                  Network Time Protocol (NTP) over the OSI
                          Remote Operations Service

Status of this Memo

   This memo suggests an Experimental Protocol for the OSI and Internet
   communities. Hosts in either community, and in particular those on
   both are encouraged to experiment with this mechanism. Please refer
   to the current edition of the "IAB Official Protocol Standards" for
   the standardization state and status of this protocol. Distribution
   of this memo is unlimited.

Table of Contents

1. Introduction

   This document describes the Remote Operations and Abstract Syntax for
   the operation of the Network Time Protocol (NTP) over an ISO OSI
   stack.

   NTP itself is documented in great detail in RFC 1119.

1.1 Motivation

   The motivation behind the implementation of a Remote Operations

Service implementation of NTP is fourfold.

1. The inclusion of a useful service to an OSI
    environment.

2. The feasibility of automatically checking a ROS/ASN.1
    specification, and automatically generating code to
    implement the protocol.

3. The feasibility of running NTP on connection oriented
    network services (CONS or X.25), and consequentially,
    the ability to use connection success or failure to
    optimise reachability discovery.

4. The generalisation of the last point: the use of ROS
    makes NTP independent of the underlying communications
    architecture.

The need for time synchronisation is clear, and RFC 1119 indicates a
few of the necessary uses of this service. However, it is becoming
clear that OSI applications are very much in need of this service
too. Not just in the local context but across the wide area. For
example much of the strong authentication outlined in X.511 is based
on encrypted packets with time stamps to indicate how long the packet
is valid for. If two hosts have clocks that are not closely
synchronised, the host with the faster clock will be more prone to
cryptographic attacks from the slower, and the slower host will
possibly find it is unauthentable.

A similar problem occurs with the X.500 directory and the service
control limiting the time allowed for the search.

Authentication between NTP peers and between clients and servers is
not addressed here, as the choice of mechanism is still the subject
of some debate.

2. Protocol Overview

The NTP application functions exactly as in RFC 1119. The use of
remote operations and the underlying Application support means that
for NTP daemons to peer with one another, they send an A-
ASSOCIATE.REQUEST, and receive an A-ASSOCIATE.INDICATION.

On successful association, they subsequently periodically invoke the
appropriate Remote Operation with the appropriate parameters at the
appropriate frequency.

On failure, they mark the peer as unreachable.

The states that an ntp daemon records for each peer are enhanced from
RFC 1119 to include:

    Connected: this indicates the host is connected with its peer and
    synchronisation data is being exchanged.

    Connecting: this state indicates that a connection is in progress.
    Hosts at large distances may take several seconds to connect, and
    such blocking can perturb the exchange of data with other hosts.
    Therefore, the connection is made asynchronously.

    Accepting: this state indicates that a connection is being
    accepted from another host, but the necessary negotiation of
    transport session etc has not been fulfilled yet. This is another
    asynchronous part.

    Disconnected: this state is reached if the remote host cannot be
    contacted.

3. Operation of the Protocol

   The use of a connection oriented service means that the operation of
   the NTP algorithm is slightly different. This stems firstly from
   some necessary adjustments made to the protocol and secondly from
   some optimisations that are possible through the use of connections.

   Firstly, the reachability of the host can be directly determined.
   The NTP protocol maintains a shift register to determine if it is
   likely that a peer is still responding and exchanging data. This
   works by recording over the last eight transfers how many responses
   have been received. If there have been no responses to the last
   eight packets, then the host is deemed unreachable.

   Naturally, with a connection to the remote host, the reachability is
   immediately determinable. Either a connection is established or the
   connection is broken or not yet made. For this reason it is not
   necessary to rely on the shift register to determine reachability.

   Secondly, there are a large number of optimisations that can be made
   by use of the connection oriented mode. The NTP packet format can be
   broken into several categories.

    a) Synchronisation data

    b) Authentication data

    c) Protocol data

Of these classes of data, only the first (a) is necessary to maintain
the synchronisation between hosts. Information such as protocol
version and the precision of the local clock are not likely to vary
over the lifetime of the connection. Likewise the authentication if
in use need only be done at connection establishment and is not
necessarily required for every packet.

