INTERNET MESSAGE PROTOCOL

Jonathan B. Postel

August 1980

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

(213) 822-1511

TABLE OF CONTENTS

PREFACE

This is the second edition of this specification and should be treated
as a request for comments, advice, and suggestions. A great deal of
prior work has been done on computer aided message systems and some of
this is listed in the reference section. This specification was shaped
by many discussions with members of the ARPA research community, and
others interested in the development of computer aided message systems.
This document was prepared as part of the ARPA sponsored Internetwork
Concepts Research Project at ISI, with the assistance of Greg Finn,
Suzanne Sluizer, Alan Katz, Paul Mockapetris, and Linda Sato.

Jon Postel

INTERNET MESSAGE PROTOCOL

1. INTRODUCTION

This document describes an internetwork message system. The system is
designed to transmit messages between message processing modules
according to formats and procedures specified in this document. The
message processing modules are processes in host computers. Message
processing modules are located in different networks and together
constitute an internetwork message delivery system.

This document is intended to provide all the information necessary to
implement a compatible cooperating module of this internetwork message
delivery system.

1.1. Motivation

  As computer supported message processing activities grow on individual
  host computers and in networks of computers, there is a natural desire
  to provide for the interconnection and interworking of such systems.
  This specification describes the formats and procedures of a general
  purpose internetwork message system, which can be used as a standard
  for the interconnection of individual message systems, or as a message
  delivery system in its own right.

  This system also provides for the communication of data items beyond
  the scope of contemporary message systems. Messages can include data
  objects which could represent drawings, or facsimile images, or
  digitized speech. One can imagine message stations equipped with
  speakers and microphones (or telephone hand sets) where the body of a
  message or a portion of it is recorded digitized speech. The output
  terminal could include a graphics display, and the message might
  present a drawing on the display, and verbally (via the speaker)
  describe certain features of the drawing. This specification provides
  for the composition of complex data objects and their encoding in
  machine independent basic data elements.

1.2. Scope

  The Internet Message Protocol is intended to be used for the
  transmission of messages between networks. It may also be used for
  the local message system of a network or host. This specification was

developed in the context of the ARPA work on the interconnection of
networks, but it is thought that it has a more general scope.

The focus here is on the internal mechanisms to transmit messages,
rather than the external interface to users. It is assumed that a
number of user interface programs will exist. These will be both new
programs designed to work with this system and old programs designed
to work with earlier systems.

1.3. The Internetwork Environment

The internetwork message environment consists of processes which run
in hosts which are connected to networks which are interconnected by
gateways. Each network consists of many different hosts. The
networks are tied together through gateways. The gateways are
essentially hosts on two (or more) networks and are not assumed to
have much storage capacity or to "know" which hosts are on the
networks to which they are attached [1,2].

1.4. Model of Operation

This protocol is implemented in a process called a Message Processing
Module or MPM. The MPMs exchange messages by establishing full duplex
communication and sending the messages in a fixed format described in
this document. The MPM may also communicate other information by
means of commands described here.

A message is formed by a user interacting with a User Interface
Program or UIP. The user may utilize several commands to create
various fields of the message and may invoke an editor program to
correct or format some or all of the message. Once the user is
satisfied with the message it is submitted for transmission by placing
it in a data structure read by the MPM.

The MPM discovers the unprocessed input data (either by a specific
request or by a general background search), examines it, and, using
routing tables (or some other method), determines which outgoing link
to use. The destination may be another user on the same host, one on
another host on a network in common with the same host, or a user in
another network.

In the first case, another user on this host, the MPM places the
message in a data structure read by the destination user, where that
user's UIP will look for incoming messages.

In the second case, the user on another host in this network, the MPM
transmits the message to the MPM on that host. That MPM then repeats

the routing decision, and discovering the destination is local to it, places the message in the data structure shared with the destination user.

In the third case, the user on a host in another network, the MPM transmits the messages to an MPM in that network if it knows how to establish a connection directly to it; otherwise, the MPM transmits the message to an MPM that is "closer" to the destination. An MPM might not know of direct connections to MPMs in all other networks, but it must be able to select a next MPM to handle the message for each possible destination network.

An MPM might know a way to establish direct connections to each of a few MPMs in other nearby networks, and send all other messages to a particular big brother MPM that has a wider knowledge of the internet environment.

An individual network's message system may be quite different from the internet message system. In this case, intranet messages will be delivered using the network's own message system. If a message is addressed outside the network, it is given to an MPM which then sends it through the appropriate gateways to (or towards) the MPM in the destination network. Eventually, the message gets to an MPM on the network of the recipient of the message. The message is then sent via the local message system to that host.

When local message protocols are used, special conversion programs are required to transform local messages to internet format when they are going out, and to transform internet messages to local format when they come into the local environment. Such transformations potentially lead to information loss. The internet message format attempts to provide features to capture all the information any local message system might use. However, a particular local message system is unlikely to have features equivalent to all the possible features of the internet message system. Thus, in some cases the transformation of an internet message to a local message discards some of the information. For example, if an internet message carrying mixed text and speech data in the body is to be delivered in a local system which only carries text, the speech data may be replaced by the text string "There was some speech here". Such discarding of information is to be avoided when at all possible, and to be deferred as long as possible; still, the possibility remains that in some cases it is the only reasonable thing to do.

1.5. Interfaces

   The MPM calls on a reliable communication procedure to communicate
   with other MPMs. This is a Transport Level protocol such as the
   Transmission Control Protocol (TCP) [3]. The interface to such a
   procedure conventionally provides calls to open and close connections,
   send and receive data on a connection, and some means to signal and be
   notified of special conditions (i.e., interrupts).

   The MPM receives input and produces output through data structures
   that are produced and consumed respectively by user interface (or
   other) programs.

2. FUNCTIONAL DESCRIPTION

This section gives an overview of the Internet Message System and its
environment.

2.1. Terminology

   The messages are routed by a process called the Message Processing
   Module or MPM. Messages are created and consumed by User Interface
   Programs (UIPs) in conjunction with users.

   The basic unit transferred between MPMs is called a message. A
   message is made up of a transaction identifier (which uniquely
   identifies the message), a command (which contains the necessary
   information for delivery), and document. The document may have a
   header and a body.

   For a personal letter the document body corresponds to the contents of
   the letter; the document header corresponds to the date line,
   greeting, and signature.

   For an inter-office memo the document body corresponds to the text;
   the document header corresponds to the header of the memo.

   The commands correspond to the information used by the Post Office or
   the mail room to route the letter or memo. Some of the information in
   the command is supplied by the UIP.

2.2. Assumptions

   The following assumptions are made about the internetwork environment:

   In general, it is not known what format intranet addresses will
   assume. Since no standard addressing scheme would suit all networks,
   it is safe to assume there will be several and that they will change
   with time. Thus, frequent software modification throughout all
   internet MPMs would be required if such MPMs were to know about the
   formats on many networks. Therefore, each MPM which handles internet
   messages is required to know only the minimum necessary to deliver
   them.

   Each MPM is required to know completely only the addressing format of
   its own network(s). In addition, the MPM must be able to select an
   output link for each message addressed to another network or host.
   This does not preclude more intelligent behavior on the part of a
   given MPM, but at least this minimum is necessary. Each network has a
   unique name and numeric address. Such names and addresses are

registered with a naming authority and may be listed in documents such
as Assigned Numbers [4].

Each MPM will have a unique internet address. This feature will
enable every MPM to place a unique "handling-stamp" on a message which
passes through the MPM enroute to delivery.

2.3. General Specification

There are several aspects to a distributed service to be specified.
First, there is the service to be provided; that is, the
characteristics of the service as seen by its users. Second, there is
the service it uses; that is, the characteristics it assumes to be
provided by some lower level service. And third, there is the
protocol used between the modules of the distributed service.

```
       User                                        User
        \                                           /
         UIP                                    UIP
          \                                       /
       --+-------------------------------------+-- Service
         |   \                          /    | Interface
         |  +--------+           +--------+   |
         |  | Module | <--Protocol--> | Module |   |
         |  +--------+           +--------+   |
         |         \                  /         |
         |       +---------------------+       |
         |       | Communication Service |       |
         |       +---------------------+       |
         |                                     |
         +-------------------------------------+
```
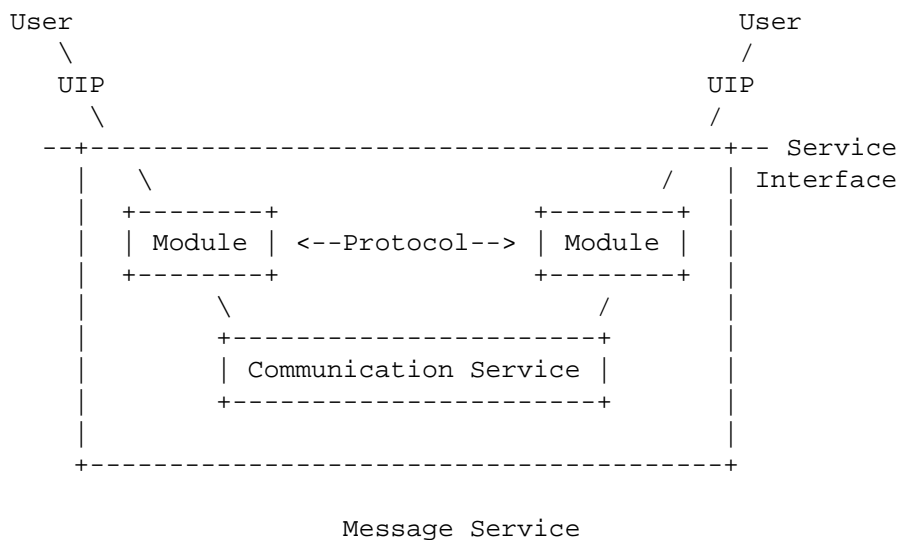
                      Message Service

                       Figure 1.

  The User/Message Service Interface

     The service the message delivery system provides is to accept
     messages conforming to a specified format, to attempt to deliver
     those messages, and to report on the success or failure of the
     delivery attempt. This service is provided in the context of an
     interconnected system of networks and may involve relaying a message
     through several intermediate MPMs via different communication
     services.

   The Message/Communication Service Interface

      The message delivery system calls on a communication service to
      transfer information from one MPM to another. There may be
      different communication services used between different pairs of
      MPMs, though all communication services must meet the service
      characteristics described below.

      It is assumed that the communication service provides a reliable
      two-way data stream. Such a data stream can usually be obtained in
      computer networks from the transport level protocol, for example,
      the Transmission Control Protocol (TCP) [3]. In any case, the
      properties the communication service must provide are:

         o Logical connections for two way simultaneous data flow of
            arbitrary data (i.e., no forbidden codes). All data sent is
            delivered in order.

         o Simple commands to open and close the connections, and to send
            and receive data on the connections.

         o Controlled flow of data so that data is not transmitted faster
            that the receiver chooses to consume it (on the average).

         o Transmission errors are corrected without user notification or
            involvement of the sender or receiver. Complete breakdown on
            communication is reported to the sender or receiver.

   The Message-Message Protocol

      The protocol used between the distributed modules of the message
      delivery system, that is, the MPMs, is a small set of commands which
      convey requests and replies. These commands are encoded in a highly
      structured and rigidly specified format.

