON and EMCON mode of operation. The transmitting MTA shall, based on the supplied mode of operation information, create a list of non-EMCON receiving MTAs, from which it shall expect to receive ACK_PDUs.

On completion of transmission of the Address_PDU the transmitting MTA shall proceed with the transmission of the Data_PDU(s), with the Message_ID field set equal to the Message_ID field of the associated Address_PDU. On completion of transmission of the last Data_PDU of a message the transmitting MTA shall initialise a "Transmitter Expiry_Time Timer" and an "EMCON Re-transmission Counter" (EMCON_RTC), if one or more of the receiving MTAs is in the EMCON mode, and enter the non-EMCON or EMCON Re-transmission mode (see para. 4.1.3 resp. 4.1.4).

4.1.1. Transmitter Expiry_Time Timer

This timer indicates the time remaining before the contents of the transmit message is considered invalid. It shall be initialised in accordance with the value transmitted in the Expiry_Time field of the Address_PDU.

If, in the event that one or more of the receiving MTAs have not acknowledged receipt of the complete message, this timer expires, the transmitting MTA shall transmit a Discard_Message_PDU with the Message_ID field set equal to the Message_ID field in the associated Address_PDU and Data_PDUs.

If, after having transmitted a Discard_Message_PDU, the transmitting MTA receives an ACK_PDU indicating only partial receipt of the message, it shall re-transmit the Discard_Message_PDU.

If, after having transmitted a Discard_Message_PDU, the transmitting MTA receives an ACK_PDU indicating receipt of the complete message, it shall disregard the ACK_PDU.

If all the receiving MTAs addressed in the Destination part of the Address_PDU acknowledge the receipt of the complete message before
this timer expires, the timer shall be stopped.

4.1.2. EMCON Re-transmission Counter

This counter (EMCON_RTC) indicates the maximum number of times the complete message may be re-transmitted while in the EMCON Re-transmission mode. It shall be set to a value based on such parameters as the priority, etc. of the message.

4.1.3. Non-EMCON Re-transmission

The transmitting MTA shall enter this mode when one or more of the receiving MTAs is in the non-EMCON mode. On entering this mode the transmitting MTA shall either:

a. having transmitted the last Data_PDU, initialise the "Ack Re-transmission Timer" or

b. having left the "EMCON Re-transmission" mode, re-transmit all the indicated missing Data_PDUs and then initialise the "Ack Re-transmission Timer".

4.1.3.1. Ack Re-transmission Timer

This timer shall be initialised when one or more of the receiving MTAs are in the non-EMCON mode. The duration of this timer shall be derived from the maximum round trip delay plus a safeguard.

If, in the event all the receiving MTAs in the non-EMCON mode respond with an ACK_PDU before this timer expires, the transmitting MTA shall either:

a. in the event that one or more of the ACK_PDUs indicate that a receiving MTA is missing Data_PDUs, re-transmit all the indicated missing Data_PDUs accompanied by a possibly updated Address_PDU and re-initialise the timer and remain in the "Non-EMCON Re-transmission" mode or

b. in the event that all the ACK_PDUs indicate that the complete message has been received, stop the timer. If, at this stage there are receiving MTAs in the EMCON mode the transmitting MTA shall enter the EMCON Re-transmission mode.

If one or more of the receiving MTAs in the non-EMCON mode does not respond with an ACK_PDU indicating partial or complete reception of the message, this timer expires, the transmitting MTA shall either:

a. in the event that it has not received a previous ACK_PDU from
the receiving MTA, re-transmit the complete message and re-initialise the timer or

b. in the event that it has received a previous ACK_PDU from the receiving MTA, only re-transmit the possibly updated Address_PDU and all the previously indicated missing Data_PDUs and re-initialise the timer.

4.1.3.2. Receipt of Message Complete ACK_PDU

When a transmitting MTA receives an ACK_PDU from a non-EMCON or EMCON receiving MTA indicating that it has received the complete message, it shall transmit an acknowledgement to this message complete ACK_PDU. The acknowledgement shall be done in the form of an Address_PDU with the destination address of the associated receiving MTA removed from the List_of_Destination_Entries field. Any subsequent re-transmission of the relevant message will be sent with the Destination_Entry of the receiving MTA removed from the List_of_Destination_Entries field in the Address_PDU.

