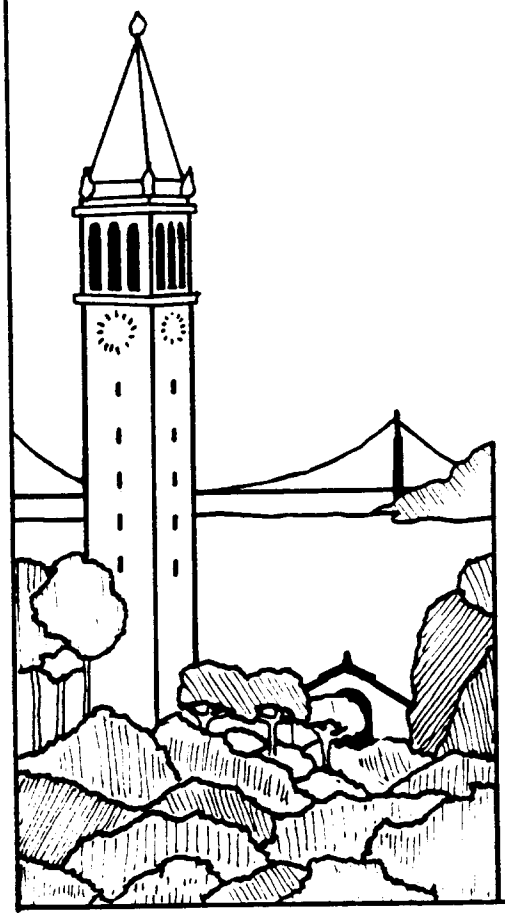
The Berkeley UNIX 4.3BSD Time Synchronization Protocol

Protocol Specification

Riccardo Gusella and Stefano Zatti



Report No. UCB/CSD 85/250

June 1985
PROGRES Report No. 85.13

Computer Science Division (EECS)
University of California
Berkeley, California 94720

The Berkeley UNIX 4.3BSD Time Synchronization Protocol

Protocol Specification

Riccardo Gusella and Stefano Zatti

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

Introduction

The Time Synchronization Protocol (TSP) has been designed for specific use by the program *timed*, a local area network clock synchronizer for the UNIX 4.3BSD operating system. *Timed* is built on the DARPA UDP protocol and is based on a master slave scheme.

TSP serves a dual purpose. First, it supports messages for the synchronization of the clocks of the various hosts in a local area network. Second, it supports messages for the election that occurs among slave time daemons when, for any reason, the master disappears. The synchronization mechanism and the election procedure employed by the program *timed* are described in two other documents [1,2].

While some messages need not be sent in a reliable way, most communication in TSP does require reliability. Reliability is achieved by the use of acknowledgements, sequence numbers, and retransmission when message losses occur. When a message that requires acknowledgment is not acknowledged, the time daemon which has sent the message will assume that the addressee is down. This document will not describe the details of how reliability is implemented, but will only point out when a message type requires a reliable transport mechanism.

The message format in TSP is the same for all message types; however, in some instances, one or more fields are not used. The next section describes the message format. The following sections describe in detail the different message types, their use and the contents of each field.

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089, and by the Italian CSELT Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency, of the US Government, or of CSELT.

Message Format

The structure of a TSP message is the following:

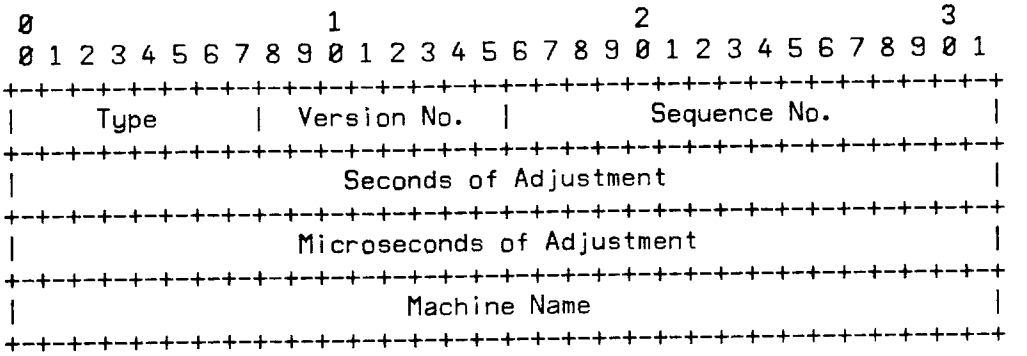
1) An 8-bit message type.
2) An 8-bit version number, specifying the protocol version a message refers to.
3) A 16-bit sequence number to be used for recognizing duplicate messages that occur when acknowledgments are lost.
4) Two 32-bit quantities that contain timing information expressed in seconds and microseconds.
5) A null-terminated string of 32 characters with the name of the machine sending the message.