2.4. Mechanisms

   MPMs are processes which use some communication service. A pair of
   MPMs which can communicate reside in a common interprocess
   communication environment. An MPM might exist in two (or more)
   interprocess communication environments, and such an MPM might act to
   relay messages between MPMs. Messages may be held for a time in an
   MPM; the total path required for delivery need not be available
   simultaneously.

   From the time a message is accepted from a UIP by an MPM until it is
   delivered to a UIP by an MPM and an acknowledgment is returned to the

   originating UIP, the message is considered to be active in the message
   system.

```
        User                                              User
         \                                                 /
          UIP                                            UIP
           \                                             /
        +------------------------------------------------------------+
        |    \                                         /     |
        |  +-----+              +-----+              +-----+  |
        |  | MPM | <--Protocol--> | MPM | <--Protocol--> | MPM |  |
        |  +-----+              +-----+              +-----+  |
        |    |                 /   \                   |     |
        |  +--------------------+  +----------------------+  |
        |  |Communication Service A|  |Communication Service B|  |
        |  +--------------------+  +----------------------+  |
        |                                                    |
        +------------------------------------------------------------+
```
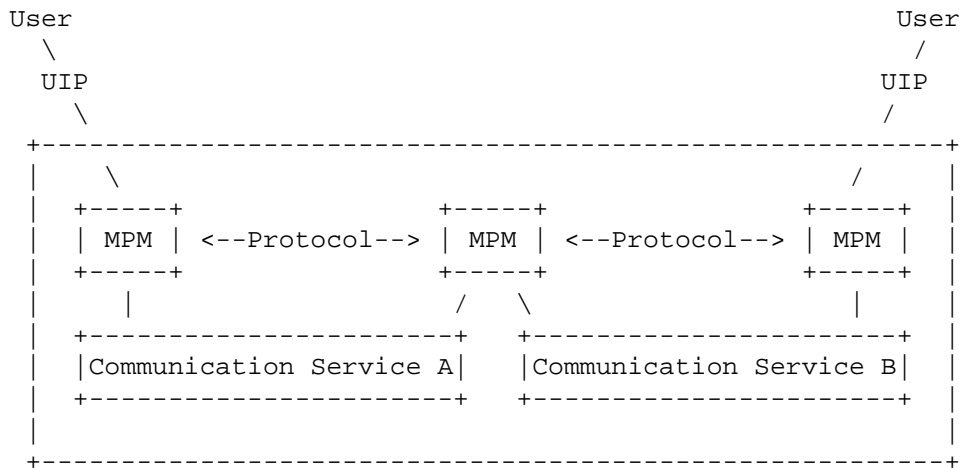
                 Message Service with Internal Relaying

                             Figure 2.

   It should be clear that there are two roles an MPM can play, an
   end-point MPM or a relay MPM. Most MPMs will play both roles. A
   relay MPM acts to relay messages from one communication environment to
   another. An end-point MPM acts as a source or destination of
   messages.

   The transfer of data between UIPs and MPMs is viewed as the exchange
   of data structures which encode messages. The transfer of data
   between MPMs is also in terms of the transmission of structured data.

```
              +-----+      DATA        +-----+
     USER-->| UIP |-->STRUCTURES-->| MPM |-->other
              +-----+    +-----+      +-----+    MPMs
                            |     |
                            |  +-----+
                          +--|     |
                            |  +-----+
                          +--|        |
                            |        |
                            +-----+
```

```
              +-----+      DATA        +-----+
  other-->| MPM |-->STRUCTURES-->| UIP |-->USER
  MPMs    +-----+    +-----+      +-----+
                            |     |
                            |  +-----+
                          +--|        |
                            |  +-----+
                          +--|        |
                            |        |
                            +-----+
```
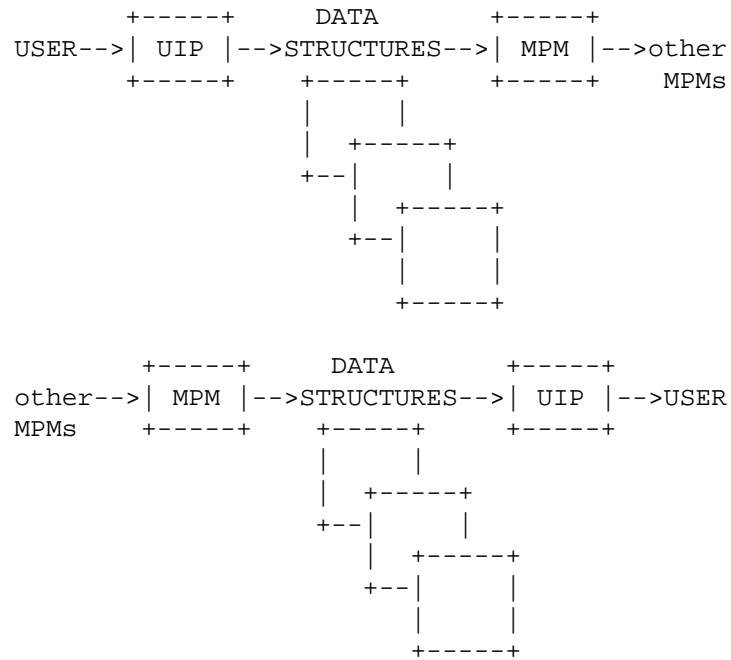
                        Message Flow

                        Figure 3.

   In the following, a message will be described as a structured data
   object represented in a particular kind of typed data elements. This
   is how a message is presented when transmitted between MPMs or
   exchanged between an MPM and a UIP. Internal to an MPM (or a UIP), a
   message may be represented in any convenient form.

2.5. Relation to Other Protocols

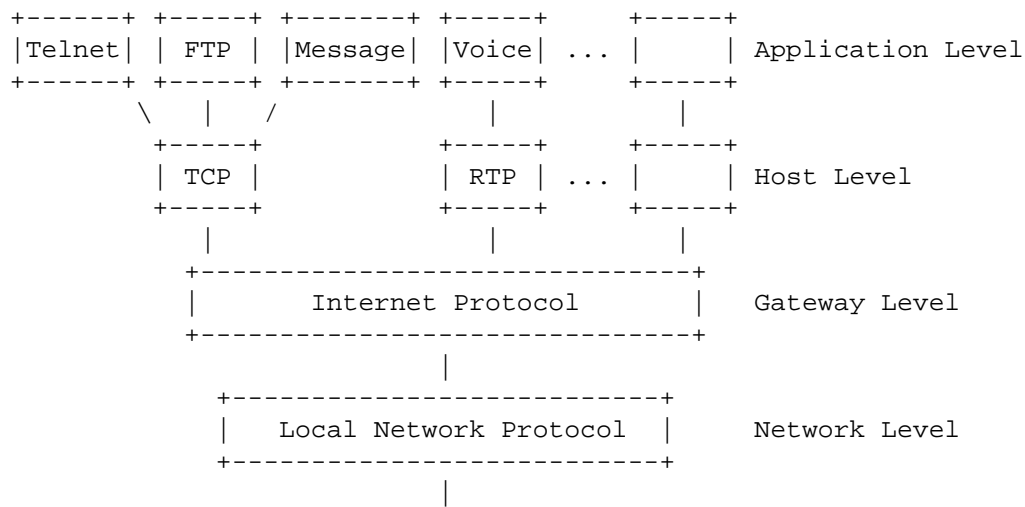   This protocol the benefited from the earlier work on message protocols
   in the ARPA Network [5,6,7,8,9], and the ideas of others about the
   design of computer message systems
   [10,11,12,13,14,15,16,17,18,19,20,21].

   Figure 4 illustrates the place of the message protocol in the ARPA
   internet protocol hierarchy:

```
  +------+ +-----+ +-------+ +-----+      +-----+
  |Telnet| | FTP | |Message| |Voice| ... |     | Application Level
  +------+ +-----+ +-------+ +-----+      +-----+
        \    |   /            |         |
         +-----+           +-----+    +-----+
         | TCP |           | RTP | ... |     | Host Level
         +-----+           +-----+    +-----+
            |                 |          |
         +------------------------------+
         |      Internet Protocol       | Gateway Level
         +------------------------------+
                        |
           +--------------------------+
           |   Local Network Protocol  |   Network Level
           +--------------------------+
                        |
```


                   Protocol Relationships

                        Figure 4.

  Note that "local network" means an individual or specific network.
  For example, the ARPANET is a local network.

  The message protocol interfaces on one side to user interface programs
  and on the other side to a reliable transport protocol such as TCP.

  In this internet message system the MPMs communicate directly using
  the lower level transport protocol. In the old ARPANET system,
  message transmission was part of the file transfer protocol.

```
      +------+   +-----+   +-------+
      |Telnet|   | FTP |---|Message|           Application Level
      +------+   +-----+   +-------+
           \      /
  +-----+   +-----+
  |Voice|---| NCP |                            Host Level
  +-----+   +-----+
              |
              |
              |                                Gateway Level
              |
              |
      +----------------+
      |    ARPA NET    |                       Network Level
      +----------------+
```
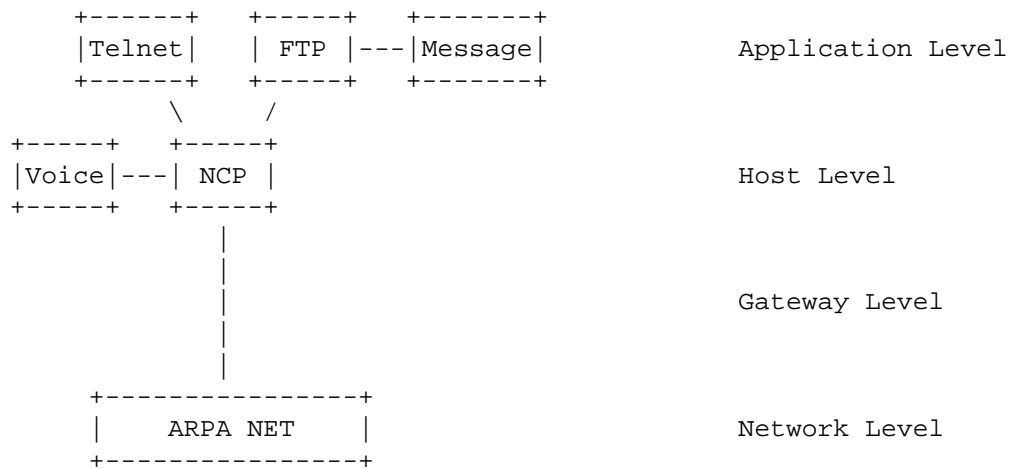

                        Old ARPANET Protocols

                           Figure 5.

 Note that in the old ARPANET protocols one can't send messages (or
 communicate in any way) to other networks since it has no gateway
 level or internet protocol [5].

3. DETAILED SPECIFICATION

The presentation of the information in this section is difficult since
everything depends on everything, and since this is a linear medium it
has to come in some order. In this attempt, a brief overview of the
message structure is given, the detail of the message is presented in
terms of data objects, the various data objects are defined, and finally
the representation of the data elements is specified. Several aspects
of the message structure are based on the NSW Transaction Protocol [22],
and similar (but more general) proposals [23,24].

3.1. Overview of Message Structure

   A message is normally composed of three parts: the identification,
   the command, and the document. Each part is in turn composed of data
   objects.

   The identification part is composed of a transaction number assigned
   by the originating MPM and the MPM identifier.