4.1.4. EMCON Re-transmission

The transmitting MTA shall enter this mode as soon as all receiving MTAs are in the EMCON mode. On entering this mode the transmitting MTA shall initialise the "EMCON Re-transmission Timer" (EMCON_RTI).

4.1.4.1. EMCON Re-transmission Timer

This timer (EMCON_RTI) shall be initialised when all the receiving MTAs are in the EMCON mode. The duration of this timer will be based on such parameters as the priority, etc. of the message.

If this timer expires, which means that since the last EMCON Re-transmission none of the receiving MTAs in the EMCON mode has entered the non-EMCON mode - by sending one or more ACK_PDUs -, and the "EMCON Re-transmission Counter" has not reached its maximum count, the transmitting MTA shall re-transmit the Address_PDU and all Data_PDUs not already acknowledged. It shall re-initialise this timer and increment the "EMCON Re-transmission Counter" and the message sending MTA shall wait until either:

a. one or more of the receiving MTAs in EMCON mode respond with an ACK_PDU at which point it shall enter the "Non-EMCON Re-transmission" mode or

b. the "Transmitter Expiry_Time Timer" expires at which point it shall transmit a Discard_Message_PDU.
If, in the event one or more of the receiving MTAs in the EMCON mode responds, by entering the non-EMCON mode, with an ACK_PDU indicating partial or complete reception of the message, before this timer expires, the transmitting MTA shall stop the timer and enter the "Non-EMCON Re-transmission" mode (see para. 4.1.3).

4.2. Reception of a Message

The "Reception of a Message" mode shall be initiated when the receiving MTA receives either an Address_PDU or Data_PDU.

4.2.1. Receipt of an Address_PDU

On receipt of an Address_PDU the receiving MTA shall first check whether the Address_PDU has already been received.

If the Address_PDU has already been received the receiving MTA shall:

a. if its own ID exists in the List_of_Destination_Entries and it has previously sent a message complete ACK_PDU for this message, re-transmit the message complete ACK_PDU and discard the Address_PDU else

b. discard the Address_PDU.

If the Address_PDU has not been previously received the receiving MTA shall:

a. if its own ID does not exist in the List_of_Destination_Entries, check whether it has previously received any Data_PDUs associated with this Address_PDU (i.e. same Source_ID and Message_ID) and discard the Address_PDU. If there exists Data_PDUs associated with this Address_PDU, the receiving MTA shall discard these Data_PDUs.

b. if its own ID exists in the List_of_Destination_Entries determine whether it has previously received any Data_PDUs associated with this Address_PDU.

If there exists no Data_PDUs associated with this Address_PDU, the receiving MTA shall create a message entry and await reception of the associated Data_PDUs.

If there exists Data_PDUs associated with this Address_PDU, the receiving MTA shall update the status of the Data_PDU entry (see para. 4.2.2) to a message entry. In addition the receiving MTA shall stop the "Delete-Data_PDUs Timer" (see para. 4.2.2.1) and initialise the "Receiver Expiry_Time Timer". The receiving MTA shall then
determine whether to enter the "Acknowledgement of a Message" mode (see para. 4.2.3).

4.2.1.1. Receiver Expiry_Time Timer

This timer indicates the time remaining before the contents of the received message is considered invalid. It shall be initialised in accordance with the value received in the Expiry_Time field of the Address_PDU.

If this timer expires and the receiving MTA has not received all the Data_PDUs associated with a message, the receiving MTA shall discard the associated Data_PDUs and Address_PDU.

4.2.2. Receipt of a Data_PDU

On receipt of a Data_PDU the receiving MTA shall first check whether the Data_PDU has already been received.