The following figure shows the definition of the TSP message format and the various message types.

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved. The Berkeley software License Agreement
 * specifies the terms and conditions for redistribution.
 */

/*  @(#)timed.h   1.1  (Berkeley)    6/14/85 */

/*  Time Synchronization Protocol    */

struct tsp {
    u_char   tsp_type;
    u_char   tsp_vers;
    short    tsp_seq;
    struct timeval tsp_time;
    char tsp_name[32];
};

#define  TSPVERSION  1
/*
 * Command types.
 */
#define  TSP_ADJTIME      1    /* send adjtime */
#define  TSP_ACK          2    /* generic acknowledgement */
#define  TSP_MASTERREQ    3    /* ask for master's name */
#define  TSP_MASTERACK    4    /* acknowledge master request */
#define  TSP_SETTIME      5    /* send network time */
#define  TSP_MASTERUP     6    /* inform slaves that master is up */
#define  TSP_SLAVEUP      7    /* slave is up but not polled */
#define  TSP_ELECTION     8    /* advance candidature for master */
#define  TSP_ACCEPT       9    /* support candidature for master */
#define  TSP_REFUSE       10   /* reject candidature for master */
#define  TSP_CONFLICT     11   /* two or more masters present */
#define  TSP_RESOLVE      12   /* masters' conflict resolution */
#define  TSP_QUIT         13   /* reject candidature when master is up */
#define  TSP_DATE         14   /* reset time (date command) */
#define  TSP_DATEREQ      15   /* remote request to reset time */
#define  TSP_DATEACK      16   /* acknowledge time setting  */
#define  TSP_TRACEON      17   /* turn tracing on */
#define  TSP_TRACEOFF     18   /* turn tracing off */
#define  TSP_MSITE        19   /* find out master's site */
#define  TSP_MSITEREQ     20   /* remote master's site request */
#define  TSP_TEST         21   /* test of election algorithm */
```

The following charts describe the message types, show their fields, and explain their usages. For the purpose of the following discussion, a time daemon can be considered to be in one of three states: slave, master, or candidate. Also, the word *broadcast* refers to the sending of a message to all active time daemons.

Adjtime Message

