extended Mail Transfer Protocol (XMTP)

J. Gordon Beattie, Jr., N2DSY
Andrew R. Funk, KB7UV
Frank Warren, KB4CYC

The Radio Amateur Telecommunications Society
206 North Vivyen Street
Bergenfield, New Jersey 07621
Telephone: +1.201.387.8896

ABSTRACT

The amount of store-and-forward traffic in the Amateur Radio Packet Network has increased to the point where significant optimization of the message forwarding scheme for packet bulletin board and other servers is required. The purpose of this paper is to present an enhanced message forwarding system which we call the "eXtended Mail Transfer Protocol" or XMTP which addresses this requirement. Further, an overall client/server model is included as a possible implementation enhancement to systems which plan to implement this protocol.

INTRODUCTION

In the current Amateur Packet Radio Network there are servers which are usually based on MS-DOS computers which use the ARRL AX.25 Link Layer Protocol to move store-and-forward messages across the network. The ROSE X.25 networking protocol is used in the authors' network and for the examples in this paper, but any of the other common networking protocols could be substituted.

The basic application environment architecture for the evolving ROSErver environment depends on the ROSE X.25 and the AX.25 protocol to convey bits and bytes. Over these connections (or "path" for all you connectionless folks), the ROSErver environment uses a client/server protocol called "Serialized Transaction Interface Protocol" or S-TIP to provide a remote operations invoker/responder facility. One of the "users" of S-TIP is XMTP. XMTP depends on the underlying services of S-TIP to provide the facilities of a transaction monitor. These functions include signaling and verification of transaction completeness, confirmation delivery if required by the S-TIP user, transfer syntax signaling including compression, application addressing, and application capability negotiation between systems.

S-TIP will be described in a future paper, but it is based on some of the concepts and capabilities of CCITT X.219/X.229 Remote Operations Services, Sun Microsystems' Remote Procedure Call (RPC) and the communications facilities of the "MINIX" operating system.

The role of XMTP is to provide a uniform and extensible platform for the movement of store-and-forward traffic. XMTP can be implemented using the current message header data, but the use of additional header elements as found in either the CCITT X.400 Message Handliny System and RFC-822 is strongly recommended. The convergence of these two protocols as outlined in RFC-987, RFC-1138 and RFC-1148 would facilitate the automatic interface of a great body of existing OSI and Internet software and systems. Gateways are only a part of the' benefit. The software available would improve the user interface, add multiple personal mail delivery and conference services, allow for mixed graphical, voice and text messages, as well as other interesting features.

XMTP FEATURES

XMTP provides for the rapid transmission of store-and-forward messages between systems. The data flow diagrammes inlcuded in this paper illustrate the connection-oriented process, but the same basic procedure could be used in connectionless or "multicast" environments.

The current dialogue between store-and-forward systems causes systems to send and wait for a response. In most networks, the network transit delay is high and this wastes time. In order to reduce this loss of time, XMTP optimizes the dialogue by reducing the number of times that a transaction "holds-up" the data flow during a forwarding cycle. The basic changes that XMTP brings to the store-and-forward message environment are:

1. The pre-registration of the capabilities of a system eliminates the need to exchange them before each data transfer. The current "SSID" or Smart System ID which is sent in brackets "[...]" at the start of every session is now eliminated except when such a message is received from the other system. This allows for backwards compatibility as well as for the recovery of peer system capability information after a crash or change.

2. The headers of all mail queued for forwarding are sent in a single exchange. Currently, systems send a short header including the Bulletin or Message ID and wait for a "go/no-go" response.- After receipt of a "go", the system sends the message and then waits for an acknowledgement before sending the next header.

3. XMTP supports simultaneous bidirectional forwarding. This reduces the store-and-forward transfer time significantly by filling both directions of the channel at the same time.

Some have suggested that all messages should be compressed into a single file and then transmitted. This could lead to very large transfer files and possibly cause the less than timely delivery of a particular message. XMTP submits each message to S-TIP as a separate transaction. Each transaction is compressed, transferred, and decompressed by the underlying service provided by S-TIP. S-TIP does not control this process, but simply acts as directed by XMTP. As such, if future changes to XMTP include the transfer of multiple messages in a single compressed package, then S-TIP will transparently handle the requirement.