If the Data_PDU has already been received the receiving MTA shall:

a. if the receiving MTA has not received a duplicate of the associated Address_PDU (see para. 4.2.1) and it has previously sent a message complete ACK_PDU for this message, re-transmit the message complete ACK_PDU and discard the Data_PDU else

b. discard the Data_PDU.

If the Data_PDU has not been previously received, the receiving MTA shall check whether it has received the associated Address_PDU.

If the associated Address_PDU has been received and a message entry exists the receiving MTA shall update the status of the message entry and determine whether to enter the "Acknowledgement of a Message" mode (see para. 4.2.3). If the associated Address_PDU has been received but no message entry exists the receiving MTA shall discard the Data_PDU.

If the associated Address_PDU has not been received the receiving MTA shall check whether there exists a Data_PDU entry associated with the Source_ID and Message_ID contents of the received Data_PDU.

If there exists no Data_PDU entry associated with this Data_PDU, the receiving MTA shall create a Data_PDU entry and await reception of the associated Address_PDU. In addition the receiving MTA shall initialise a "Delete Data_PDUs Timer". If there exists a Data_PDU entry associated with this Data_PDU, the receiving MTA shall update the status of the Data_PDU entry and await reception of the
associated Address_PDU.

4.2.2.1. Delete Data_PDUs Timer

This timer indicates the time remaining before the Data_PDUs in a Data_PDU entry are considered to be no longer valid and can therefore be discarded. It shall be initialised to a value based on such parameters as TBD.

If, in the event the receiving MTA does not receive the Address_PDU or a Discard_Message_PDU associated with the Data_PDUs in the Data_PDU entry, this timer expires, the receiving MTA shall discard all Data_PDUs.

4.2.3. Entry to "Acknowledgement of a Message"

Having updated the status of a message entry, the receiving MTA shall:

a. if in the non-EMCON mode, check whether the last Data_PDU of the message has been received or there are M (see para. 3.4) missing Data_PDUs. In the event that either is true the receiving MTA shall enter the "Acknowledgement of a Message" mode (see para. 4.3). In the event neither of the above is true the receiving MTA shall remain in the "Reception of a Message" mode or

b. if in the EMCON mode, check whether all the Data_PDUs for the message have been received. If there are missing Data_PDUs it shall remain in the "Reception of a Message" mode. If all the Data_PDUs have been received the message shall be tagged as complete and ready for acknowledgement when the non-EMCON mode is entered. The completed message can then be passed up to the 'Higher Layer Application'. The receiving MTA shall remain in the "Reception of a Message" mode.

4.2.4. Receipt of a Discard_Message_PDU

In the event that a receiving MTA receives a Discard_Message_PDU, it shall discard all the PDUs (i.e. data and address) associated with the message identified by the combination of the Source_ID and Message_ID fields in the Discard_Message_PDU. It shall also stop any associated timers.

4.3. Acknowledgement of a Message

The "Acknowledgement of a Message" mode shall only be entered by a receiving MTA that is in the non-EMCON mode of operation. The "Acknowledgement of a Message" mode procedures shall be dependent on
whether the receiving MTA received the messages in the non-EMCON mode, (see para. 4.3.1) or in the EMCON mode, (see para. 4.3.2).

To avoid the problem of ACK implosion on the message transmitting site, in the event that the receiving MTA has several ACK_PDUs to transmit, each transmission of an ACK_PDU should be delayed by a randomly determined period of time interval.

4.3.1. Receiving MTA in Non-EMCON

A receiving MTA shall, if operating in the non-EMCON mode enter the "Acknowledgement of a Message" mode when either the last Data_PDU of a message is received or there are M (see para. 3.4) missing Data_PDUs.

4.3.1.1. Last Data_PDU Received

If the last Data_PDU of a message has been received, the receiving MTA shall determine whether there are any missing Data_PDUs in the message.

4.3.1.2. Missing Data_PDUs

If there are missing Data_PDUs the receiving MTA shall:

a. if no ACK_PDU associated with this message has been previously transmitted, transmit an ACK_PDU listing which Data_PDUs are missing and return to the "Reception of a Message" mode or

b. if an ACK_PDU associated with this message had been previously transmitted, the receiving MTA shall return to the "Reception of a Message" mode.