   The command part is composed of an operation type, an operation code,
   the arguments to the operation, error information, the destination
   mailbox, and a trace. The trace is a list of the MPMs that have
   handled this message.

   The document part is a data structure. The message delivery system
   does not depend on the contents of the document part. A standard for
   the document part is defined in reference [25].

   The following sections define the representation of a message as a
   structured object composed of other objects. Objects in turn are
   represented using a set of basic data elements.

   The basic data elements are defined in section 3.7. In summary, these
   are exact forms for representing integers, strings, booleans, et
   cetera. There are also two elements for building data structures:
   list and property list. Lists are simple lists of elements, including
   lists. Property lists are lists of pairs of elements, where the first
   element of each pair names the pair. That is, a property list is a
   list of <name,value> pairs. In general, when an object is composed of
   multiple instances of a simpler object it is represented as a list of
   the simpler objects. When an object is composed of a variety of
   simpler objects it is represented as a property list of the simpler
   objects. In most uses of the property list representation, the
   presence of <name,value> pairs in addition to those specifically
   required is permitted.

3.2. Message Structure

   An internet message is composed of two or three parts. The first is
   the Identification which identifies the transaction; the second is the
   Command; and the optional third part is the Document.

   When shipped between two MPMs, a message will take the form of a
   property list, with the <name,value> pairs in this order.

      MESSAGE is:

        (Identification, Command [, Document])

   It is convenient to batch several messages together, shipping them
   as a unit from one MPM to another. Such a group of messages is
   called a message-bag.

   A message-bag will be a list of Messages; each Message is of the
   form described above.

      MESSAGE-BAG is:

        (Message, Message, ...)

   The Identification

      This is the transaction identifier. It is assigned by the
      originating MPM. The identification is composed of the MPM
      identifier, and a transaction number unique in that context for this
      message.

   The Command

      The command is composed of a mailbox, an operation code, the
      arguments to that operation, some error information, and a trace of
      the route of this message. The command is implemented by a property
      list which contains <name,value> pairs, where the names are used to
      identify the associated argument values.

   The Document

      The document portion of an internet message is optional and when
      present is a data structure as defined in [25].

3.3. Identification

   Each message must have a unique identifier while it exists in the
   message delivery system. This is provided by the combination of the
   unique identifier of the MPM and a unique transaction number chosen
   for the message by this MPM.

      IDENTIFICATION is:

         (mpm-identifier, transaction-number)

   The mpm-identifier is based on the host address of the computer in
   which the MPM resides. If there is more than one MPM in a host the
   mpm-identifier must be extended to distinguish between the co-resident
   MPMs.

3.4. Command

   This section describes the commands MPMs use to communicate between
   themselves. The commands come in pairs, with each request having a
   corresponding reply.

      COMMAND is:

         (mailbox, operation, [arguments,]
                                     [error-class, error-string,] trace)

   The mailbox is the "To" specification of the message. Mailbox is a
   property list of general information, some of which is the essential
   information for delivery, and some of which could be extra information
   which may be helpful for delivery. Mailbox is different from address
   in that address is a very specific property list without extra
   information. The mailbox includes a specification of the user, when
   a command is addressed to the MPM itself (rather than a user it
   serves) the special user name "*MPM*" is specified.

   The operation is the name of the operation or procedure to be
   performed.

   The arguments to the operation vary from operation to operation.

   The error information is composed of a error class code and a
   character string, and indicates what, if any, error occurred. The
   error information is normally present only in replies, and not present
   in requests.

The trace is a list of the MPMs that have handled the message. Each
MPM must add its handling-stamp to the list.

   3.4.1. Command: DELIVER

      function: Sends a document to a mailbox.

      reply: The reply is ACKNOWLEDGE.

      arguments:

        type-of-service: one or more of the following:

           "REGULAR" regular delivery
           "FORWARD" message forwarding
           "GENDEL"   general delivery
           "PRIORITY" priority delivery

   3.4.2. Command: ACKNOWLEDGE

      function: Reply to DELIVER.

      arguments:

        reference: the identifier of the originating message.

        address:

           The address is the final mailbox the message was delivered to.
           This would be different from the original mailbox if the message
           was forwarded, and is limited to the essential information
           needed for delivery.

        type-of-service: one of the following:

           "GENDEL"   message was accepted for general delivery
           "REGULAR"   message was accepted for normal delivery
           "PRIORITY" message was accepted for priority delivery

        error-class:
        error-string:

           If the document was delivered successfully, the error
           information has class 0 and string "ok". Otherwise, the error
           information has a non-zero class and the string would be one of
           "no such user", "no such host", "no such network", "address
           ambiguous", or a similar response.

        trail: the trace from the DELIVER command.

   3.4.3. Command: PROBE

      function: Finds out if specified mailbox (specified in mailbox of
      the command) exists at a host.

      reply: The reply is RESPONSE.

      arguments: none.

   3.4.4. Command: RESPONSE

      function: Reply to PROBE.

      arguments:

        reference: the identification of the originating PROBE.

        address: a specific address.

        error-class:
        error-string:

          If the mailbox was found the error class is 0 and the error
          string is "OK". If the mailbox has moved and a forwarding
          address in known the error class is 1 and the error string is
          "Mailbox moved, see address". Otherwise the error class is
          greater than 1 and the error string may be one of the following:
          "Mailbox doesn't exist", "Mailbox full", "Mailbox has moved, try
          the new location indicated in the address".

        trail: the trace which came from the originating PROBE.

   3.4.5. Command: CANCEL

      function: Abort request for specified transaction.

      reply: The reply is CANCELED.

      arguments:

        reference: identification of transaction to be canceled.

   3.4.6. Command: CANCELED

      function: Reply to CANCEL.

      arguments:

        reference: identification of canceled transaction.

        error-class:
        error-string:

           If the command was canceled the error class is 0 and the error
           string is "OK". Otherwise the error class is positive and the
           error string may be one of the following: "No such transaction",
           or any error for an unreachable mailbox.

        trail: the trace of the CANCEL command.

   To summarize again, a command generally consists of a property list of
   the following objects:

      name   value
      ----   -----
      mailbox   property list of address information
      operation   name of operation
      arguments   ---
      error-class   numeric class of the error
      error-string   text description of the error
      trace   list (handling-stamp, ...)

3.5. Document

   The actual document follows the command. The message delivery system
   does not depend on the document, examine it, or use it in any way.
   The standard for the contents of the document is reference [25]. The
   document must be the last <name,value> pair in the message property
   list.

3.6. Message Objects

   In the composition of messages, we use a set of objects such as
   mailbox or date. These objects are encoded in basic data elements.
   Some objects are simple things like integers or character strings,
   other objects are more complex things built up of lists or property
   lists.

   The following is a list of the objects used in messages. The object
   descriptions are in alphabetical order.

   Action

      The type of handling action taken by the MPM when processing a
      message. One of ORIGIN, RELAY, FORWARD, or DESTINATION.

   Address

      Address is intended to contain the minimum information necessary to
      deliver a message, and no more (compare with mailbox).

         An address is a property list. An address contains the following
         <name,value> pairs:

            name   description
            ----   -----------
            NET   network name
            HOST   host name
            USER   user name

         or:

            name   description
            ----   -----------
            MPM   mpm-identifier
            USER   user name

   Answer

      A yes (true) or no (false) answer to a question.

   Arguments

      Many operations require arguments, which differ from command to
      command. This "object" is a place holder for the actual arguments
      when commands are described in a general way.

City

   The character string name of a city.

Command

   (mailbox, operation [,arguments]
                                 [,error-class, error-string], trace)

Country

   The character string name of a country.

Date

   The date and time are represented according to the International
   Standards Organization (ISO) recommendations [26,27,28]. Taken
   together the ISO recommendations 2014, 3307, and 4031 result in the
   following representation of the date and time:

     yyyy-mm-dd-hh:mm:ss,fff+hh:mm

   Where yyyy is the four-digit year, mm is the two-digit month, dd is
   the two-digit day, hh is the two-digit hour in 24 hour time, mm is
   the two-digit minute, ss is the two-digit second, and fff is the
   decimal fraction of the second. To this basic date and time is
   appended the offset from Greenwich as plus or minus hh hours and mm
   minutes.

   The time is local time and the offset is the difference between
   local time and Coordinated Universal Time (UTC). To convert from
   local time to UTC algebraically subtract the offset from the local
   time.

   For example, when the time in
           Los Angeles is 14:25:00-08:00
           the UTC is   22:25:00

   or when the time in
           Paris is   11:43:00+01:00
           the UTC is   10:43:00

Document

   The document is the user's composition and is not used by the
   message delivery system in any way.

Error-Class

A numeric code for the class of the error. The error classes are
coded as follows:

= 0: indicates success, no error.
  This is the normal case.
= 1: failure, address changed.
  This error is used when forwarding is possible, but not allowed
  by the type of service specified.
= 2: failure, resources unavailable.
  These errors are temporary and the command they respond to may
  work if attempted at a later time.
= 3: failure, user error.
  For example, unknown operation, or bad arguments.
= 4: failure, MPM error. Recoverable.
  These errors are temporary and the command they respond to may
  work if attempted at a later time.
= 5: failure, MPM error. Permanent.
  These errors are permanent, there is no point in trying the same
  command again.
= 6: Aborted as requested by user.
  The response to a successfully canceled command.

   Error-String

     This is a character string describing the error. Possible errors:

            error-string   error-class

       No errors   0
       Ok   0
       Mailbox Moved, see address   1
       Mailbox Full, try again later   2
       Syntax error, operation unrecognized   3
       Syntax error, in arguments   3
       No Such User   3
       No Such Host   3
       No Such Network   3
       No Such Transaction   3
       Mailbox Does Not Exist   3
       Ambiguous Address   3
       Server error, try again later   4
       No service available   5
       Command not implemented   5
       Aborted as requested by user   6

   Handling-Stamp

     The handling-stamp indicates the MPM, the date (including the time)
     that a message was processed by an MPM, and the type of handling
     action taken.

     (mpm-identifier, date, action)

   Host

     The character string name of a host.

   Identification

     This is the transaction identifier associated with a particular
     message. It is the transaction number, and the MPM identifier of
     the originating MPM.

     (mpm-identifier, transaction-number)

   Internet Address

      This identifies a host in the ARPA internetwork environment. When
      used as a part of identification, it identifies the originating host
      of a message. The internet address is a 32 bit number, the higher
      order 8 bits identify the network, and the lower order 24 bits
      identify the host on that network [2]. For use in the MPMs the
      internet address is divided into eight bit fields and the value of
      each field is represented in decimal digits. For example, the
      ARPANET address of ISIE is 167837748 and is represented as
      10,1,0,52. Further, this representation may be extended to include
      an address within a host, such as the TCP port of the MPM, for
      example, 10,1,0,52,0,45.

   Mailbox

      This is the destination address of a user of the internetwork mail
      system. Mailbox contains information such as network, host,
      location, and local user indentifier of the recipient of the
      message. Some information contained in mailbox may not be necessary
      for delivery.

      As an example, when one sends a message to someone for the first
      time, he may include many items which are not necessary simply to
      insure delivery. However, once he gets a reply to this message, the
      reply will contain an Address (as opposed to Mailbox) which may be
      used from then on.

        A mailbox is a property list. A mailbox might contain the
        following <name,value> pairs:

           name   description
           ----   -----------
           MPM   mpm-identifier
           NET   network name
           HOST   host name
           PORT   address of MPM within the host
           USER   user name
           ORG   organization name
           CITY   city
           STATE   state
           COUNTRY   country
           ZIP   zip code
           PHONE   phone number

      The minimum mail box is an Address.