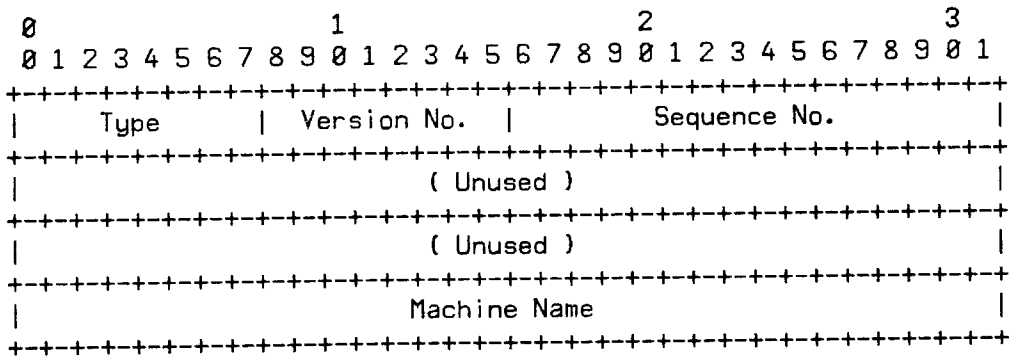


Type: TSP_ADJTIME (1)

Description:

The master sends this message to a slave to communicate the difference between the clock of the slave and the network time the master has just computed. The slave will accordingly adjust the time of its machine. This message requires acknowledgment.

Acknowledgment Message

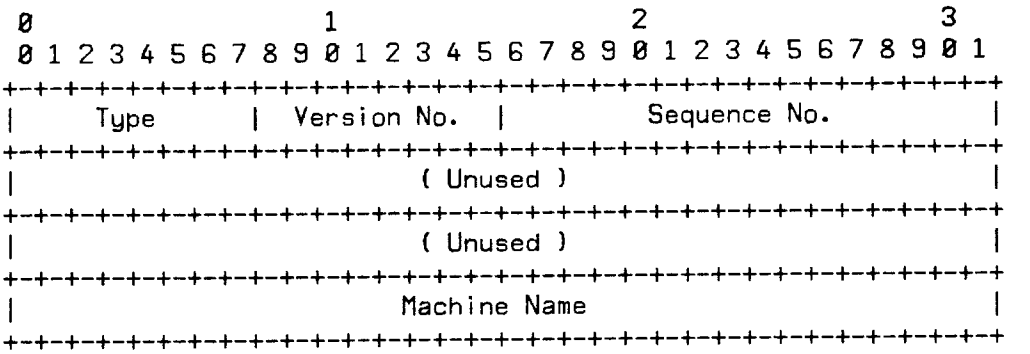


Type: TSP_ACK (2)

Description:

Both the master and the slaves use this message for acknowledgment only. It is used in several different contexts, for example in replay to an Adjtime message.

Master Request Message

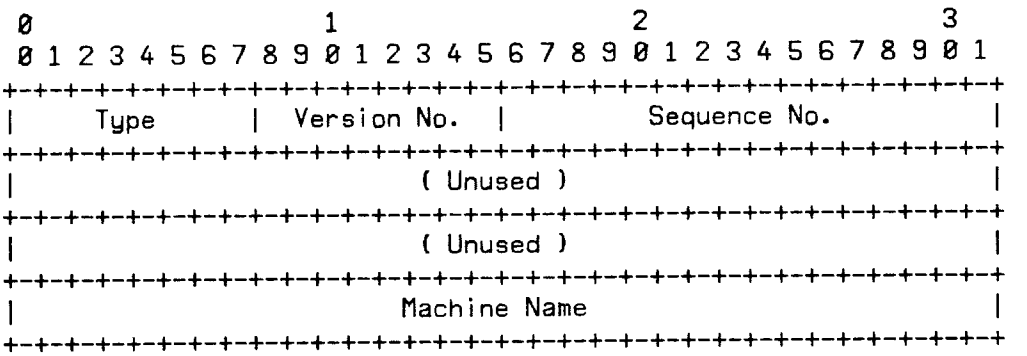


Type: TSP_MASTERREQ (3)

Description:

A newly-started time daemon broadcasts this message to inform the master of its name so that it can be added to the list of machines participating in the synchronization. It requires acknowledgment.

Master Acknowledgement

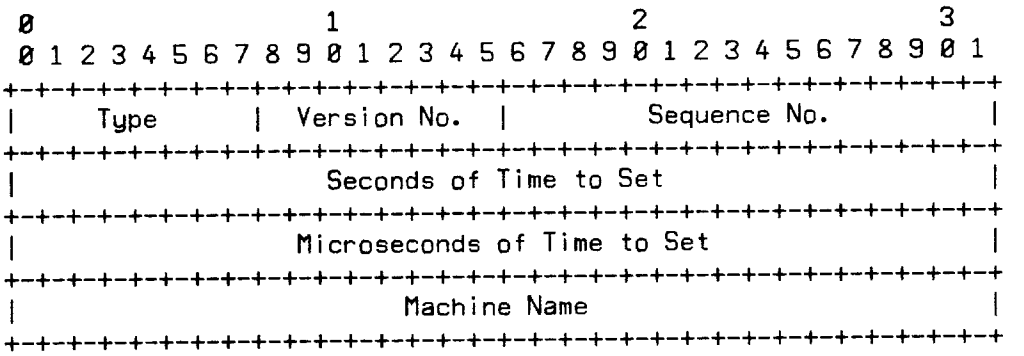


Type: TSP_MASTERACK (4)

Description:

The master sends this message to acknowledge the Master Request message.

Set Network Time Message

