Internet Reliable Transaction Protocol
Functional and Interface Specification

STATUS OF THIS MEMO

   This RFC is being distributed to members of the DARPA research
   community in order to solicit their reactions to the proposals
   contained in it. While the issues discussed may not be directly
   relevant to the research problems of the DARPA community, they may be
   interesting to a number of researchers and implementors. This RFC
   suggests a proposed protocol for the ARPA-Internet community, and
   requests discussion and suggestions for improvements. Distribution
   of this memo is unlimited.

ABSTRACT

   The Internet Reliable Transaction Protocol (IRTP) is a transport
   level host to host protocol designed for an internet environment. It
   provides reliable, sequenced delivery of packets of data between
   hosts and multiplexes/demultiplexes streams of packets from/to user
   processes representing ports. It is simple to implement, with a
   minimum of connection management, at the possible expense of
   efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ICMP   Internet Control Message Protocol
IP   Internet Protocol
IRTP   Internet Reliable Transaction Protocol
RDP   Reliable Data Protocol
TCP   Transmission Control Protocol
UDP   User Datagram Protocol

CHAPTER 1 - INTRODUCTION

   The Internet Reliable Transaction Protocol (IRTP) is a full duplex,
   transaction oriented, host to host protocol which provides reliable
   sequenced delivery of packets of data, called transaction packets.

   Note: throughout this document the terms host and internet address
   are used interchangeably.

   1.1 Purpose

      The IRTP was designed for an environment in which one host will
      have to maintain reliable communication with many other hosts. It
      is assumed that there is a (relatively) sporadic flow of
      information with each destination host, however information flow
      may be initiated at any time at either end of the connection. The
      nature of the information is in the form of transactions, i.e.
      small, self contained messages. There may be times at which one
      host will want to communicate essentially the same information to
      all of its known destinations as rapidly as possible.

      In effect, the IRTP defines a constant underlying connection
      between two hosts. This connection is not established and broken
      down, rather it can be resynchronized with minimal loss of data
      whenever one of the hosts has been rebooted.

      Due to the lack of connection management, it is desirable that all
      IRTP processes keep static information about all possible remote
      hosts. However, the IRTP has been designed such that minimal state
      information needs to be associated with each host to host pair,
      thereby allowing one host to communicate with many remote hosts.

      The IRTP is more complex than UDP in that it provides reliable,
      sequenced delivery of packets, but it is less complex than TCP in
      that sequencing is done on a packet by packet (rather than
      character stream) basis, and there is only one connection defined
      between any two internet addresses (that is, it is not a process
      to process protocol.)

   1.2 Underlying Mechanisms

      The IRTP uses retransmission and acknowledgments to guarantee
      delivery. Checksums are used to guarantee data integrity and to
      protect against misrouting. There is a host to host
      synchronization mechanism and packet sequencing to provide
      duplicate detection and ordered delivery to the user process. A
      simple mechanism allows IRTP to multiplex and demultiplex streams

of transaction packets being exchanged between multiple IRTP users
on this host and statically paired IRTP users on the same remote
host.

1.3 Relationship to Other Protocols

The IRTP is designed for use in a potentially lossy internet
environment. It requires that IP be under it. The IP protocol
number of IRTP is 28.

Conversely, IRTP provides a reliable transport protocol for one or
more user processes. User processes must have well-known IRTP
port numbers, and can communicate only with matching processes
with the same port number. (Note that the term port refers to a
higher level protocol. IRTP connections exists between two hosts,
not between a host/port and another host/port.)