    MPM-Identifier

      The internetwork address of the MPM. This may be the ARPA Internet
      Address or an X.121 Public Data Network Address [29]. The
      mpm-identifier is a property list which has one <name,value> pair.
      This unusual structure is used so that it will be easy to determine
      the type of address used.

   Network

      This character string name of a network.

   Operation

      This names the operation or procedure to be performed. It is a
      character string name.

   Organization

      This character string name of a organization.

   Phone

      This character string name representation of a phone number. For
      example the phone number of ISI is 1 (international region) + 213
      (area code) + 822 (central office) + 1511 (station number) =
      12138221511.

   Port

      This names the port or subaddress within a host of the MPM. The
      default port for the MPM is 45 (55 octal) [4].

   Reference

      The reference is an identification from an earlier message.

   State

      The character string name of a state.

Trace

   Each MPM that handles the message must add its handling-stamp to
   this list. This will allow detection of messages being sent in a
   loop within the internet mail system, and aid in fault isolation.

Trail

   When a message is sent through the internetwork environment, it
   acquires this trace, a list of MPMs that have handled the message.
   This list is then carried as the trail in a reply or acknowledgment
   of that message. Requests and replies always have a trace and each
   MPM adds its handling-stamp to this trace. Replies, in addition,
   have a trail which is the complete trace of the original message.

Transaction Number

   This is a number which is uniquely associated with this transaction
   by the originating MPM. It identifies the transaction. (A
   transaction is a message and acknowledgment.) A transaction number
   must be unique during the time which the message (a request or
   reply) containing it could be active in the network.

Type-of-Service

   A service parameter for the delivery of a message, for instance a
   message could be delivered (REGULAR), forwarded (FORWARD), turned
   over to general delivery (GENDEL) (i.e., allow a person to decide
   how to further attempt to deliver the message), or require priority
   handling (PRIORITY).

User

   The character string name of a user.

X121 Address

   This identifies a host in the Public Data Network environment. When
   used as a part of identifier, it identifies the originating host of
   a message. The X121 address is a sequence of up to 14 digits [29].
   For use in the MPMs the X121 address is represented in decimal
   digits.

Zip Code

  The character string representation of a postal zip code. The zip
  code of ISI is 90291.

3.7. Data Elements

  The data elements defined here are similar to the data structure and
  encoding used in NSW [30].

  Each of the diagrams which follow represent a sequence of octets.
  Field boundaries are denoted by the "|" character, octet boundaries by
  the "+" character. Each element begins with a one-octet code. The
  order of the information in each element is left-to-right. In fields
  with numeric values the high-order (or most significant) bit is the
  left-most bit. For transmission purposes, the leftmost octet is
  transmitted first. Cohen has described some of the difficulties in
  mapping memory order to transmission order [31].


```
Code  Type           Representation
----  ----           --------------


                     +------+
  0   No Operation   |  0   |
                     +------+


                     +------+------+------+------+------
  1   Padding        |  1   |    octet count   | Data ...
                     +------+------+------+------+------


                     +------+------+
  2   Boolean        |  2   | 1/0  |
                     +------+------+


                     +------+------+------+
  3   Index          |  3   |    Data     |
                     +------+------+------+


                     +------+------+------+------+------+
  4   Integer        |  4   |          Data           |
                     +------+------+------+------+------+
```

```
        Extended        +------+------+------+------+------
     5  Precision       |  5   |    octet count    | Data ...
        Integer         +------+------+------+------+------


                        +------+------+------+------+------
     6  Bit String      |  6   |      bit count    | Data ...
                        +------+------+------+------+------


                        +------+------+------
     7  Name String     |  7   | count|  Data ...
                        +------+------+------


                        +------+------+------+------+------
     8  Text String     |  8   |    octet count    | Data ...
                        +------+------+------+------+------


                        +------+------+------+------+-----
     9  List            |  9   |    octet count    | Data ...
                        +------+------+------+------+-----


                        +------+------+------+------+------
    10  Proplist        |  10  |    octet count    | Data ...
                        +------+------+------+------+------


                        +------+
    11  End of List     |  11  |
                        +------+
```

Element code 0 (NOP) is an empty data element used for padding when it
is necessary. It is ignored.

Element code 1 (PAD) is used to transmit large amounts of data with a
message for test or padding purposes. The type-octet is followed by a
three-octet count of the number of octets to follow. No action is
taken with this data but the count of dummy octets must be correct to
indicate the next element code.

Element code 2 (BOOLEAN) is a boolean data element. The octet
following the type-octet has the value 1 for True and 0 for False.

Element code 3 (INDEX) is a 16-bit unsigned integer datum. Element
code 3 occupies only 3 octets.

Element code 4 (INTEGER) is a signed 32-bit integer datum. This will
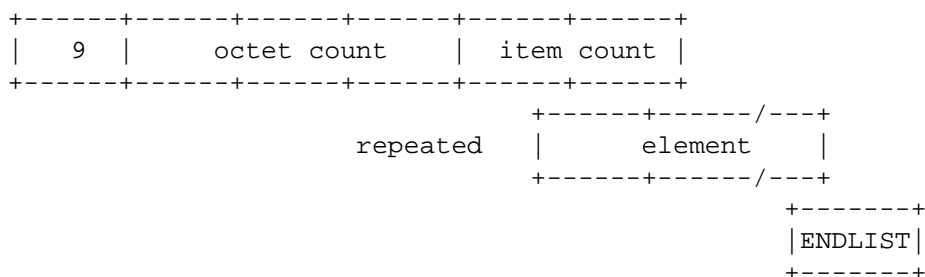always occupy five octets. Representation is two's complement.

Element code 5 (EPI) is an extended precision integer. The type octet
is followed by a three-octet count of the number of data octets to
follow. Representation is two's complement.

Element code 6 (BITSTR) is a bit string element for binary data. The
bit string is padded on the right with zeros to fill out the last
octet if the bit string does not end on an octet boundary. This data
type must have the bit-count in the three-octet count field instead of
the number of octets.

Element code 7 (NAME) is used for the representation of character
string names (or other short strings). The type octet is followed by
a one-octet count of the number of characters (one per octet) to
follow. Seven bit ASCII characters are used, right justified in the
octet. The high order bit in the octet is zero.

Element code 8 (TEXT) is used for the representation of text. The
type octet is followed by a three-octet count of the number of
characters (one per octet) to follow. Seven bit ASCII characters are
used, right justified in the octet. The high order bit in the octet
is zero.

Element code 9 (LIST) can be used to create structures composed of
other elements. The three-octet octet count specifies the number of
octets in the whole list (i.e., the number of octets following this
count field to the end of the list, not including the ENDLIST octet).
The two-octet item count contains the number of elements which follow.
Any element may be used including list itself.

```
    +------+------+------+------+------+------+
    |  9   |    octet count    |  item count |
    +------+------+------+------+------+------+
                               +------+------/---+
                   repeated    |     element     |
                               +------+------/---+
                                         +-------+
                                         |ENDLIST|
                                         +-------+
```

In some situations it may not be possible to know the length of a list

until the head of it has been transmitted. To allow for this a
special ENDLIST element is defined. A list of undetermined length is
transmitted with the octet count cleared to zero, and the item count
cleared to zero. A null or empty List, one with no elements, has an
octet count of two (2) and an item count of zero (0). The ENDLIST
element always follows a LIST, even when the length is determined.

Element code 10 (PROPLIST) is the Property List element. It is a
special case of the list element, in which the elements are in pairs
and the first element of each pair is a name. It has the following
form:

```
   +------+------+------+------+------+
   | 10   |     octet count    | pair |
   +------+------+------+------+------+
                     +------+------/---+------+------/---+
         repeated    | name element   | value element   |
                     +------+------/---+------+------/---+
                                                     +-------+
                                                     |ENDLIST|
                                                     +-------+
```

The Property List structure consists of a set of unordered
<name,value> pairs. The pairs are composed of a name which must be a
NAME element and a value which may be any kind of element. Following
the type code is a three-octet octet count of the following octets.
Following the octet count is a one-octet pair count of the number of
<name,value> pairs in the property list.

The name of a <name,value> pair is to be unique within the property
list, that is, there shall be at most one occurrence of any particular
name in one property list.

In some situations it may not be possible to know the length of a
property list until the head of it has been transmitted. To allow for
this the special ENDLIST element is defined. A property list of
undetermined length is transmitted with the octet count cleared to
zero, and the pair count cleared to zero. A null or empty property
list, one with no elements, has an octet count of one (1) and an pair
count of zero (0). The ENDLIST element always follows a property
list, even when the length is determined.

Element code 11 (ENDLIST) is the end of list element. It marks the
end of the corresponding list or property list.

   Structure Sharing

     When messages are batched in message-bags for transmission, it may
     often be the case that the same document will be sent to more than
     one recipient. Since the document portion can usually be expected
     to be the major part of the message, much repeated data would be
     sent if a copy of the document for each recipient were to be shipped
     in the message-bag.

     To avoid this redundancy, messages may be assembled in the
     message-bag so that actual data appears on its first occurrence and
     only references to it appear in later occurrences. When data is
     shared, the first occurrence of the data will be tagged, and later
     locations where the data should appear will only reference the
     earlier tagged location. All references to copied data point to
     earlier locations in the message-bag. The data to be retrieved is
     indicated by the tag.

     This is a very general sharing mechanism. PLEASE NOTE THAT THE MPM
     WILL NOT SUPPORT THE FULL USE OF THIS MECHANISM. THE MPM WILL ONLY
     SUPPORT SHARING OF WHOLE DOCUMENTS. No other level of sharing will
     be supported by the MPMs.

     This sharing mechanism may be used within a document as long as all
     references refer to tags within the same document.

     Sharing is implemented by placing a share-tag on the first
     occurrence of the data to be shared, and placing a share-reference
     at the locations where copies of that data should occur.

                            +------+------+------+
       12 Share Tag | 12 | share-index |
                            +------+------+------+

                            +------+------+------+
       13 Share Reference | 13 | share-index |
                            +------+------+------+

     Element code 12 (S-TAG) is a share tag element. The two octets
     following the type-octet specify the shared data identification code
     for the following data element. Note that s-tag is not a DATA
     element, in the sense that data elements encode higher level
     objects.

     Element code 13 (S-REF) is a share reference element. The two

octets following the type-octet specify the referenced shared data
identification code.

An example of using this mechanism is

   ((<a>, <b>) (<c>, <b>))

could be coded as follows to share <b>

   ((<a>, <s-tag-1><b>) (<c>, <s-ref-1>))

To facilitate working with structures which may contain shared data,
the two high-order bits of the list and property list element codes
are reserved for indicating if the structure contains data to be
shared or contains a reference to shared data. That is, if the
high-order bit of the list or property list element code octet is
set to one then the property list contains a share-reference to
shared data. Or, if the second high-order bit is set to one the
structure contains a share-tag for data to be shared.

The example above is now repeated in detail showing the use of the
high-order bits.


```
     +------+------+------+------+------+------+------+------+
     |11 - 9|01 - 9|  <a> |  12  |  0   |  1   |  <b> |  11  |
     +------+------+------+------+------+------+------+------+
                   +------+------+------+------+------+------+------+
                   |10 - 9|  <c> |  13  |  0   |  1   |  11  |  11  |
                   +------+------+------+------+------+------+------+
```

It is not considered an error for an element to be tagged but not
referenced.