4.3.1.3. Received Complete Message

If there are no missing Data_PDUs the receiving MTA shall transmit an ACK_PDU indicating that the message is complete. In addition it shall stop the "Receiver Expiry_Time Timer" associated with this message. The complete message shall be passed up to the 'Higher Layer Application'. On completion of transmission of the ACK_PDU the receiving MTA shall return to the "Reception of a Message" mode.

4.3.1.4. M Missing Data_PDUs

M has been defined as the maximum number of entries within the List_of_Missing_Data_PDUs (see para. 3.4). It is possible for a message to consists of more than M Data_PDUs, therefore it may be possible for the receiving MTA to be missing M Data_PDUs, without
having received the last Data_PDU, therefore:

A receiving MTA in the non-EMCON mode shall in the event that it is missing M Data_PDUs from a message, construct and transmit an ACK_PDU listing the M missing Data_PDUs.

If the receiving MTA proceeds to miss a further set of M Data_PDUs, it shall transmit a new ACK_PDU listing the further set of M missing Data_PDUs.

The receiving MTA shall repeat the procedure of transmitting an ACK_PDU for every further set of M missing Data_PDUs until the last Data_PDU is received.

On completion of transmission of every ACK_PDU the receiving MTA shall return to the "Reception of a Message" mode.

4.3.2. Receiving MTA leaving EMCON

If the receiving MTA is operating in the EMCON mode, it shall enter the "Acknowledgement of a Message" mode, when its mode of operation has changed to the non-EMCON mode. Upon entering the non-EMCON mode it shall determine whether it has received the last Data_PDU of a message.

Note: The following procedures are relevant only for those messages (complete or partial) received while in the EMCON mode. All new messages received before the change into the EMCON mode or after the change to the non-EMCON mode of operation shall be acknowledged in accordance with the procedures described in para. 4.3.1.

4.3.2.1. Last Data_PDU Received

If the last Data_PDU of a message has been received, the receiving MTA shall determine whether there are any missing Data_PDUs in the message.

4.3.2.2. Missing Data_PDUs

If there are missing Data_PDUs, the receiving MTA shall transmit an ACK_PDU listing denoting, which Data_PDUs are missing, and initialise an associated "ACK_PDU Timer" (see para. 4.3.2.4).

If there are N missing Data_PDUs and N > M the receiving MTA shall transmit \(\text{entier}((N-1)/M) + 1\) ACK_PDUs in order to indicate the total number of missing Data_PDUs. No ACK_PDU shall indicate more than M missing Data_PDUs, (see para. 4.3.1.4). The receiving MTA shall initialise an associated "ACK_PDU Timer" with each ACK_PDU.
On completion of transmission of the ACK_PDU(s) the receiving MTA shall then return to the "Reception of a Message" mode.

4.3.2.3. Received Complete Message

If there are no missing Data_PDUs, (e.g. tagged as complete, see para. 4.2.3), the receiving MTA shall transmit an ACK_PDU indicating that the message is complete and initialise the "ACK_PDU Timer". The receiving MTA shall, if it has not already done so, stop the "Receiver_Expiry_Time Timer" associated with this message and pass the complete message to the ‘Higher Layer Application’. The receiving MTA shall return to the "Reception of a Message" mode.

4.3.2.4. ACK_PDU Timer

The "ACK_PDU Timer" shall be initialised for every ACK_PDU transmitted by a receiving MTA, in the non-EMCON mode, having previously been in the EMCON mode. The duration of the timer shall be derived from the maximum round trip delay plus a safeguard.

If the receiving MTA receives a response to a transmitted ACK_PDU from the transmitting MTA in the form of the requested missing Data_PDUs or an Address_PDU, it shall stop the associated "ACK_PDU Timer".

If, in the event the receiving MTA does not receive a response to the transmitted ACK_PDU(s) from the transmitting MTA in the form of the requested missing Data_PDUs or an Address_PDU, an "ACK_PDU Timer" expires, the receiving MTA shall re-transmit the associated ACK_PDU(s) and re-initialise the timer.