These relationships are depicted below.

```
    +--------+    +--------+    +-----------+
    | port a |....| port x |    | TCP users |    Application Level
    +--------+    +--------+    +-----------+
        |             |            | ... |
      +--------------+           +-----------+
      |    IRTP      |           |    TCP    |    Host Level
      +--------------+           +-----------+
             |                        |
    +--------------------------------------+
    |     Internet Protocol and ICMP       |    Internet Level
    +--------------------------------------+
                    |
    +--------------------------------------+
    |      Local Network Protocol          |    Network Level
    +--------------------------------------+
```
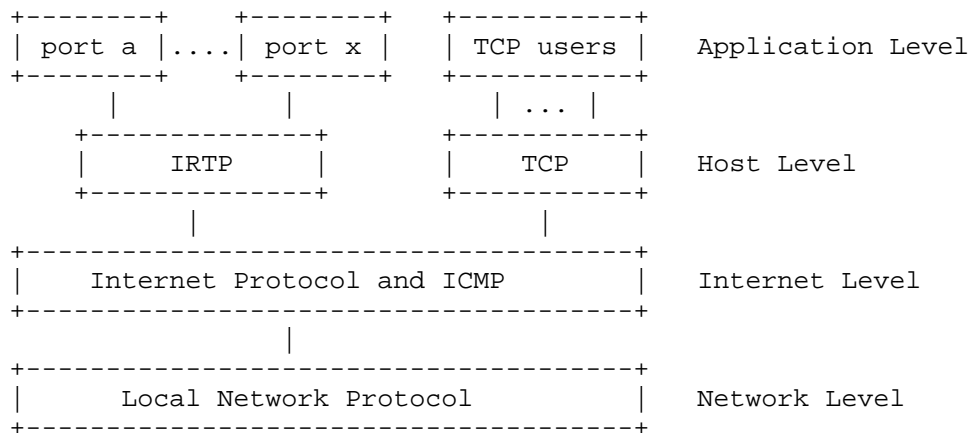
Figure 1-1. Relationship of IRTP to Other Protocols

CHAPTER 2 - IRTP HEADERS

    2.1 Header Format

      Each IRTP packet is preceded by an eight byte header depicted
      below. The individual fields are described in the following
      sections.

            0   7 8   15 16   31
            +--------+--------+--------+--------+
            | packet | port | sequence |
            | type | number | number |
            +--------+--------+--------+--------+
            | length | checksum |
            | | |
            +----------------+----------------+
            | |
            | optional data octets |
            + ... |

        Figure 2-1. IRTP Header Format
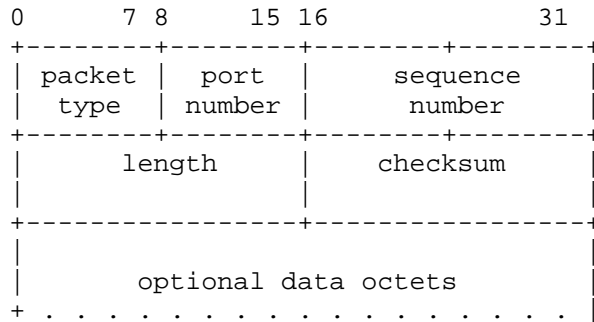
    2.2 Packet Type

      Five packet types are defined by the IRTP. These are:

      packet type   numeric code

      SYNCH   0
      SYNCH ACK   1
      DATA   2
      DATA ACK   3
      PORT NAK   4

      The use of individual packet types is discussed in MODEL OF
      OPERATION.

    2.3 Port Number

      This field is used for the multiplexing and demultiplexing of
      packets from multiple user processes across a single IRTP
      connection. Processes which desire to use IRTP must claim port
      numbers. A port number represents a higher level protocol, and
      data to/from this port may be exchanged only with a process which
      has claimed the same port number at a remote host. A process can
      claim multiple port numbers, however, only one process may claim
      an individual port number. All port numbers are well-known.

2.4 Sequence Number

   For each communicating pair of hosts, there are two sequence
   numbers defined, which are the send sequence numbers for the two
   ends. Sequence numbers are treated as unsigned 16 bit integers.
   Each time a new transaction packet is sent, the sender increases
   the sequence number by one. Initial sequence numbers are
   established when the connection is resynchronized (see Section
   4.3.)

2.5 Length

   The length is the number of octets in this transaction packet,
   including the header and the data. (This means that the minimum
   value of the length is 8.)

2.6 Checksum

   The checksum is the 16-bit one's complement of the one's
   complement sum of the IRTP header and the transaction packet data
   (padded with an octet of zero if necessary to make an even number
   of octets.)