A substructure with internal sharing may be created. If such a
substructure is closed with respect to sharing -- that is, all
references to its tagged elements are within the substructure --
then there is no need for the knowledge of the sharing to propagate
up the hierarchy of lists. For example, if the substructure is:

   00-LIST (a b c b)

which with sharing is:

   11-LIST (a T1:b c R1)

When this substructure is included in a large structure the high

   order bits can be reset since the substructure is closed with
   respect to sharing. For example:

      00-LIST (x 11-LIST (a T1:b c R1) y)

   Note: While sharing adds transmission and memory efficiency, it is
   costly in processing to separate shared elements. This is the main
   reason for restricting the sharing supported by the MPM. At some
   later time these restrictions may be eased.

   It is possible to create loops, "strange loops" and "tangled
   hierarchies" using this mechanism [32]. The MPM will not check for
   such improper structures within documents, and will not deliver
   messages involved in such structures between documents.

   If an encryption scheme is used to ensure the privacy of
   communication it is unlikely that any parts of the message can be
   shared. This is due to the fact that in most case the encryption
   keys will be specific between two individuals. There may be a few
   cases where encrypted data may be shared. For example, all the
   members of a committee may use a common key when acting on committee
   business, or in a public key scheme a document may be "signed" using
   the private key of the sender and inspected by anyone using the
   public key of the sender.

4. OTHER ISSUES

This section discusses various other issues that need to be dealt with
in a computer message system.

4.1. Accounting and Billing

   Accounting and billing must be performed by the MPM. The charge to
   the user by the message delivery system must be predictable, and so
   cannot depend on the actual cost of sending a particular message which
   incurs random delays, handling and temporary storage charges. Rather,
   these costs must be aggregated and charged back to the users on an
   average cost basis. The user of the service may be charged based on
   the destination or distance, the length of the message, type of
   service, or other parameters selected as the message is entered into
   the delivery system, but must not depend on essentially random
   handling by the system of the particular message.

   This means it is pointless to have each message carry an accumulated
   charge (or list of charges). Rather, the MPM will keep a log of
   messages handled and periodically bill the originators of those
   messages.

   It seems that the most reasonable scheme is to follow the practice of
   the international telephone authorities. In such schemes the
   authority where the message originates bills the user of the service
   for the total charge. The authorities assist each other in providing
   the international message transfer and the authorities periodically
   settle any differences in accounts due to an imbalance in
   international traffic.

   Thus the MPMs will keep logs of messages handled and will periodically
   charge their neighboring MPM for messages handled for them. This
   settlement procedure is outside the message system and between the
   administrators of the MPMs.

   As traffic grows it will be impractical to log every message
   individually. It will be necessary to establish categories of
   messages (e.g., short, medium, large) and only count the number in
   each category.

   The MPM at the source of the message will have a local means of
   identifying the user to charge for the message delivery service. The
   relay and destination MPMs will know which neighbor MPMs to charge (or
   settle with) for delivery of their messages.

4.2. Addressing and Routing

   The mailbox provides for many types of address information. The MPMs
   in the ARPA internet can most effectively use the internet address
   [2]. The use of other address information is not yet very clear.
   Some thoughts on addressing issues may be found in the references
   [33,34,35].

   An MPM sometimes must make a routing decision when it is acting as a
   relay-MPM (or source MPM). It must be able to use the information
   from the mailbox to determine to which of its neighbor MPMs to send
   the message. One way this might be implemented is to have a table of
   destination networks with corresponding neighbor MPM identifiers to
   use for routing toward that network.

   It is not expected that such routing tables would be very dynamic.
   Changes would occur only when new MPMs came into existence or MPMs
   went out of service for periods of days.

   Even with relatively slowly changing routing information the MPMs need
   an automatic mechanism for adjusting their routing tables. The
   routing problem here is quite similar to the problem of routing in a
   network of packet switches such as the ARPANET IMPs or a set of
   internet gateways. A great deal of work has been done on such
   problems and many simple schemes have been found faulty. There are
   details of these procedures which may become troublesome when the
   number of nodes grows beyond a certain point or the frequency of
   update exchanges gets large.

   A basic routing scheme is to have a table of <network-name,
   mpm-identifier> pairs. The MPM could look up the network name found
   in the mailbox of the message and determine the internet
   mpm-identifier of the next MPM to which to route the message. To
   permit automatic routing updates another column would be added to
   indicate the distance to the destination. This could be measured in
   several ways, for example, the number of relay MPM (or hops) to the
   final destination. In this case each entry in the table is a triple
   of <network-name, mpm-identifier, distance>.

   To update the routing information when changes occur an MPM updates
   its table. It then sends to each next MPM in its table a table of
   pairs <network-name, distance>, which say in effect "I can get a
   message to each of these networks with "cost" distance." An MPM which
   receives such an update will add to all the distances the distance to
   the MPM sending the update (e.g., one hop) and compare the information
   with its own table.

If the update information shows that the distance to a destination
network is now smaller via the MPM which sent the update, the MPM
changes its own table to reflect the better route, and the new
distance. If the MPM has made changes in its table it sends update
information to all the MPMs listed as next-MPMs in its table.

One further feature is that when a new network comes into existence an
entry must be added to the table in each MPM. The MPMs should
therefore expect the case that update information may contain entries
which are new networks, and in such an event add these entries to
their own tables.

When a new MPM comes into existence it will have an initial table
indicating that it is a good route (short distance) to the network it
is in, and will have entries for a few neighbor networks. It will
send an initial "update" to those neighbor MPMs which will respond
with more complete tables, thus informing the new MPM of routes to
many networks.

This routing update mechanism is a simple minded scheme and may have
to be replaced as the system of MPMs grows. In addition it ignores
the opportunity for MPMs to use other information (besides destination
network name) for routing. MPMs may have tables that indicate
next-MPMs based on city, telephone number, organization, or other
categories of information.

4.3. Encryption

It is straightforward to add the capability to have the document
portion of messages either wholly or partially encrypted. An
additional basic data element is defined to carry encrypted data. The
data within this element may be composed of other elements, but that
could only be perceived after the data was decrypted.

```
                   +------+------+------+------+
     14 Encrypt    |  14  |    octet count    |
                   +------+------+------+------+

                     +------+------+------+-------
                     |alg id|  key id    | Data ...
                     +------+------+------+--------
```

Element code 14 (ENCRYPT) is used to encapsulate encrypted data. The
format is the one-octet type code, the three-octet octet count, a
one-octet algorithm identifier, a two-octet key identifier, and count
octets of data. Use of this element indicates that the data it

Postel   [Page 37]

contains is encrypted. The encryption scheme is indicated by the
algorithm identifier, and the key used is indicated by the key
identifier (this is not the key itself). The NBS Data Encryption
Standard (DES) [36], public key encryption [37,38,39], or other
schemes may be used.

To process this data element, the user is asked for the appropriate
key and the data can then be decrypted. The data thus revealed will
be in the form of complete data element fields. Encryption cannot
occur over a partial field. The revealed data is then processed
normally.

Note that there is no reason why all fields of a document could not be
encrypted including all document header information such as From,
Date, etc.

5.   THE MPM: A POSSIBLE ARCHITECTURE

The heart of the internet message system is the MPM which is responsible
for routing and delivering messages. Each network must have at least
one MPM. These MPMs are logically connected together, and internet mail
is always transferred along logical channels between them. The MPMs
interface with existing local message systems.

Since the local message system may be very different from the internet
system, special programs may be necessary to convert incoming internet
messages to the local format. Likewise, messages outgoing to other
networks may be converted to the internet format and sent via the MPMs.

5.1. Interfaces

  User Interface

    It is assumed that the interface between the MPM and the UIP
    provides for passing data structures which represent the document
    portion of the message. In addition, this interface must pass the
    delivery address information (which becomes the information in the
    mailbox field of the command). It is assumed that the information
    is passed between the UIP and the MPM via shared files, but this is
    not the only possible mechanism. These two processes may be more
    strongly coupled (e.g., by sharing memory), or less strongly coupled
    (e.g., by communicating via logical channels).

    When a UIP passes a document and a destination address to the MPM,
    the MPM assigns a transaction-number and forms a message to send.
    The MPM must record the relationship between the transaction-number,
    the document, and the UIP, so that it can inform the UIP about the
    outcome of the delivery attempt for that document when the
    acknowledgment message is received at some later time.

    Assuming a file passing mode of communication between the UIP and
    the MPM the sending and receiving of mail might involve the
    following interactions:

      A user has an interactive session with a UIP to compose a document
      to send to a destination (or list of destinations). When the user
      indicates to the UIP that the document is to be sent, the UIP
      places the information into a file for the MPM. The UIP may then
      turn to the next request of the user.

      The MPM finds the file and extracts the the information. It
      creates a message, assigning a transaction-number and forming a
      deliver command. The MPM records the UIP associated with this
      message. The MPM sends the message toward the destination.

Postel   [Page 39]

When the MPM receives a deliver message from another MPM addressed
to a user in its domain, it extracts the document and puts it into
a file for the UIP associated with the destination user. The MPM
also sends an acknowledge message to the originating MPM.

When the MPM receives an acknowledgment for a message it sent, the
MPM creates a notification for the associated UIP and places it in
a file for that UIP.

The format of these files is up to each UIP/MPM interface pair.
One reasonable choice is to use the same data structures used in
the MPM-MPM communication.

Communication Interface

It is assumed here that the MPMs use an underlying communication
system, and TCP [3] has been taken as the model. In particular, the
MPM is assumed to be listening for a TCP connection on a TCP port,
i.e., it is a server process. The port is either given explicitly
in the mpm-identifier or takes the default vaule 45 (55 octal) [4].
Again, this is not intended to limit the implementation choices;
other forms of interprocess communication are allowed, and other
types of physical interconnection are permitted. One might even use
dial telephone calls to interconnect MPMs (using suitable protocols
to provide reliable communication) [12,19,20,21].

5.2. The MPM Organization

Messages in the internet mail system are transmitted in lists called
message-bags (or simply bags), each bag containing one or more
messages. Each MPM is expected to implement functions which will
allow it to deliver local messages it receives and to forward
non-local ones to other MPMs presumably closer to the message's
destination.

Loosely, each MPM can be separated into six components:

  1--Acceptor

    Receives incoming message-bags, from other MPMs, from UIPs, or
    from conversion programs.

  2--Message-Bag Processor

    Splits a bag into these three portions:

a.   Local Host Messages
        b.   Local Net Messages
        c.   Foreign Net Messages

   3--Local Host Delivery

      Delivers local host messages, may call on conversion program.

   4--Local Net Delivery

      Delivers local net messages, may call on conversion program.

   5--Foreign Net Router

      Forms message-bags for transmission to other MPMs and determines
      the first step in the route.

   6--Foreign Net Sender

      Activates transmission channels to other MPMs and sends
      message-bags to foreign MPMs.

If the local net message system uses the protocol of the MPMs, then
there need be no distinction between local net and foreign net
delivery procedures.

All of these components can be thought of as independent. The
function of the Acceptor is to await incoming message-bags and to
insert them into the Bag-Input Queue.