5. Example

The following example shall explain, how the PDUs are used between MTAs on a multicast network. This example shall give only an impression, how the protocol works. Therefore the packets are described as a sort of function calls.

It is assumed that M0 is the sending MTA, which sends one message to the MTAs M1, M2, M3 and M4. Only M3 and M4 are assumed to be under EMCON. The message length is assumed to be a little larger than the maximum PDU size, which means that the message (DATA) is split into two fragments. Further we assume, that until now M0 had send 99 messages to M1, 77 messages to M2, 10 messages to M3 and 14 messages to M4.
M0 constructs an Address_PDU and the 2 Data_PDUs. It sends all 3 PDUs to the multicast network.

Address_PDU( Source_ID=M0, MSID=9876, TTL=1000, Total_PDUs=2, Dests=( Cnt=4, LDES=0, Dest_ID=((M1,100), (M2,78), (M3,11), (M4,15)))

Data_PDU( Source_ID=M0, MSID=9876, PDU_No=1, Data= First N octets of message)

Data_PDU( Source_ID=M0, MSID=9876, PDU_No=2, Data= Remaining octets of message)

Assuming M1, M2, M3 and M4 will receive the Address_PDU. M1, M3 and M4 shall receive the Data_PDUs without failures, while M2 shall receive the first Data_PDU incorrectly and the second one correctly. As M3 and M4 are in EMCON mode, they cannot send any ACK_PDU. M1 will send a completion ACK_PDU, while M2 will send an ACK_PDU indicating that the first Data_PDU is missing.

ACK_PDU( Source_ID=M1, Dest_ACK_Infos=(Cnt=1, ACK_Info=(M0, 9876, 0)))

ACK_PDU( Source_ID=M2, Dest_ACK_Infos=(Cnt=1, ACK_Info=(M0, 9876, 1,0)))

Now M0 has to re-transmit the first part of the message for M2, because M2 marked this PDU as missing. As M0 has received the completion ACK_PDU of M1, M1 is deleted from the Destination_List.

Address_PDU( Source_ID=M0, MSID=9876, TTL=1000, Total_PDUs=2, Dests=( Cnt=3, LDES=0, Dest_ID=((M2,78), (M3,11), (M4,15))))

Data_PDU( Source_ID=M0, MSID=9876, PDU_No=1, Data= First N octets of message)

Assuming that M2 will receive this PDU correctly, it will acknowledge it with the following ACK_PDU.

ACK_PDU( Source_ID=M2, Dest_ACK_Infos=(Cnt=1, ACK_Info=(M0, 9876, 0)))

As M0 has received the completion ACK_PDU of M2, M2 will be deleted from the Destination_List.

Supposing that M3 has left the EMCON situation, M3 will send its ACK to M0 as:
ACK_PDU( Source_ID=M3, Dest_ACKInfos=(Cnt=1, ACK_Info=(M0, 9876, 0)))

Assuming that an EMCON Re-transmission for M4 should take place, M0 will re-transmit the total message. As M1, M2 and M3 already have completely acknowledged the message, the Address_PDU will hold only one Destination_ID for M4.

Address_PDU( Source_ID=M0, MSID=9876, TTL=1000, Total_PDUs=2,
             Dests=( Cnt=1, LDES=0, Dest_ID=((M4,15)))

Data_PDU( Source_ID=M0, MSID=9876, PDU_No=1, Data= First N octets of message)

Data_PDU( Source_ID=M0, MSID=9876, PDU_No=2, Data= Remaining octets of message)

Supposing that M4 is still under EMCON and that now the Expiry_Time out of the according Address_PDU has been reached. M0 will send the following sequence of PDUs:

Discard_Message_PDU( Source_ID=M0, MSID=9876)

After some time M4 may leave EMCON and has to send its ACK_PDU. As assumed M4 has received the total message before the Expiry_Time has been reached, it will send