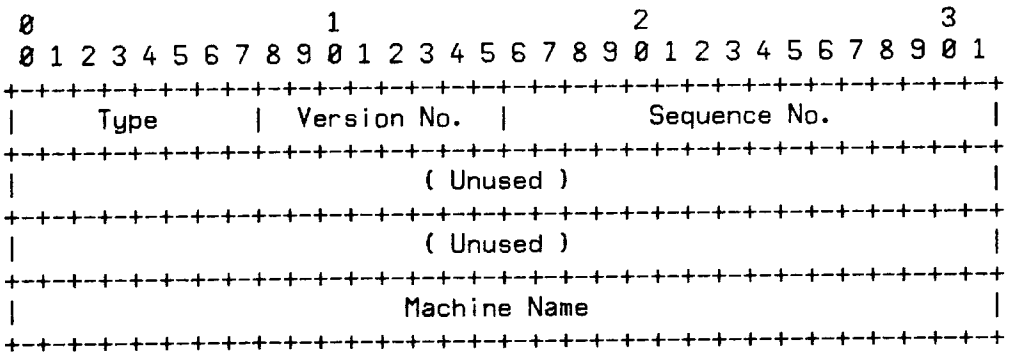


Type: TSP_SETTIME (5)

Description:

The master sends this message to a newly-started time slave to set its time. It contains the master's time as an approximation of the network time. It requires acknowledgment. After sending this message, the master starts a synchronization round to eliminate the small time difference caused by the random delay in the communication channel.

Master Active Message

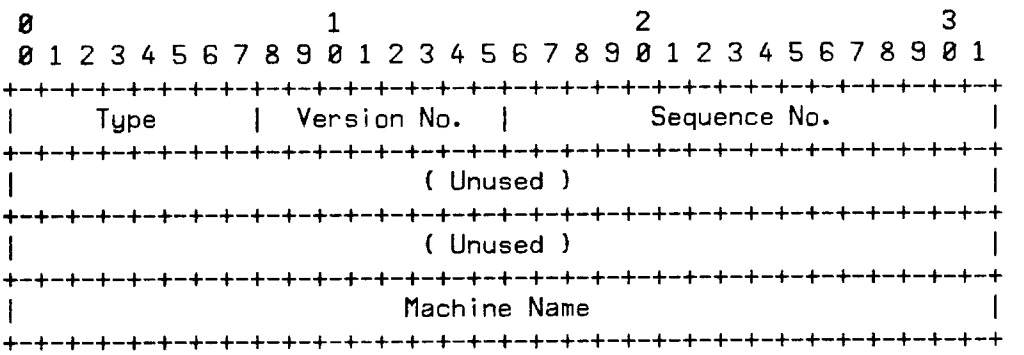


Type: TSP_MASTERUP (6)

Description:

The master broadcasts this message to solicit the names of the active slaves. Slaves will reply with a Slave Active message.

Slave Active Message

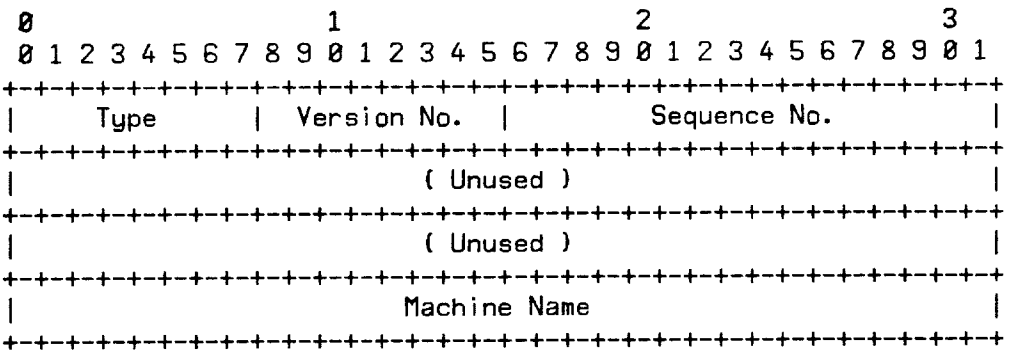


Type: TSP_SLAVEUP (7)

Description:

A slave sends this message to the master in answer to a Master Active message. In addition, time daemons which are not allowed by a system administrator to become masters broadcast this message to inform the master that they are not controlled by it.

Master Candidature Message

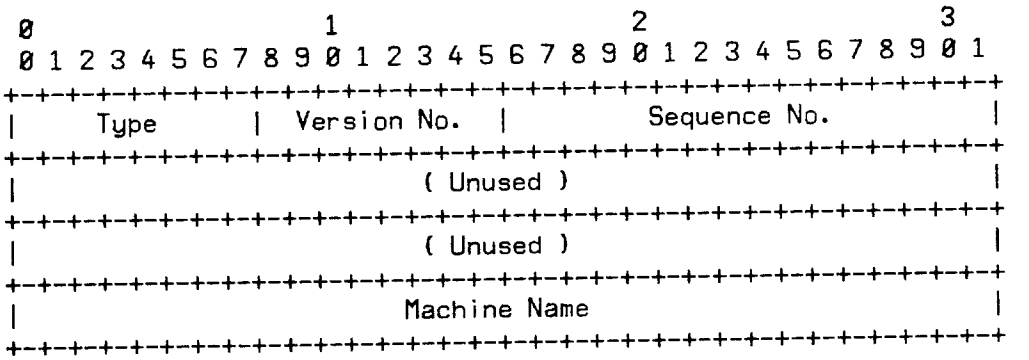


Type: TSP_ELECTION (8)

Description:

A slave eligible to become a master broadcasts this message when its election timer expires. The message declares that the slave wishes to become the new master.

Candidature Acceptance Message

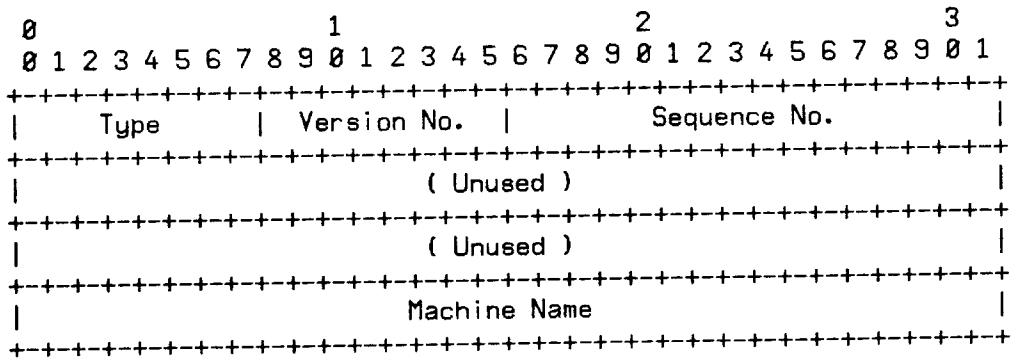


Type: TSP_ACCEPT (9)

Description:

A slave sends this message to accept the candidature of the time daemon that has broadcast an Election message. The candidate will add the slave's name to the list of machines that it will control should it become the master.

Candidature Rejection Message

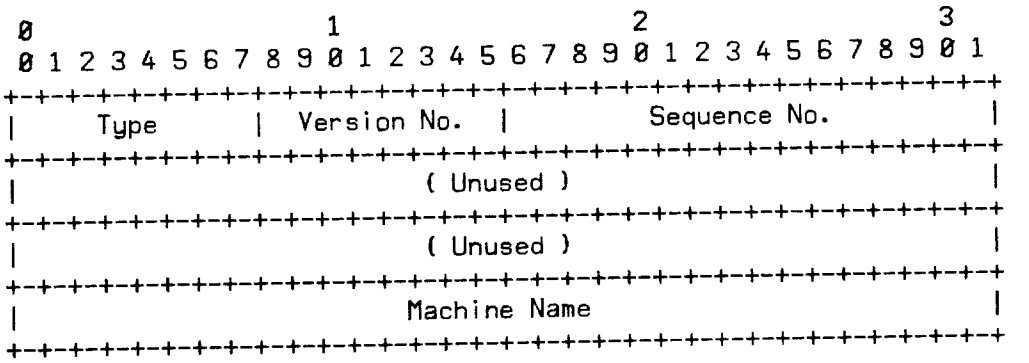


Type: TSP_REFUSE (10)

Description:

After a slave accepts the candidature of a time daemon, it will replay to any election messages from other slaves with this message. This rejects any candidature other than the first received.

Multiple Master Notification Message

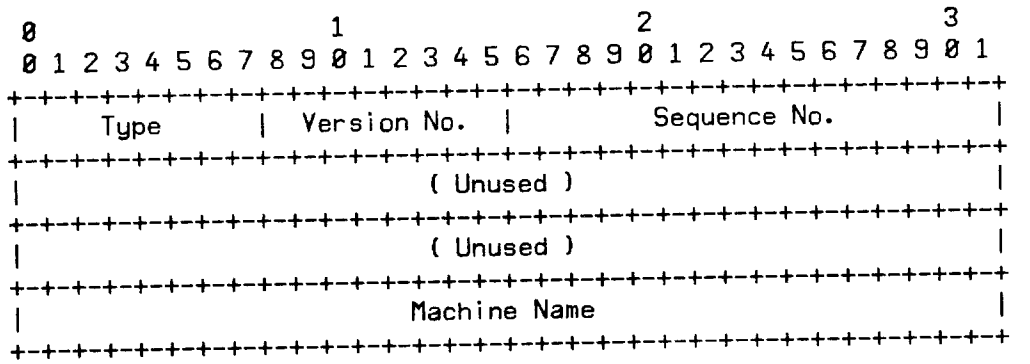


Type: TSP_CONFLICT (11)

Description:

When two or more masters reply to a Master Request message, the slave uses this message to inform one of them that more than one master exists.

Conflict Resolution Message