The Bag-Input queue is read by the message-bag Processor which will
separate and deliver suitable portions of the message-bags it
retrieves from the queue to one of three queues:

   a.   Local Host Queue
   b.   Local Net Queue
   c.   Foreign Net Queue

When an MPM has a message to send to another MPM, it must add its own
handling-stamp to the trace field of the command. The trace then
becomes a record of the route the message has taken. An MPM should
examine the trace field to see if the message is in a routing loop.
All commands require the return of the trace as a trail in the
matching reply command.

All of these queues have as elements complete message-bags created by
selecting messages from the input message-bags.

The Local Host queue serves as input to the Local Host Delivery
process. This component is responsible for delivering messages to its
local host. It may call on a conversion program to reformat the
messages into a form the local protocol will accept. This will
probably involve such things as copying shared information.

The Local Net queue serves as input to the Local Net Delivery process.
This component is responsible for delivering messages to other hosts
on its local net. It must be capable of handling whatever error
conditions the local net might return, and should include the ability
to retransmit. It may call on a conversion program to reformat the
messages into a form the local protocol will accept. This will
probably involve such things as copying shared information.

The other two processes are more closely coupled. The Foreign Net
Router takes its input bags from the Foreign Net Queue. From the
internal information it contains, it determines which of the MPMs to
which it is connected should receive the bag.

It then places the bag along with the routing information into the
Send Mail Queue. The Foreign Net Sender retrieves it from that queue
and transmits it across a channel to the intended foreign MPM. The
Sender aggregates messages to the same next MPM into a bag.

The Foreign Net Router should be capable of receiving external input
to its routing information table. This may come from the Foreign Net
Sender in the case of a channel going down, requiring a decision to
either postpone delivery or to determine a new route. The Router is
responsible for maintaining sufficient information to determine where
to send any incoming message-bag.

Forwarding

  An MPM may have available information on the correct mailboxes of
  users which are not at its location. This information, called a
  forwarding data base, may be used to return the correct address in
  response to a probe command, or to actually forward a deliver
  command (if allowed by the type of service).

  Because such forwarding may cause the route of a message to pass
  through an MPM already on the trace of this message, only the
  portion of the trace back to the most recent forward action should
  be used for loop detection by a relay relaying MPM, and only the
  forward action entries in the trace should be checked by a
  forwarding MPM.

Implementation Recommendations

Transaction numbers can be assigned sequentially, with wrap around
when the highest value is reached. This should ensure that no
message with a particular transaction number from this source is in
the network when another instance of this transaction number is
chosen.

The processing to separate shared elements when the routes of the
shared elements diverge while still preserving the sharing possible
appears to be an O(N*M**2) operation where N is the number of
distinct objects in a message which may be shared across message
boundaries and M is the number of messages in the bag.

Also note that share-tags may be copied into separate message bags
which are not referenced. These could be removed with another pass
over the message bag.

6. EXAMPLES & SCENARIOS

Example 1: Message Format

  Suppose we want to send the following message:

    Date: 1979-03-29-11:46-08:00
    From: Jon Postel <Postel@ISIE>
    Subject: Meeting Thursday
    To: Danny Cohen <Cohen@USC-ISIB>
    CC: Linda

    Danny:

    Please mark your calendar for our meeting Thursday at 3 pm.

    --jon.

  It will be encoded in the structured format. The following will
  present successive steps in the top down generation of this message.
  The actual document above will not be shown in the coded form.

  1. message

  2. (identification, command, document)

  3. (ID:(mpm-identifier, transaction-number),
       CMD:(MAILBOX:mailbox, OPERATION:operation,
                                        arguments, TRACE:trace),
       DOC:<<document>>)

  4. (ID:(mpm-identifier, transaction-number),
       CMD:(MAILBOX:mailbox, OPERATION:operation,
                           TYPE-OF-SERVICE:regular, TRACE:trace),
       DOC:<<document>>)

  5. (ID:(MPM:(IA:12,1,0,52,0,45), TRANSACTION:37),
       CMD:(MAILBOX:(MPM:(IA:12,3,0,52,0,45),
                    NET:ARPA,
                    HOST:ISIB,
                    PORT:45,
                    USER:Cohen),
            OPERATION:DELIVER,
            TYPE-OF-SERVICE:REGULAR,
            TRACE:(MPM:(IA:12,1,0,52,0,45)
                    DATE:1979-03-29-11:46-08:00,
                    ACTION:ORIGIN)),
       DOC:<<document>>)

```
   6.  PROPLIST:(
         ID:PROPLIST:(
             MPM:PROPLIST:(
                  IA:12,1,0,52,0,45),
                ENDLIST
             TRANSACTION:37)
           ENDLIST,

         CMD:PROPLIST(
             MAILBOX:(PROPLIST:(
                     MPM:PROPLIST(
                           IA:12,3,0,52,0,45),
                         ENDLIST
                     NET:ARPA,
                     HOST:ISIB,
                     PORT:45,
                     USER:Cohen ),
                   ENDLIST
             OPERATION:DELIVER,
             TYPE-OF-SERVICE:REGULAR,
             TRACE:(PROPLIST:MPM:
                     (PROPLIST:
                        IA:12,1,0,52,0,45)
                      ENDLIST
                   DATE:1979-03-29-11:46-08:00,
                   ACTION:ORIGIN)),
                 ENDLIST
         ENDLIST
       DOC:<<document>>)
     ENDLIST
```

Example 2: Delivery and Acknowledgment

   The following are four views of the message of example 1 during the
   successive transmission from the origination MPM, through a relay MPM,
   to the destination MPM, and the return of the acknowledgment, through
   a relay MPM, to the originating MPM.

```
   +----------------------------------------------------------------+
   |                        A          B                            |
   | sending --> originating --> relay --> destination --> receiving |
   |   user          MPM         MPM          MPM            user   |
   |                                                                |
   |                        D          C                            |
   |           originating <-- relay <-- destination                |
   |               MPM         MPM          MPM                     |
   +----------------------------------------------------------------+
```
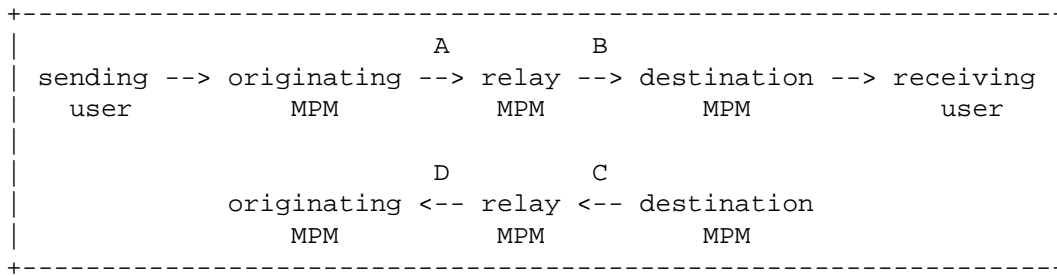
                         Transmission Path

                          Figure 6.

   A.   Between the originating MPM and the relay MPM.

      PROPLIST:
        NAME:"ID",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:"10,1,0,52,0,45"
              ENDLIST
            NAME:"TRANSACTION", INTEGER:37
          ENDLIST
        NAME:"CMD",
          PROPLIST:
            NAME:"MAILBOX",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"NET", NAME:"ARPA"
                NAME:"HOST", NAME:"ISIB"
                NAME:"PORT", NAME:"45"
                NAME:"USER", NAME:"Cohen"
              ENDLIST
            NAME:"OPERATION", NAME:"DELIVER"
            NAME:"TYPE-OF-SERVICE", NAME:"REGULAR"
            NAME:"TRACE",
              LIST:
                PROPLIST:
                  NAME:"MPM",
                    PROPLIST:
                      NAME:"IA", NAME:"10,1,0,52,0,45"
                    ENDLIST
                  NAME:"DATE", NAME:"1979-03-29-11:47.5-08:00"
                  NAME:"ACTION", NAME:"ORIGIN"
                ENDLIST
              ENDLIST
          ENDLIST
        NAME:"DOC", <<document>>
      ENDLIST

   B. Between the relay MPM and the destination MPM.

      PROPLIST:
        NAME:"ID",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:"10,1,0,52,0,45"
              ENDLIST
            NAME:"TRANSACTION", INTEGER:37
          ENDLIST
        NAME:"CMD",
          PROPLIST:
            NAME:"MAILBOX",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"NET", NAME:"ARPA"
                NAME:"HOST", NAME:"ISIB"
                NAME:"PORT", NAME:"45"
                NAME:"USER", NAME:"Cohen"
              ENDLIST
            NAME:"OPERATION", NAME:"DELIVER"
            NAME:"TYPE-OF-SERVICE", NAME:"REGULAR"
            NAME:"TRACE",
              LIST:
                PROPLIST:
                  NAME:"MPM",
                    PROPLIST:
                      NAME:"IA", NAME:"10,1,0,52,0,45"
                    ENDLIST
                  NAME:"DATE", NAME:"1979-03-29-11:47.5-08:00"
                  NAME:"ACTION", NAME:"ORIGIN"
                ENDLIST
                PROPLIST:
                  NAME:"MPM",
                    PROPLIST:
                      NAME:"IA", NAME:"10,2,0,52,0,45"
                    ENDLIST
                  NAME:"DATE", NAME:"1979-03-29-11:48-08:00"
                  NAME:"ACTION", NAME:"RELAY"
                ENDLIST
              ENDLIST
          ENDLIST
        NAME:"DOC", <<document>>

         ENDLIST

   C. Between the destination MPM and the relay MPM.

      PROPLIST:
        NAME:"ID",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:"10,3,0,52,0,45"
              ENDLIST
            NAME:"TRANSACTION", INTEGER:1993
          ENDLIST
        NAME:"CMD",
          PROPLIST:
            NAME:"MAILBOX",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", INTEGER:"10,1,0,52,0,45"
                  ENDLIST
                NAME:"NET", NAME:"ARPA"
                NAME:"HOST", NAME:"ISIE"
                NAME:"PORT", NAME:"45"
                NAME:"USER", NAME:"*MPM*"
              ENDLIST
            NAME:"OPERATION", NAME:"ACKNOWLEDGE"
            NAME:"REFERENCE",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,1,0,52,0,45"
                  ENDLIST
                NAME:"TRANSACTION", INTEGER:37
              ENDLIST
            NAME:"ADDRESS",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", INTEGER:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"USER", NAME:"Cohen"
              ENDLIST
            NAME:"TYPE-OF-SERVICE", NAME:"REGULAR"
            NAME:"ERROR-CLASS", INDEX:0
            NAME:"ERROR-STRING", NAME:"Ok"
            NAME:"TRAIL",

```
            LIST:
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,1,0,52,0,45"
                  ENDLIST
                NAME:"DATE", NAME:"1979-03-29-11:47.5-08:00"
                NAME:"ACTION", NAME:"ORIGIN"
              ENDLIST
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,2,0,52,0,45"
                  ENDLIST
                NAME:"DATE", NAME:"1979-03-29-11:48-08:00"
                NAME:"ACTION", NAME:"RELAY"
              ENDLIST
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"DATE", NAME:"1979-03-29-11:51.567-08:00"
                NAME:"ACTION", NAME:"DESTINATION"
              ENDLIST
            ENDLIST
          NAME:"TRACE",
            LIST:
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"DATE", NAME:"1979-03-29-11:52-08:00"
                NAME:"ACTION", NAME:"ORIGIN"
              ENDLIST
            ENDLIST
        ENDLIST
    ENDLIST
```