CHAPTER 3 - INTERFACES

   3.1 User Services Provided by IRTP

      The exact interface to the TRTP from the using processes is
      implementation dependent, however, IRTP should provide the
      following services to the using processes.

         o user processes must be able to claim a port number

         o users must be able to request that data be sent to a
            particular port at an internet address (the port must be one
            which the user has claimed)

         o users must be able to request transaction data from a
            particular port at any (unspecified) remote internet address
            (the port must be one which the user has claimed)

         o if a port is determined to be unreachable at a particular
            destination, the using process which has claimed that port
            should be notified

      In addition to these minimal data transfer services, a particular
      implementation may want to have a mechanism by which a
      "supervisory" (that is, port independent) module can define
      dynamically the remote internet addresses which are legal targets
      for host to host communication by this IRTP module. This
      mechanism might be internal or external to the IRTP module itself.

   3.2 IP Services Expected by IRTP

      IRTP expects a standard interface to IP through which it can send
      and receive transaction packets as IP datagrams. In addition, if
      possible, it is desirable that IP or ICMP notify IRTP in the event
      that a remote internet address is unreachable.

      If the IP implementation (including ICMP) is able to notify IRTP
      of source quench conditions, individual IRTP implementations may
      be able to perform some dynamic adjustment of transmission
      characteristics.

CHAPTER 4 - MODEL OF OPERATION

   The basic operation of IRTP is as follows. The first time two hosts
   communicate (or the first time after both have simultaneously
   failed,) synchronization is established using constant initial
   sequence numbers (there is a sequence number for each direction of
   transmission). The TCP "quiet time" is used following reboots to
   insure that this will not cause inaccurate acknowledgment processing
   by one side or the other.

   Once synchronization has been achieved data may be passed in both
   directions. Each transaction packet has a 16 bit sequence number.
   Sequence numbers increase monotonically as new packets are generated.
   The receipt of each sequence number must be acknowledged, either
   implicitly or explicitly. At most 8 unacknowledged packets may be
   outstanding in one direction. This number (called MAXPACK) is fixed
   for all IRTP modules. Unacknowledged packets must be periodically
   retransmitted. Sequence numbers are also used for duplicate
   detection by receiving IRTP modules.

   If synchronization is lost due to the failure of one of the
   communicating hosts, after a reboot that host requests the remote
   host to communicate sequence number information, and data transfer
   continues.

4.1 State Variables

   Each IRTP is associated with a single internet address. The
   synchronization mechanism of the IRTP depends on the requirement
   that each IRTP module knows the internet addresses of all modules
   with which it will communicate. For each remote internet address,
   an IRTP module must maintain the following information (called the
   connection table):

   rem_addr   (32 bit remote internet address)
   conn_state   (8 bit connection state)
   snd_nxt   (16 bit send sequence number)
   rcv_nxt   (16 bit expected next receive sequence number)
   snd_una   (16 bit first unacknowledged sequence number)

   In addition to maintaining the connection tables defined above, it
   is required that every IRTP module have some mechanism which
   generates "retransmission events" such that SYNCH packets are
   periodically retransmitted for any connection in synch_wait state
   (see Section 4.3), and the appropriate DATA packet is periodically
   retransmitted for any connection in data_transfer state (see
   Section 4.4.2). It is implementation dependent whether this

mechanism is connection dependent, or a uniform mechanism for all
connections, so it has not been made part of the connection state
table. See Chapter 5 for more discussion.

4.2 IRTP Initialization

Whenever a remote internet address becomes known by an IRTP
process, a 2 minute "quiet time" as described in the TCP
specification must be observed before accepting any incoming
packets or user requests. This is to insure that no old IRTP
packets are still in the network. In addition, a connection table
is initialized as follows:

    rem_addr   =   known internet address
    conn_state   =   0 = out-of-synch
    snd_nxt   =   0
    rcv_nxt   =   0
    snd_una   =   0

Strictly speaking, the IRTP specification does not allow
connection tables to be dynamically deleted and recreated,
however, if this happens the above procedure must be repeated.
See Chapter 5 for more discussion.

4.3 Host-to-Host Synchronization

An IRTP module must initiate synchronization whenever it receives
a DATA packet or a user request referencing an internet address
whose connection state is out-of-synch. Typically, this will
happen only the first time that internet address is active
following the reinitialization of the IRTP module. A SYNCH packet
as shown below is transmitted. Having sent this packet, the host
enters connection state synch_wait (conn_state = 1). In this
state, any incoming DATA, DATA ACK or PORT NAK packets are
ignored. The SYNCH packet itself must be retransmitted
periodically until synchronization has been achieved.