XMTP FLOW DIAGRAMMES

XMTP supports three basic modes of operation: Simple Forwarding, Polled Forwarding and Duplex Forwarding. This section outlines the data flow process for each mode. The flow diagrammes along with a general architecture figure are at the end of this paper.

SIMPLE FORWARDING CASE

In the Simple forwarding case, the connection is established and then a Mail-Q-Event-Report is sent to the receiving system. This report summarizes all mail traffic for that system. It provides information which allows the receiving system to determine the priority which it will use to receive these messages, as well as if there is sufficient space and if the message is a duplicate. The message elements are outlined in the "XMTP DATA ELEMENTS" section of this paper. The receiving system then sets the appropriate status for each message on the sending system by sending a Mail-Q-Set-Status message. The sending system then starts to send each message as Mail-Create-Requests without waiting for individual acknowledgements. It should be noted at this point that what the sending system is doing is creating a message just like the one it has, on the receiving system. The receiving system can then send Mail-Log-Event-Report-Request messages reflecting the successful receipt of one or more messages. This may be repeated as additional messages are received.

SIMPLE POLLING CASE

In the Simple polling case, the connection is established and then a Mail-Q-Set-Request is sent to trigger the "sending" system into providing a list of queued messages. The sending system then sends a Mail-Q-Event-Report to the receiving system as was done in the Simple Forwarding case. The receiving system then sets the appropriate status for each message on the sending system by sending a Mail-Q-Set-Status message. The sending system then starts to send each message as Mail-Create-Requests without waiting for individual acknowledgements. It should be noted at this point that what the sending system is doing is creating a message just like the one it has, on the receiving system. The receiving system can then send Mail-Log-Event-Report-Request messages reflecting the successful receipt of one or more messages. This may be repeated as additional messages are received.

DUPLEX FORWARDING CASE

In the Duplex forwarding case, the flow is the same as in either or both of the other two cases, but data is allowed to flow in each direction at once.

ACKNOWLEDGEMENTS

The authors wish to thank all the users of the ROSErver/PRMBS software package as well as it's author, Brian Riley, KA2BQE, for their work and comments on the protocol and the research that went into this paper. The authors also would like to thank Nancy Beattie, N2FWI, Ted Beauchamp, KA2USU, David Elliot, KD6TH, Tom Moulton, W2VY, Bob Nelson, KB1BD, Buck Rogers, K4ABT, Don Rotolo, N2IRZ, and Bill Slack, NX2P for their research and observations.

XMTP DATA ELEMENTS

[S-TIP-.../XMTP-...]

This is the Smart System ID message element included in the "[]" message exchange performed after connection time. This signals each side that S-TIP and XMTP Services are available and that the Application Manager Managed Object is present. Specific selection of a suitable string is needed.

S-TIP Header

The S-TIP Header is based on the OSI model and uses the connectionless, Unit-Data Services to implement the protocol. Each Protocol Data Unit (PDU) is sent to the service interface by an application, and then delivered to a peer application entity. Some minimal state information about the PDU is maintained by the S-TIP provider. Some changes and enhancements are currently being implemented. Please contact the authors for the revised format.

```
S-TIP Header ::= SEQ OF {
        Network- Source Address, Dest Address, Header Chk,
                        Data Length
        Transport- Source Address, Dest Address, Checksum
        Session- Source Address, Dest Address
        Presentation- Source Address, Dest Address, PCI
        Application Context Name
        Operation Code  /* Such as mail message type */
        Mode /* Best Effort, Atomic, etc. */
        InvokeID  /* This is a transaction number */
        }
```

Mail-Q-Event-Report

This message is a notification sent by a system to signal the other system of the availability of mail. This message may be sent any time during a communication. The format is as follows:

```
Mail-Q-Event-Report ::= SEQ OF {
        S-TIP Header
        Object Class = Mail-Q-Summary-Record
        Object Instance = SystemName&RecordID
        Operation Time = UTC
        Operation Type = Mail-Q-Event-Report
        Operation Data = SET OF Mail-Q-Record
        }
```

```
Mail-Q-Record ::= SEQ OF {
        Object Class = Mail-Q-Record
        Object Instance = Mail-Q-RecordID
        Date/Time = UTC
        Hold Date/Time = UTC
        Mail Type ::= CHOICE { P, T, B, 0, . ..)
        Status ::= CHOICE { Y, N, D, K, . ..}
        Q-Status ::= CHOICE { G, R, H,  ...}
        Size = Uncompressed Byte Count
        To = FullyQualifiedAddressee
        From = FullyQualifiedAddressee
        Subject = OctetString
        Path = SEQ OF { Mail-Q-RecordID )
        }

Mail-Q-RecordID ::= SEQ OF {
        Message Number
        "@"
        SystemID
        }

FullyQualifiedAddressee ::= SEQ OF (
        CHOICE { Callsign | ApplicationName )
        Device ::= Callsign + Unique Identifier
        Organization ::= OctetString /* the # stuff */
        Locality ::= CHOICE { State | Province | etc. }
        Country ::= ISO3166-Alpha-2
        }

Subject = OctetString
```