   D. Between the relay MPM and the originating MPM.

      PROPLIST:
        NAME:"ID",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:"10,3,0,52,0,45"
              ENDLIST
            NAME:"TRANSACTION", INTEGER:1993
          ENDLIST
        NAME:"CMD",
          PROPLIST:
            NAME:"MAILBOX",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", INTEGER:"10,1,0,52,0,45"
                  ENDLIST
                NAME:"NET", NAME:"ARPA"
                NAME:"HOST", NAME:"ISIE"
                NAME:"PORT", NAME:"45"
                NAME:"USER", NAME:"*MPM*"
              ENDLIST
            NAME:"OPERATION", NAME:"ACKNOWLEDGE"
            NAME:"REFERENCE",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", NAME:"10,1,0,52,0,45"
                  ENDLIST
                NAME:"TRANSACTION", INTEGER:37
              ENDLIST
            NAME:"ADDRESS",
              PROPLIST:
                NAME:"MPM",
                  PROPLIST:
                    NAME:"IA", INTEGER:"10,3,0,52,0,45"
                  ENDLIST
                NAME:"USER", NAME:"Cohen"
              ENDLIST
            NAME:"TYPE-OF-SERVICE", NAME:"REGULAR"
            NAME:"ERROR-CLASS", INDEX:0
            NAME:"ERROR-STRING", NAME:"Ok"
            NAME:"TRAIL",
              LIST:
                PROPLIST:

```
             NAME:"MPM",
               PROPLIST:
                 NAME:"IA", NAME:"10,1,0,52,0,45"
                 ENDLIST
               NAME:"DATE", NAME:"1979-03-29-11:47.5-08:00"
               NAME:"ACTION", NAME:"ORIGIN"
             ENDLIST
             PROPLIST:
               NAME:"MPM",
                 PROPLIST:
                   NAME:"IA", NAME:"10,2,0,52,0,45"
                 ENDLIST
               NAME:"DATE", NAME:"1979-03-29-11:48-08:00"
               NAME:"ACTION", NAME:"RELAY"
             ENDLIST
             PROPLIST:
               NAME:"MPM",
                 PROPLIST:
                   NAME:"IA", NAME:"10,3,0,52,0,45"
                 ENDLIST
               NAME:"DATE", NAME:"1979-03-29-11:51.567-08:00"
               NAME:"ACTION", NAME:"DESTINATION"
             ENDLIST
           ENDLIST
         NAME:"TRACE",
           LIST:
             PROPLIST:
               NAME:"MPM",
                 PROPLIST:
                   NAME:"IA", NAME:"10,3,0,52,0,45"
                 ENDLIST
               NAME:"DATE", NAME:"1979-03-29-11:52-08:00"
               NAME:"ACTION", NAME:"ORIGIN"
             ENDLIST
             PROPLIST:
               NAME:"MPM",
                 PROPLIST:
                   NAME:"IA", NAME:"10,2,0,52,0,45"
                 ENDLIST
               NAME:"DATE", NAME:"1979-03-29-11:52.345-08:00"
               NAME:"ACTION", NAME:"RELAY"
             ENDLIST
           ENDLIST
       ENDLIST
   ENDLIST
```

7. SPECIFICATION SUMMARY

7.1. Message Fields

All keywords used in this protocol are to be recognized independent of
case.

```
action: NAME (one of)
  "ORIGIN" | "RELAY" | "FORWARD" | "DESTINATION"

address: PROPLIST (one of)

  NAME: "MPM", <mpm-identifier>
  NAME: "USER", <user>

    or

  NAME: "NET", <net>
  NAME: "HOST", <host>
  NAME: "PORT", <port>
  NAME: "USER", <user>

answer: BOOLEAN

city: NAME

command: PROPLIST
  NAME: "MAILBOX", <mailbox>
  NAME: "OPERATION", <operation>
  <<arguments>>
  NAME: "ERROR-CLASS", <error-class>   (only in replies)
  NAME: "ERROR-STRING", <error-string> (only in replies)
  NAME: "TRACE", <trace>

country: NAME

document: <<document>>

error-class: INDEX

error-string: NAME

host: NAME
```

```
handling-stamp: PROPLIST
  NAME: "MPM", <mpm-identifier>
  NAME: "DATE", <date>
  NAME: "ACTION", <action>

identification: LIST
  NAME: "MPM", <mpm-identifier>
  NAME: "TRANSACTION", <transaction-number>

internet-address: NAME

mailbox:  PROPLIST (some of)
  NAME: "MPM", <mpm-identifier>
  NAME: "NET", <net>
  NAME: "HOST", <host>
  NAME: "PORT", <port>
  NAME: "USER", <user>
  NAME: "ORG", <organization>
  NAME: "CITY", <city>
  NAME: "STATE", <state>
  NAME: "COUNTRY", <country>
  NAME: "ZIP", <zip-code>
  NAME: "PHONE", <phone-number>
  <<other-items>>

message: PROPLIST
  NAME: "ID", <identification>
  NAME: "CMD", <command>
  NAME: "DOC", <document> (only in deliver)

mpm-identifier: PROPLIST (one of)

  NAME: "IA", <internet-address>

    or

  NAME: "X121", <x121-address>

net: NAME

operation: NAME (one of)
    "DELIVER" | "ACKNOWLEDGE
  | "PROBE"   | "RESPONSE
  | "CANCEL"  | "CANCELED"

organization: NAME

phone-number: NAME
```

```
port: NAME

state: NAME

trace: LIST
  <handling-stamp>
   ...

trail: LIST
  <handling-stamp>
   ...

transaction-number: INTEGER

type-of-service: NAME (one or more of)
   "REGULAR" | "FORWARD" | "GENDEL" | "PRIORITY"

user: NAME

x121-address: NAME

zip-code: NAME
```

7.2. Deliver Message

```
PROPLIST:
  NAME:"ID",
    PROPLIST:
      NAME:"MPM",
        PROPLIST:
          NAME:"IA", NAME:<internet-address>
        ENDLIST
      NAME:"TRANSACTION", INTEGER:<transaction-number>
    ENDLIST
  NAME:"CMD",
    PROPLIST:
      NAME:"MAILBOX",
        PROPLIST:
          NAME:"MPM",
            PROPLIST:
              NAME:"IA", INTEGER:<internet-address>
            ENDLIST
          NAME:"NET", NAME:<net>
          NAME:"HOST", NAME:<host>
          NAME:"PORT", NAME:<port>
          NAME:"USER", NAME:<user>
          NAME:"ORG", NAME:<organization>
          NAME:"CITY", NAME:<city>
          NAME:"STATE", NAME:<state>
          NAME:"COUNTRY", NAME:<country>
          NAME:"ZIP", NAME:<zip-code>
          NAME:"PHONE", NAME:<phone-number>
          <<other-items>>
        ENDLIST
      NAME:"OPERATION", NAME:"DELIVER"
      NAME:"TYPE-OF-SERVICE", NAME:<type-of-service>
      NAME:"TRACE",
        LIST:
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"DATE", NAME:<date>
            NAME:"ACTION", NAME:<action>
          ENDLIST
          ...
        ENDLIST
    ENDLIST
  NAME:"DOC", <<document>>
ENDLIST
```

7.3. Acknowledge Message

```
 PROPLIST:
   NAME:"ID",
     PROPLIST:
       NAME:"MPM",
         PROPLIST:
           NAME:"IA", NAME:<internet-address>
         ENDLIST
       NAME:"TRANSACTION", INTEGER:<transaction-number>
     ENDLIST
   NAME:"CMD",
     PROPLIST:
       NAME:"MAILBOX",
         PROPLIST:
           NAME:"MPM",
             PROPLIST:
               NAME:"IA", INTEGER:<internet-address>
             ENDLIST
           NAME:"USER", NAME:"*MPM*"
           NAME:"NET", NAME:<net>
           NAME:"PORT", NAME:<port>
           NAME:"HOST", NAME:<host>
         ENDLIST
       NAME:"OPERATION", NAME:"ACKNOWLEDGE"
       NAME:"REFERENCE",
         PROPLIST:
           NAME:"MPM",
             PROPLIST:
               NAME:"IA", NAME:<internet-address>
             ENDLIST
           NAME:"TRANSACTION", INTEGER:<transaction-number>
         ENDLIST
       NAME:"ADDRESS",
         PROPLIST:
           NAME:"MPM",
             PROPLIST:
               NAME:"IA", INTEGER:<internet-address>
             ENDLIST
           NAME:"USER", NAME:<user>
         ENDLIST
       NAME:"TYPE-OF-SERVICE", NAME:<type-of-service>
       NAME:"ERROR-CLASS", INDEX:<error-class>
       NAME:"ERROR-STRING", NAME:<error-string>
       NAME:"TRAIL",
         LIST:
           PROPLIST:
             NAME:"MPM",
```

```
                PROPLIST:
                  NAME:"IA", INTEGER:<internet-address>
                ENDLIST
              NAME:"DATE", NAME:<date>
              NAME:"ACTION", NAME:<action>
            ENDLIST
            ...
          ENDLIST
        NAME:"TRACE",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
                  NAME:"IA", INTEGER:<internet-address>
                ENDLIST
              NAME:"DATE", NAME:<date>
              NAME:"ACTION", NAME:<action>
            ENDLIST
            ...
          ENDLIST
        ENDLIST
      ENDLIST
    ENDLIST
```

7.4. Probe Message

```
  PROPLIST:
    NAME:"ID",
      PROPLIST:
        NAME:"MPM",
          PROPLIST:
            NAME:"IA", NAME:<internet-address>
          ENDLIST
        NAME:"TRANSACTION", INTEGER:<transaction-number>
      ENDLIST
    NAME:"CMD",
      PROPLIST:
        NAME:"MAILBOX",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"NET", NAME:<net>
            NAME:"HOST", NAME:<host>
            NAME:"PORT", NAME:<port>
            NAME:"USER", NAME:<user>
            NAME:"ORG", NAME:<organization>
            NAME:"CITY", NAME:<city>
            NAME:"STATE", NAME:<state>
            NAME:"COUNTRY", NAME:<country>
            NAME:"ZIP", NAME:<zip-code>
            NAME:"PHONE", NAME:<phone-number>
            <<other-items>>
          ENDLIST
        NAME:"OPERATION", NAME:"PROBE"
        NAME:"TRACE",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
                  NAME:"IA", INTEGER:<internet-address>
                ENDLIST
              NAME:"DATE", NAME:<date>
              NAME:"ACTION", NAME:<action>
            ENDLIST
            ...
          ENDLIST
      ENDLIST
  ENDLIST
```