4.3.1 Response to SYNCH Packets -

    Whenever a SYNCH packet is received, the recipient, regardless
    of current connection state, is required to to return a SYNCH
    ACK packet as shown below. At this point the recipient enters
    data_transfer state (conn_state = 2).

        4.3.2 Response to SYNCH ACK Packet -

        On receipt of a SYNCH ACK packet, the behavior of the recipient
        depends on its state. If the recipient is in synch_wait state
        the recipient sets rcv_nxt to the sequence number value, sets
        snd_nxt and snd_una to the value in the two-octet data field,
        and enters data_transfer state (conn_state = 2). Otherwise,
        the packet is ignored.
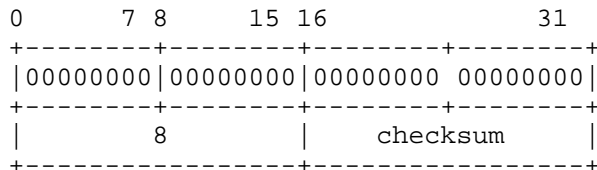
```
        0       7 8     15 16                31
        +--------+--------+--------+--------+
        |00000000|00000000|00000000 00000000|
        +--------+--------+--------+--------+
        |        8        |     checksum     |
        +-----------------+-----------------+
```

        Figure 4-1. SYNCH Packet Format

```
        0       7 8     15 16                31
        +--------+--------+--------+--------+
        |00000001| unused |      snd_una     |
        +--------+--------+--------+--------+
        |        10       |     checksum     |
        +-----------------+-----------------+
        |      rcv_nxt    |
        +-----------------+
```
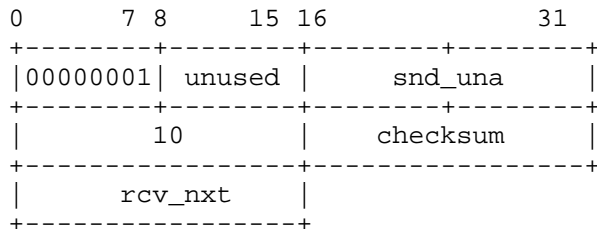
        Figure 4-2. SYNCH ACK Packet Format

    4.4 Transmitting Data

        Once in data_transfer state DATA, DATA ACK and PORT NAK packets
        are used to achieve communication between IRTP processes, subject
        to the constraint that no more than MAXPACK unacknowledged packets
        may be transmitted on a connection at any time. Note that all
        arithmetic operations and comparisons on sequence numbers
        described in this chapter are to be done modulo 2 to the 16.

        4.4.1 Receiving Data From Using Processes -

        User processes may request IRTP to send packets of at most 512
        user data octets to a remote internet address and IRTP port.
        When such a request is received, the behavior of the IRTP
        depends on the state of the connection with the remote host and
        on implementation dependent considerations. If the connection

between this IRTP module and the remote host is not in
data_transfer state, that state must be achieved (see Section
4.3) before acting on the user request.

Once the connection is in data_transfer state, the behavior of
the IRTP module in reaction to a write request from a user is
implementation dependent. The simplest IRTP implementations
will not accept write requests when MAXPACK unacknowledged
packets have been sent to the remote connection and will
provide interested users a mechanism by which they can be
notified when the connection is no longer in this state, which
is called flow controlled. Such implementations are called
blocking IRTP implementations. These implementations check, on
receipt of a write request, to see if the value of snd_nxt is
less than snd_una+MAXPACK. If it is, IRTP prepends a DATA
packet header as shown below, and transmits the packet. The
value of snd_nxt is then incremented by one. In addition, the
packet must be retained in a retransmission queue until it is
acknowledged.

```
    0       7 8     15 16                    31
    +--------+--------+--------+--------+
    |00000010|port num|      snd_nxt     |
    +--------+--------+--------+--------+
    |     length       |     checksum     |
    +----------------+----------------+
    |            data octet(s)          |
    + . . . . . . . . . . . . . . . . . +
```
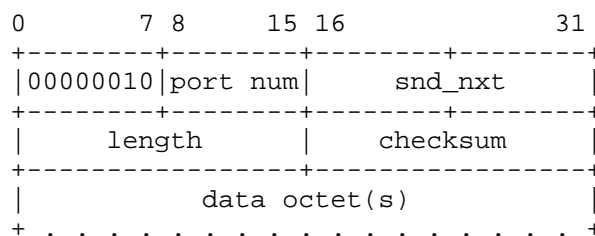
    Figure 4-3. DATA Packet Format