The default values for Q-Status and Hold Date/Time are "H" and "0". Q-Status values are: G for Get, R for Remove and H for Hold.

Mail-Q-Set-Request

This message signals the other system to alter the status of Mail-Q-Record Q-Status attribute. This change can cause a mail message to be held until a later time (H), removed (R), or retrieved (G). The default values for Q-Status and Hold Date/Time are "H" and "0". This message may be sent anytime during a communication. The format is as follows:

```
Mail-Q-Set-Request ::= SEQ OF (
        S-TIP Header
        Object Class = Mail-Q-Record
        Object Instance = SystemName
        Operation Type = Mail-Q-Set-Request
        Operation Data = SET OF Mail-Q-Status
        }

Mail-Q-Set-Status ::= SEQ OF {
        Object Class = Mail-Q-Record
        Object Instance = Mail-Q-RecordID
        Hold Date/Time = UTC
        Q-Status ::= CHOICE { G, R, H,  ...}
        }
```

Mail-Create-Request

This message signals the other system to create a new mail object. The default values for Q-Status and Hold Date/Time are "H" and "0". This message may be sent anytime during a communication. The format is as follows:

```
Mail-Create-Request ::= SEQ OF {
        S-TIP Header
        Object Class = Mail-Record
        Object Instance = Mail-Q-RecordID
        Operation Time = UTC
        Operation Type = Mail-Create-Request
        Operation Data = SEQ OF {
                        Mail-Q-Record
                        Message-Body
                        }
        }
```

The default values for Q-Status and Hold Date/Time are "H" and "0". The Object Instance (Record ID) of the Mail-Create-Request is the message number used on the local system. The sender will always use its local message number. The receiver will replace the received message number with its own local value.

Mail-Q-Set-Request

This message signals the other system to send the list of Mail-Q-Records. This message may be sent anytime during a communication. The format is as follows:

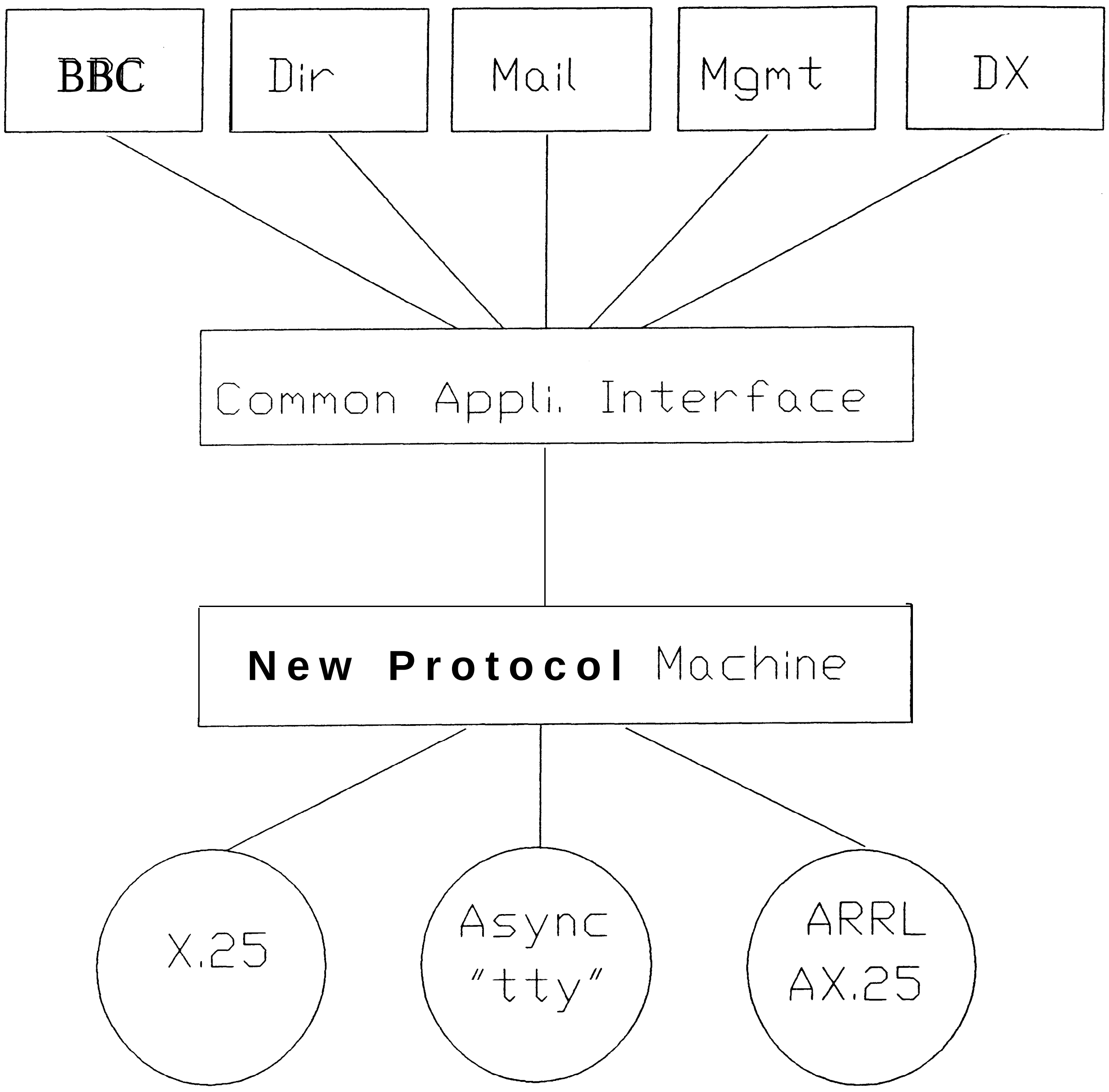
```
Mail-Q-Set-Request ::= SEQ OF {
        S-TIP Header
        Object Class = Mail-Q-Record
        Object Instance = SystemName
        Operation Type = Mail-Q-Set-Request
        Operation Data = SET OF { Mail-Q-RecordID }
        }
```

Mail-Q-Set-Response

This message is the response to the request for a list of Mail-Q-Records. The format is as follows:

```
Mail-Q-Set-Response ::= SEQ OF {
        S-TIP Header
        Object Class = Mail-Q-Summary-Record
        Object Instance = SystemName&RecordID
        Operation Time = UTC
        Operation Type = Mail-Q-Event-Report
        Operation Data = SET OF Mail-Q-Record
        }
```

New Protocol Implementation



file:stack

eXtended Mail Transfer Protocol (XMTP)

Simple Forwarding Case

System A		System B
>	Connection Request	>
<	Connection Accept	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Set-Request	<
>	Mail-Create-Request (Msg_1)	>
>	Mail-Create-Request (Msg_2)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report-Request (1-4)	<

extended Mail Transfer Protocol (XMTP)

Simple Polling Case

System A		System B
<	Connection Request	<
>	Connection Accept	>
c	Mail-Q-Set-Request	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Set-Request	<
>	Mail-Create-Request (Msg_1)	>
>	Mail-Create-Request (Msg_2)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report (l-4)	C

eXtended Mail Transfer Protocol (XMTP)

Duplex Forwarding Case

System A		System B
>	Connection Request	>
<	Connection Accept	<
>	Mail-Q-Event-Report	>
<	Mail-Q-Event-Report	<
<	Mail-Q-Set-Request	<
>	Mail-Q-Set-Request	>
>	Mail-Create-Request (Msg_1)	>
<	Mail-Create-Request (Msg_A)	<
>	Mail-Create-Request (Msg_2)	>
<	Mail-Create-Request (Msg_B)	<
>	Mail-Log-Event-Report (A-B)	>
>	Mail-Create-Request (Msg_3)	>
>	Mail-Create-Request (Msg_4)	>
<	Mail-Log-Event-Report (1-4)	<