For these reason, the NTP protocol can be simplified slightly to
remove this information. This can be seen in the specification for
the Packet in Appendix A.

4. Network Considerations

Although on first inspection it might be thought that a high speed
network is necessary for accurate synchronisation, this is not the
case. What is more important is the dispersion of the packet
traversal times. It is normally the case that a low speed network
with little variance in packet transit times will give better results
than a high speed network with large differences in individual packet
transit times. This would lead us to think that connection oriented
networks with resource allocation done at connection time might lead
to higher accuracies than connectionless networks which can suffer
large swings in packet transit time under high loading. (This is
heresy!)

5. Implementation Model

Ideally, the implementor will provide interoperability between the
existing UDP based NTP service, and a ROS based service.

To this end, the internal records that hold NTP state information,
can be kept the same as existing implementations, and for
optimisation reasons, the internal representations of NTP packets can
be the same. Translation between these and appropriate ROS/ASN
concrete encodings can be provided by automatic translators such as
Rosy [ISODE].

6. Constructing NTP Data Fields

The way in which the data fields in the Packet described in Appendix
A is unchanged from RFC 1119. This simplifies implementations based
on existing ones, and encourages interworking.

7. Discussion

From the limited testing of this model so far done, the results would
seem to indicate that the ROS based model running over an X.25
service is of similar reliability as the UDP model. Until further

experimentation can be performed, specific data can not be given.

However, in the UK where the most common method of time
synchronisation is the system administrators watch and typing in the
time to the nearest minute, this method is clearly far superior.

Connection management is transparent to NTP since it is implemented
beneath the Remote Operations Service. However, an NTP
implementation must have access to the status of connections, and
uses this not only for reachability information but also to find the
information gleaned at connect time and no longer exchanged in NTP
operations.

8. Prototype Experience

   There are a number of UK sites running NTP over ROS over X.25 with an
   earlier ROS specification, with at least one site peering both over
   ROS with UK sites on X.25, and over UDP with US Internet sites.

   Initial experience is promising. The table below shows the
   reachabilities, delays, offsets and dispersions for the central UK
   site peering with 2 JANET sites (IP addresses not meaningful, but
   shown as 126.0.0.1), and three US sites.

Address	Strat	Poll	Reach	Delay	Offset	Disp
+126.0.0.1	3	64	377	718.0	0.0	3.0
+umd1.umd.edu	1	1024	177	535.0	13.0	13.0
*128.4.0.5	1	64	167	545.0	10.0	524.0

9. References

   1. Mills, D., "Network Time Protocol (Version 2) Specification and
       Implementation", RFC-1119, UDEL, September 1989.

   2. Mills, D., "Algorithms for Synchronizing Network Clocks", RFC-
       956, M/A-COM Linkabit, September 1985.

   3. Postel, J. "User Datagram Protocol", RFC-768, USC Information
       Sciences Institute, August 1980.

   4. ISO TC97, "Specification of Abstract Syntax Notation One
       (ASN.1)", Draft International Standard ISO/DIS 8824, 6 June 1985.

   5. CCITT, "Remote Operations: Model, Notation and Service
       Definition", CCITT X.ros0 or ISO/DP 9072/1, Geneva, October 1986.

   6. Mills, D., "Internet Time Synchronization: The Network Time

           Protocol (NTP)", RFC 1129, UDEL, October 1989.

     7. Mills, D., "Measured Performance of the Network Time Protocol in
         the Internet System", RFC 1128, October 1989.

     8. Rose M., et al, "The ISO Development Environment: User's Manual".

10. Acknowledgements

         The Authors would like to thank Dave Mills for his valuable
         comments on an earlier version of this document.