Other implementations may allow (some number of) write requests
to be accepted even when the connection is flow controlled.
These implementations, called non-blocking IRTP
implementations, must maintain, in addition to the
retransmission queue for each connection, a queue of accepted
but not yet transmitted packets, in order of request. This is
called the pretransmission queue for the connection.

When a non-blocking implementation receives a write request, if
the connection is not flow controlled, it behaves exactly as a
blocking IRTP. Otherwise, it prepends a DATA packet header
without a sequence number to the data, and appends the packet
to the pretransmission queue. Note that in this case, snd_nxt
is not incremented. The value of snd_nxt is incremented only
when a packet is transmitted for the first time.

4.4.2 Packet Retransmission -

The IRTP protocol requires that the transaction packet with
sequence number snd_una be periodically retransmitted as long
as there are any unacknowledged, but previously transmitted,
packets (that is, as long as the value of snd_una is not equal
to that of snd_nxt.)

The value of snd_una increases over time due to the receipt of
DATA ACK or PORT NAK packets from a remote host (see Sections
4.5.3 and 4.5.4 below). When either of these packet types is
received, if the incoming sequence number in that packet is
greater than the current value of snd_una, the value of snd_una
is set to the incoming sequence number in that packet. Any
DATA packets with sequence number less than the new snd_una
which were queued for retransmission are released.

(If this is a non-blocking IRTP implementation, for each DATA
packet which is thus released from the retransmission queue,
the earliest buffered packet may be transmitted from the
pretransmission queue, as long as the pretransmission queue is
non-empty. Prior to transmitting the packet, the current value
of snd_nxt is put in the sequence number field of the header.
The value of snd_nxt is then incremented by one.)

Finally, if the acknowledgment is a PORT NAK, the user process
with the nacked port number should be notified that the remote
port is not there.

It is also to be desired, though it is not required, that IRTP
modules have some mechanism to decide that a remote host is not
responding in order to notify user processes that this host is
apparently unreachable.

4.5 Receiving Data

When an IRTP module in data_transfer state receives a DATA packet,
its behavior depends on the port number, sequence number and
implementation dependent space considerations.

DATA ACK and PORT NAK packets are used to acknowledge the receipt
of DATA packets. Both of these acknowledgment packets acknowledge
the receipt of all sequence numbers up to, but not including, the
sequence number in their headers. Note that this value is denoted
"rcv_nxt" in the figures below. This number is the value of
rcv_nxt at the source of the acknowledgment packet when the
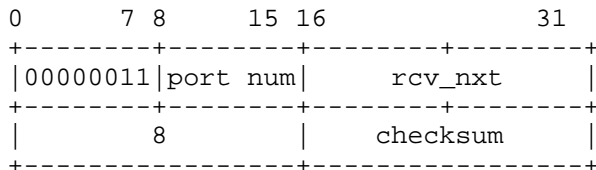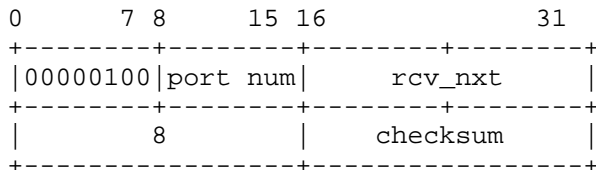acknowledgment was generated.

```
0       7 8      15 16                  31
+--------+--------+--------+--------+
|00000011|port num|     rcv_nxt      |
+--------+--------+--------+--------+
|        8        |     checksum     |
+-----------------+-----------------+
```

Figure 4-4.   DATA ACK Packet Format

```
0       7 8      15 16                  31
+--------+--------+--------+--------+
|00000100|port num|     rcv_nxt      |
+--------+--------+--------+--------+
|        8        |     checksum     |
+-----------------+-----------------+
```

Figure 4-5.   PORT NAK Packet Format

It is not required that a receiving IRTP implementation return an
acknowledgment packet for every incoming DATA packet, nor is it
required that the acknowledged sequence number be that in the most
recently received packet. The exact circumstances under which
DATA ACK and PORT NAK packets are sent are detailed below. The
net effect is that every sequence number is acknowledged, a sender
can force reacknowledgment if an ACK is lost, all acknowledgments
are cumulative, and no out of order acknowledgments are permitted.