7.5. Response Message

```
  PROPLIST:
    NAME:"ID",
      PROPLIST:
        NAME:"MPM",
          PROPLIST:
            NAME:"IA", NAME:<internet-address>
          ENDLIST
        NAME:"TRANSACTION", INTEGER:<transaction-number>
      ENDLIST
    NAME:"CMD",
      PROPLIST:
        NAME:"MAILBOX",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"NET", NAME:<net>
            NAME:"HOST", NAME:<host>
            NAME:"PORT", NAME:<port>
            NAME:"USER", NAME:"*MPM*"
          ENDLIST
        NAME:"OPERATION", NAME:"RESPONSE"
        NAME:"REFERENCE",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:<internet-address>
              ENDLIST
            NAME:"TRANSACTION", INTEGER:<transaction-number>
          ENDLIST
        NAME:"ADDRESS",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"USER", NAME:<user>
          ENDLIST
        NAME:"ERROR-CLASS", INDEX:<error-class>
        NAME:"ERROR-STRING", NAME:<error-string>
        NAME:"TRAIL",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
```

```
                 NAME:"IA", INTEGER:<internet-address>
                ENDLIST
             NAME:"DATE", NAME:<date>
             NAME:"ACTION", NAME:<action>
          ENDLIST
          ...
       ENDLIST
      NAME:"TRACE",
       LIST:
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                 NAME:"IA", INTEGER:<internet-address>
                ENDLIST
             NAME:"DATE", NAME:<date>
             NAME:"ACTION", NAME:<action>
          ENDLIST
          ...
       ENDLIST
     ENDLIST
  ENDLIST
```

   7.6. Cancel Message

```
  PROPLIST:
    NAME:"ID",
      PROPLIST:
        NAME:"MPM",
          PROPLIST:
            NAME:"IA", NAME:<internet-address>
          ENDLIST
        NAME:"TRANSACTION", INTEGER:<transaction-number>
      ENDLIST
    NAME:"CMD",
      PROPLIST:
        NAME:"MAILBOX",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"NET", NAME:<net>
            NAME:"HOST", NAME:<host>
            NAME:"PORT", NAME:<port>
            NAME:"USER", NAME:<user>
            NAME:"ORG", NAME:<organization>
            NAME:"CITY", NAME:<city>
            NAME:"STATE", NAME:<state>
            NAME:"COUNTRY", NAME:<country>
            NAME:"ZIP", NAME:<zip-code>
            NAME:"PHONE", NAME:<phone-number>
            <<other-items>>
          ENDLIST
        NAME:"OPERATION", NAME:"CANCEL"
        NAME:"REFERENCE",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:<internet-address>
              ENDLIST
            NAME:"TRANSACTION", INTEGER:<transaction-number>
          ENDLIST
        NAME:"TRACE",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
                  NAME:"IA", INTEGER:<internet-address>
                ENDLIST
              NAME:"DATE", NAME:<date>
```

```
            NAME:"ACTION", NAME:<action>
          ENDLIST
          ...
        ENDLIST
      ENDLIST
  ENDLIST
```

7.7. Canceled Message

```
  PROPLIST:
    NAME:"ID",
      PROPLIST:
        NAME:"MPM",
          PROPLIST:
            NAME:"IA", NAME:<internet-address>
          ENDLIST
        NAME:"TRANSACTION", INTEGER:<transaction-number>
      ENDLIST
    NAME:"CMD",
      PROPLIST:
        NAME:"MAILBOX",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"NET", NAME:<net>
            NAME:"HOST", NAME:<host>
            NAME:"PORT", NAME:<port>
            NAME:"USER", NAME:"*MPM*"
          ENDLIST
        NAME:"OPERATION", NAME:"CANCELED"
        NAME:"REFERENCE",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", NAME:<internet-address>
              ENDLIST
            NAME:"TRANSACTION", INTEGER:<transaction-number>
          ENDLIST
        NAME:"ADDRESS",
          PROPLIST:
            NAME:"MPM",
              PROPLIST:
                NAME:"IA", INTEGER:<internet-address>
              ENDLIST
            NAME:"USER", NAME:<user>
          ENDLIST
        NAME:"ERROR-CLASS", INDEX:<error-class>
        NAME:"ERROR-STRING", NAME:<error-string>
        NAME:"TRAIL",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
```

```
                NAME:"IA", INTEGER:<internet-address>
                  ENDLIST
                NAME:"DATE", NAME:<date>
                NAME:"ACTION", NAME:<action>
              ENDLIST
              ...
            ENDLIST
        NAME:"TRACE",
          LIST:
            PROPLIST:
              NAME:"MPM",
                PROPLIST:
                  NAME:"IA", INTEGER:<internet-address>
                  ENDLIST
                NAME:"DATE", NAME:<date>
                NAME:"ACTION", NAME:<action>
              ENDLIST
              ...
            ENDLIST
          ENDLIST
      ENDLIST
  ENDLIST
```

7.8. Data Element Summary

```
  CODE    NAME      STRUCTURE                             LENGTH
  ----    ----      ---------                             ------

   0      NOP       CODE(1)                               1

   1      PAD       CODE(1),COUNT(3),DATA(C)              C+4

   2      BOOLEAN   CODE(1),TRUE-FALSE(1)                 2

   3      INDEX     CODE(1),INDEX(2)                      3

   4      INTEGER   CODE(1),INTEGER(4)                    5

   5      EPI       CODE(1),COUNT(3),INTEGER(C)           C+4

   6      BITSTR    CODE(1),COUNT(3),BITS(C/8)            C/8+4

   7      NAME      CODE(1),COUNT(1),NAME(C)              C+2

   8      TEXT      CODE(1),COUNT(3),TEXT(C)              C+4

   9      LIST      CODE(1),COUNT(3),ITEMS(2),DATA(C-2)   C+4

  10      PROPLIST  CODE(1),COUNT(3),PAIRS(1),DATA(C-1)   C+4

  11      ENDLIST   CODE(1)                               1

  12      S-TAG     CODE(1),INDEX(2)                      3

  13      S-REF     CODE(1),INDEX(2)                      3

  14      ENCRYPT   CODE(1),COUNT(3),ALG-ID(1),
                           KEY-ID(2),DATA(C-3)            C+4
```

The numbers in parentheses are the number of octets in the field.

                            REFERENCES

[1]   Cerf, V., "The Catenet Model for Internetworking," Information
      Processing Techniques Office, Defense Advanced Research Projects
      Agency, IEN 48, July 1978.

[2]   Postel, J., "DOD Standard Internet Protocol," USC/Information
      Sciences Institute, IEN 128, NTIS number AD A079730, January 1980.

[3]   Postel, J., "DOD Standard Transmission Control Protocol,"
      USC/Information Sciences Institute, IEN 129, NTIS number AD
      A082609, January 1980.

[4]   Postel, J., "Assigned Numbers," RFC 762, USC/Information Sciences
      Institute, January 1980.

[5]   Feinler, E. and J. Postel, eds., "ARPANET Protocol Handbook,"
      NIC 7104, for the Defense Communications Agency by the Network
      Information Center of SRI International, Menlo Park, California,
      Revised January 1978.

[6]   Neigus, N., "File Transfer Protocol for the ARPA Network,"
      RFC 542, NIC 17759, SRI International, August 1973.

[7]   Bhushan, A., K. Progran, R. Tomlinson, and J. White,
      "Standardizing Network Mail Headers," RFC 561, NIC 18516,
      September 1973.

[8]   Myer, T., and D. Henderson, "Message Transmission Protocol,"
      RFC 680, NIC 32116, 30 April 1975.

[9]   Crocker, D., J. Vittal, K. Progran, and D. Henderson, "Standard
      for the Format of ARPA Network Text Messages," RFC 733, NIC 41952,
      21 November 1977.

[10] Barber, D., and J. Laws, "A Basic Mail Scheme for EIN," INWG 192,
      February 1979.

[11] Braaten, O., "Introduction to a Mail Protocol," Norwegian
      Computing Center, INWG 180, August 1978.

[12] Crocker, D., E. Szurkowski, and D. Farber, "An Internetwork Memo
      Distribution Capability - MMDF," Sixth Data Communications
      Symposium, ACM/IEEE, November 1979.

[13]   Haverty, J., D. Henderson, and D. Oestreicher, "Proposed
       Specification of an Inter-site Message Protocol," 8 July 1975.

[14]   Thomas, R., "Providing Mail Services for NSW Users," BBN NSW
       Working Note 24, Bolt Beranek and Newman, October 1978.

[15]   White, J., "A Proposed Mail Protocol," RFC 524, NIC 17140, SRI
       International, 13 June 1973.

[16]   White, J., "Description of a Multi-Host Journal," NIC 23144, SRI
       International, 30 May 1974.

[17]   White, J., "Journal Subscription Service," NIC 23143, SRI
       International, 28 May 1974.

[18]   Levin, R., and M. Schroeder, "Transport of Electronic Messages
       Through a Network," Teleinformatics 79, Boutmy & Danthine (eds.)
       North Holland Publishing Co., 1979.

[19]   Earnest, L., and J. McCarthy, "DIALNET: A Computer Communications
       Study," Computer Science Department, Stanford University, August
       1978.

[20]   Crispin M., "DIALNET: A Telephone Network Data Communications
       Protocol," DECUS Proceedings, Fall 1979.

[21]   Caulkins, D., "The Personal Computer Network (PCNET) Project: A
       Status Report," Dr. Dobbs Journal of Computer Calisthenics and
       Orthodontia, v.5, n.6, June 1980.

[22]   Postel, J., "NSW Transaction Protocol (NSWTP)," USC/Information
       Sciences Institute, IEN 38, May 1978.

[23]   Haverty, J., "MSDTP -- Message Services Data Transmission
       Protocol," RFC 713, NIC 34739, April 1976.

[24]   Haverty, J., "Thoughts on Interactions in Distributed Services,"
       RFC 722, NIC 36806, 16 September 1976.

[25]   Postel, J., "A Structured Format for Transmission of Multi-Media
       Documents," RFC 767, USC/Information Sciences Institute,
       August 1980.

[26]   ISO-2014, "Writing of calendar dates in all-numeric form,"
       Recommendation 2014, International Organization for
       Standardization, 1975.

   [27] ISO-3307, "Information Interchange -- Representations of time of
         the day," Recommendation 3307, International Organization for
         Standardization, 1975.

   [28] ISO-4031, "Information Interchange -- Representation of local time
         differentials," Recommendation 4031, International Organization
         for Standardization, 1978.

   [29] CCITT-X.121, "International Numbering Plan for Public Data
         Networks," Recommendation X.121, CCITT, Geneva, 1978.

   [30] Postel, J., "NSW Data Representation (NSWB8)," USC/Information
         Sciences Institute, IEN 39, May 1978.

   [31] Cohen, D., "On Holy Wars and a Plea for Peace," IEN 137,
         USC/Information Sciences Institute, 1 April 1980.

   [32] Hofstadter, D., "Godel, Escher, Bach: An Eternal Golden Braid,"
         Basic Books, New York, 1979..

   [33] Harrenstien, K., "Field Addressing," ARPANET Message, SRI
         International, October 1977.

   [34] Postel, J., "Out-of-Net Host Address for Mail," RFC 754,
         USC/Information Sciences Institute, April 1979.

   [35] Shoch, J., "On Inter-Network Naming, Addressing, and Routing,"
         IEEE Computer Society, COMPCON, Fall 1978.

   [36] National Bureau of Standards, "Data Encryption Standard," Federal
         Information Processing Standards Publication 46, January 1977.

   [37] Diffie, W., and M. Hellman, "New Directions in Cryptology," IEEE
         Transactions on Information Theory, IT-22, 6, November 1976.

   [38] Rivest, R., A. Shamir, and L. Adleman, "A Method for Obtaining
         Digital Signatures and Public-Key Cryptosystems" Communications
         of the ACM, Vol. 21, Number 2, February 1978.

   [39] Merkle, R., and M. Hellman, "Hiding Information and Signatures in
         Trapdoor Knapsacks," IEEE Transactions of Information Theory,
         IT-24,5, September 1978.