ACK_PDU( Source_ID=M4, Dest_ACKInfos=(Cnt=1, ACK_Info=(M0, 9876, 0)))

At this time the Message Transmitter part can hand over the message to Multicast_OUT to update the message queue entry. Now M0 will delete M4 from the Destination_List and will send out an Address_PDU holding an empty List_of_Destination_Entries:

Address_PDU( Source_ID=M0, MSID=9876, TTL=1000, Total_PDUs=2,
             Dests=( Cnt=0, LDES=0, Dest_ID=())

This last Address_PDU sent out by M0 will inform M1, M2, M3 and M4, that M0 received the completion ACK_PDUs of all receiving MTAs. This means, that all information about the message MSID=9876 of M0 can be deleted.
6. References


A. Appendix

A.1. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMCON</td>
<td>Emission Control</td>
</tr>
<tr>
<td>MAP</td>
<td>More Address_PDUs</td>
</tr>
<tr>
<td>MTA</td>
<td>Message Transfer Agent</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>RMP</td>
<td>Reliable Multicast Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>XTP</td>
<td>Xpress Transport Protocol</td>
</tr>
</tbody>
</table>

A.2. Predefined Protocol Parameters
ACK_TIME   Time between the output of two subsequent Data_PDUs.

MPDU_SIZE  This is the maximum PDU size used to transmit messages over the multicast network.

HOST_ID    This field holds the MTA’s own network address (Internet address).

EMCON_RTC  Maximum re-transmission counter controlling the number of message re-transmissions for EMCON MTAs.

EMCON_RTI  Re-transmission interval controlling the waiting time between successive re-transmissions for EMCON MTAs.

A.3. Proposed UDP Port Numbers

To allow multiplexed multicast communication between all MTAs of a network it is necessary to define 2 UDP port numbers:

Port 2753  is used for the data traffic from the Message Transmitter of Multicast_OUT to the Message Receiver of Multicast_IN.

Port 2754  is used for the traffic from the Message Receiver of Multicast_IN to the Message Transmitter of Multicast_OUT.

A.4. Proposed Algorithm for the Checksum

    #include "includes.h"
    /* function checksum
    *
    * This function computes and sets the checksum (FLETCHER algorithm).
    * If "offset" is NULL, then the checksum will be calculated but the
    * checksum bytes will not be set.
    * If "offset" is non-NULL, the checksum bytes at "offset" will be set
    * such that the checksum accumulations, when recalculated, will both
    * be zero.
    * return : checksum accumulations
    */
    int checksum(buffer, len, offset)
    octet *buffer;
    int len, offset;
    {
        unsigned int c0 c1;
        int cs;
        long ctmp, c10, c11;
        octet *hpp, *pls;

        if (offset) {

buffer[offset] = 0;
buffer[offset+1] = 0;
ctmp = len - offset - 1;
}

pls = buffer + len;
c0 = c1 = 0;
hpp = buffer;

while (hpp < pls) {
  if ((c0 += *hpp++) > 254) { c0 -= 255; }
  if ((c1 += c0) > 254) { c1 -= 255; }
} /* while */

if (offset) {
  c10 = c0;
  c11 = c1;
  if ((cs =((ctmp * c10) - c11) % 255L) < 0) { cs += 255; }
  buffer[offset] = cs;
  if ((cs =((c11 -((ctmp + 1L) * c10)) % 255L) < 0) { cs += 255; }
  buffer[offset+1] = cs;
} /* if (offset) */

return(c0 | c1);
}

Security Considerations

Security considerations are not discussed in this memo.

Author’s Addresses:

Phil Copeland                          Christian Riechmann
TNO/FEL                                FGAN/FFM
Oude Walsdorperweg 63                  Neuenahrer Strasse 20
PO Box 96864                           D-53343 WACHTBERG
2509 JG Den Haag                       Germany
Netherlands                            Phone: (+49) 228 9435 533
Phone: (+31) 70 3740305                 Fax:   (+49) 228 348 953
Fax:   (+31) 70 3280961                 Email: riechmann@fgan.de
Email: copeland@fel.tno.nl