4.5.1 Receive and Acknowledgment Windows -

   Each IRTP module has two windows associated with the receive
   side of a connection. For convenience in the following
   discussion these are given names. The sequence number window

   rcv_nxt-MAXPACK =< sequence number < rcv_nxt

   is called the acknowledge window. All sequence numbers within
   this window represent packets which have previously been acked
   or nacked, however, the ack or nack may have been lost in the
   network.

   The sequence number window

   rcv_nxt =< sequence number < rcv_nxt+MYRCV =< rcv_nxt+MAXPACK

   is called the receive window. All sequence numbers within this
   window represent legal packets which may be in transit,
   assuming that the remote host has received acks for all packets

in the acknowledge window. The value of MYRCV depends on the
implementation of the IRTP. In the simplest case this number
will be one, effectively meaning that the IRTP will ignore any
incoming packets not in the acknowledge window or not equal to
rcv_nxt. If the IRTP has enough memory to buffer some incoming
out-of-order packets, MYRCV can be set to some number =<
MAXPACK and a more complex algorithm can be used to compute
rcv_nxt, thereby achieving potentially greater efficiency.
Note that in the latter case, these packets are not
acknowledged until their sequence number is less than rcv_nxt,
thereby insuring that acknowledgments are always cumulative.
(See 4.5.4 below.)

4.5.2 Invalid Packets -

When an IRTP receives a DATA packet, it first checks the
sequence number in the received packet. If the sequence number
is not within the acknowledge or receive window, the packet is
discarded. Similarly, if the computed checksum does not match
that in the header, the packet is discarded. No further action
is taken.

4.5.3 Sequence Numbers Within Acknowledge Window -

When an IRTP receives an incoming DATA packet whose sequence
number is within the acknowledge window, if the port specified
in the incoming DATA packet is known to this IRTP, a DATA ACK
packet is returned. Otherwise, a PORT NAK is returned.

In both cases, the value put in the sequence number field of
the acknowlegement packet is the current value of rcv_nxt at
the IRTP module which is acknowledging the DATA packet. The
DATA packet itself is discarded.

(Note that the PORT NAK acknowledges reception of all packet
numbers up to rcv_nxt. It NAKs the port number, not the
sequence number.)

4.5.4 Sequence Numbers Within the Receive Window -

If the received sequence number is within the receive window,
rcv_nxt is recomputed. How this is done is implementation
dependent. If MYRCV is one, then rcv_nxt is simply
incremented. Otherwise, rcv_nxt is set to the lowest sequence
number such that all data packets with sequence numbers less

than this number have been received and are buffered at the
receiving IRTP, or have been delivered to their destination
port.

Once rcv_nxt has been recomputed, a DATA ACK or PORT NAK is
returned, depending on whether the port number is known or not
known. The value placed in the sequence number field is the
newly computed value for rcv_nxt.

4.5.5 Forwarding Data to Using Processes -

Whenever an incoming DATA packet has been acknowledged (either
implicitly or explicitly) its header can be stripped off and it
can be queued for delivery to the user process which has
claimed its port number. If the IRTP implementation allows
MYRCV to be greater than one, care must be taken that data
which was originally received out of order is forwarded to its
intended recipient in order of original sequence number.

CHAPTER 5 - IMPLEMENTATION ISSUES

   The preceding chapter was left intentionally vague in certain ways.
   In particular, no explicit description of the use of a timer or
   timers within an IRTP module was given, nor was there a description
   of how timer events should relate to "retransmission events". This
   was done to separate the syntactic and operational requirements of
   the protocol from the performance characteristics of its
   implementation.