Appendix A. ROS "Header" Format

         -- NTP definitions for ROS specification
         --
         -- Julian Onions, Nottingham University, UK.
         --
         -- Mon Jun 5 10:07:07 1989
         --

         NTP DEFINITIONS ::=

         BEGIN

         update OPERATION
          ARGUMENT Packet
          ::= 0

         query OPERATION
          ARGUMENT NULL
          RESULT ClockInfoList
          ::= 1

         -- Data Structures

         BindArgument ::=
          fullbind SEQUENCE {
                 psap[0] IA5String OPTIONAL,
                 version[1] BITSTRING {
                         version-0(0),
                         version-1(1),
                         version-2(2)
                 } DEFAULT version-2,
                 authentication[2] Authentication OPTIONAL,
                 mode[3] BindMode
          }

```
     Authentication ::= ANY

     BindMode ::= ENUMERATED {
             normal(0),      -- standard NTP
             query(1)        -- queries only
      }

     BindResult ::=
      SEQUENCE {
             version[1] INTEGER DEFAULT 2,
             authentication[2] Authentication OPTIONAL,
             mode[3] BindMode
      }

     BindError ::=
      SEQUENCE {
             reason[0] INTEGER {
                     refused(0),
                     validation(1),
                     version(2),     -- version not supported
                     badarg(3),      -- bad bind argument
                     congested(4)    -- catch all!
             },
             supplementary[1] IA5String OPTIONAL
      }


                                     -- basic exchange packet

     Packet ::= SEQUENCE {
      leap                 Leap,
      mode                 Mode,
      stratum[1]           INTEGER,
      pollInterval[2]      INTEGER,
      precision[3]         INTEGER,
      synchDistance        SmallFixed,
      synchDispersion      SmallFixed,
      referenceClockIdentifier ClockIdentifier,
      referenceTimestamp   TimeStamp,
      originateTimestamp   TimeStamp,
      receiveTimestamp     TimeStamp,
      transmitTimestamp    TimeStamp
     }

     ClockInfoList ::= SET OF ClockInfo

     ClockInfo ::= SEQUENCE {
      remoteAddress        Address,
```

```
        localAddress            Address,
        flags[0]                BIT STRING {
                    configured(0),
                    authentable(1),
                    sane(2),
                    candidate(3),
                    sync(4),
                    broadcast(5),
                    referenceClock(6),
                    selected(7),
                    inactive(8)
        },
        packetsSent[1]          INTEGER,
        packetsReceived[2]      INTEGER,
        packetsDropped[3]       INTEGER,
        timer[4]                INTEGER,
        leap                    Leap,
        stratum[5]              INTEGER,
        ppoll[6]                INTEGER,
        hpoll[7]                INTEGER,
        precision[8]            INTEGER,
        reachability[9]         INTEGER,
        estdisp[10]             INTEGER,
        estdelay[11]            INTEGER,
        estoffset[12]           INTEGER,
        reference[13]           ClockIdentifier OPTIONAL,
        reftime                 TimeStamp,
        filters                 SEQUENCE OF Filter
        }

    Leap ::= [APPLICATION 0] ENUMERATED {
            nowarning(0),
            plussecond(1),
            minussecond(2),
            alarm(3)
     }

    SmallFixed ::= [APPLICATION 1] IMPLICIT SEQUENCE {
            integer INTEGER,
            fraction INTEGER
     }

    ClockIdentifier ::= CHOICE {
                    referenceClock[0] PrintableString,
                    inetaddr[1] OCTET STRING,
                    psapaddr[2] OCTET STRING
     }
```

```
     TimeStamp ::= [APPLICATION 2] IMPLICIT SEQUENCE {
             integer INTEGER,
             fraction INTEGER
      }

     KeyId ::= [APPLICATION 4] INTEGER

     Mode ::= [APPLICATION 4] ENUMERATED {
             unspecified (0),
             symmetricActive (1),
             symmetricPassive (2),
             client (3),
             server (4),
             broadcast (5),
             reservered (6),
             private (7)
      }

     Filter ::= SEQUENCE {
             offset INTEGER,
             delay INTEGER
      }

     Address ::= OCTET STRING -- for now
     END
```

11. Security Considerations

   Security issues are not discussed in this memo.

12. Authors' Addresses

   Jon Crowcroft
   Computer Science Department
   University College London
   Gower Street
   London WC1E 6BT UK

   EMail: JON@CS.UCL.AC.UK

   Julian P. Onions
   Computer Science Department
   Nottingham University
   University Park
   Nottingham, NG7 2RD UK

   EMail: JPO@CS.NOTT.AC.UK