   It is believed that the protocol is robust. That is, any
   implementation which strictly conforms to Chapter 4 should provide
   reliable synchronization of two hosts and reliable sequenced transfer
   of transaction data between them. However, different ways of
   defining the notion of a retransmission event can have potentially
   significant impact on the performance of the protocol in terms of
   throughput and in terms of the load it places on the network. It is
   up to the implementor to take into account overall requirements of
   the network environment and the intended use of the protocol, if
   possible, to optimize overall characteristics of the implementation.
   Several such issues will be discussed in this chapter.

   5.1 Retransmission Strategies

      The IRTP requires that a timer mechanism exists to somehow trigger
      retransmissions and requires that the packet with sequence number
      snd_una be the one retransmitted. It is not required that
      retransmission be performed on every timer event, though this is
      one "retransmission strategy". A possible alternative strategy is
      to perform a retransmission on a timer event only if no ACKs have
      been received since the last event.

      Additionally, the interval of the timer can affect the performance
      of the strategies, as can the value of MYRCV and the lossiness of
      the network environment.

      It is not within the scope of this document to recommend a
      retransmission strategy, only to point out that different
      strategies have different consequences. It might be desirable to
      allow using processes to "specify" a strategy when a port is
      claimed in order to tailor the service of the protocol to the
      needs of a particular application.

   5.2 Pinging

      It is important to make explicit that IRTP modules ping by
      definition. That is, as long as a remote internet address is

known, and is in use (that is, either synchronization or data
transfer is being attempted), the protocol requires "periodic
retransmission" of packets. Note that this is true even if the
IRTP module has determined that the remote address is currently
unreachable.

It is suggested that this situation can be made more sensible by
adding two fields to the connection table. These are:

num_retries (number of times current packet has been sent)
time_out   (current retransmission timeout)

These fields are to be used as follows. It is assumed that there
is some default initial value for time_out called DEFTIME, some
(relatively long) value for time_out called PINGTIME and some
value MAX_TRIES. The exact values of these constants are
implementation dependent. The value of DEFTIME may also be
retransmission strategy dependent.

At the time that a connection table is initialized, num_retries is
set to zero, and time_out is set to DEFTIME. Whenever a
retransmission event occurs (this will either be a retransmission
of a SYNCH packet or of the packet with sequence number snd_una),
num_retries is incremented by one unless it is equal to MAX_TRIES.
If a destination is determined to be unreachable, either via an
ICMP message or a Destination Host Dead message, num_retries is
set to MAX_TRIES. Whenever num_retries transitions to MAX_TRIES,
either by being incremented or as above, the destination is is
presumed unreachable and user processes are notified. At this
point, time_out is set to PINGTIME, the state of the connection
does not change and retransmissions occur at PINGTIME intervals
until the destination becomes reachable.

Conversely, whenever a SYNCH_ACK is received (in synch_wait
state), or an (implicit or explicit) acknowledgment of sequence
number snd_una is received (in data transfer state), time_out is
set to DEFTIME and num_retries is reset to zero. If time_out was
already set to PINGTIME, user processes are notified that the
destination is now reachable.

The effect of this system is obvious. The implementation still
pings as required, but at presumably very infrequent intervals.
Alternative solutions, which might place the decision to ping on
using processes, are considered undesirable because

    o   IRTP itself becomes more complicated in terms of states of
        the connection table

      o the user interface becomes both more complicated and more
         rigid

      o such solutions might be deadlock prone in some instances

      o it seems appropriate that the host to host protocol should
         be the place to determine destination reachability, if the
         overall application requires that such information be known
         (as it does in the environments intended for IRTP.)

5.3 Deleting Connection Tables

The protocol as defined does not allow connection tables to be
deleted (or for a connection state to transition to out_of_synch
from any other state). It might be appropriate to delete a
connection table if it is known that the destination internet
address is no longer one which this host wants to communicate
with. (The only danger there is that if the destination does not
know this, it could ping this host forever.) It is dangerous to
delete a connection table or to go into out_of_synch state to
avoid pinging when a destination does not appear to be there. Two
hosts with the same such strategy could potentially deadlock and
fail to resynchronize.

AUTHOR'S ADDRESS

Trudy Miller
Advanced Computer Communications
720 Santa Barbara Street
Santa Barbara, CA 93101
(805) 963-9431