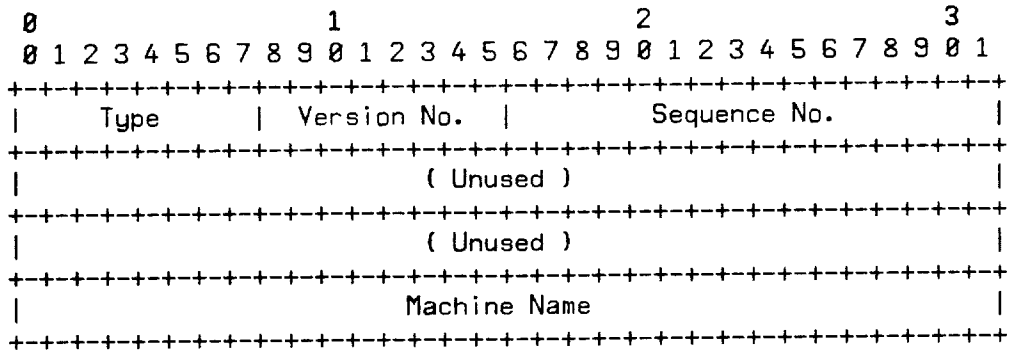


Type: TSP_RESOLVE (12)

Description:

A master which has been informed of the existence of other masters sends this message to obtain their names.

Quit Message

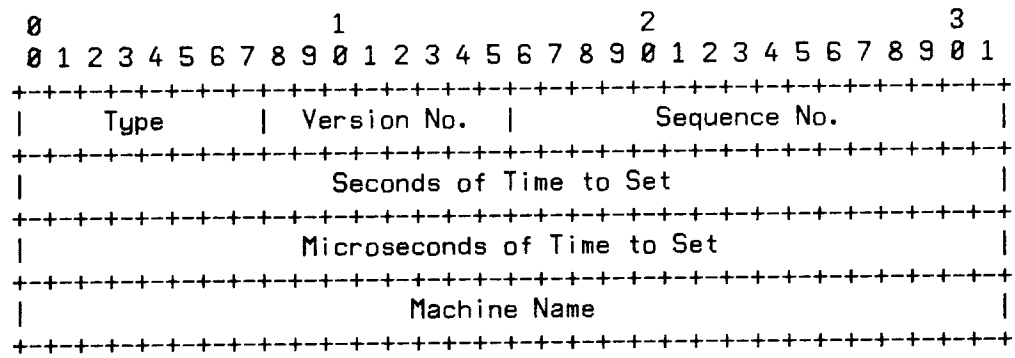


Type: TSP_QUIT (13)

Description:

This message is sent by the master in two different contexts: 1) to a candidate that broadcasts an Election message, and 2) to another master when notified of its existence. In both cases, the two time daemons will become slaves.

Set Date Message

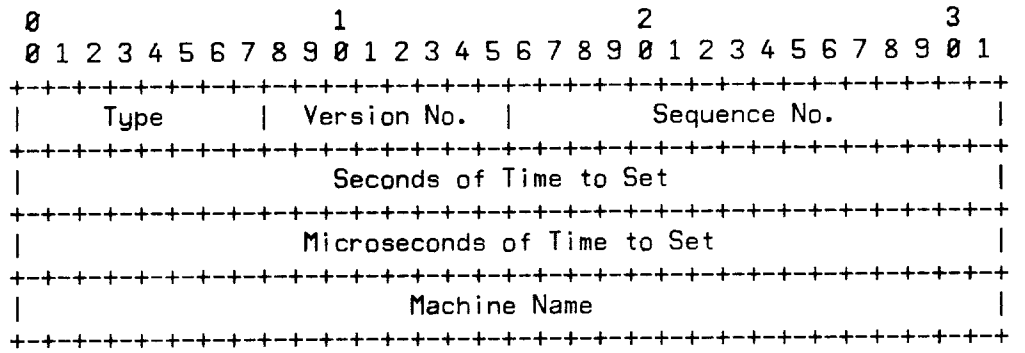


Type: TSP_DATE (14)

Description:

The program *date(1)* sends this message to the local time daemon when a super-user wants to set the network date. If the local time daemon is the master, it will set the time; if it is a slave, it will communicate the desired date to the master.

Set Date Request Message

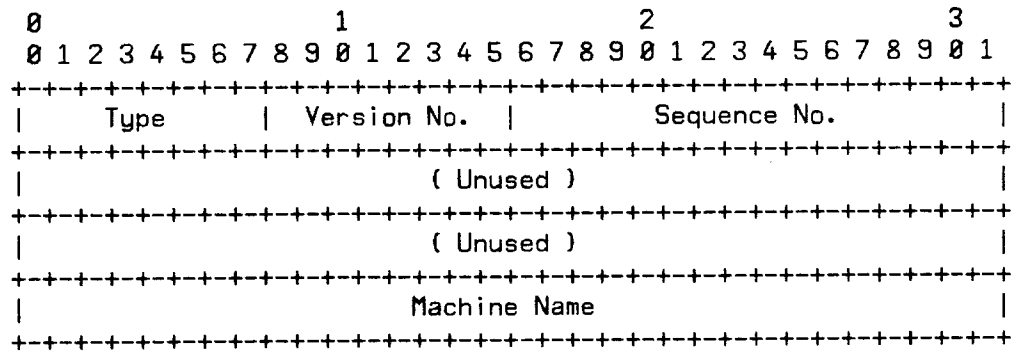


Type: TSP_DATEREQ (15)

Description:

A slave that has received a Set Date message will communicate the desired date to the master using this message.

Set Date Acknowledgment Message

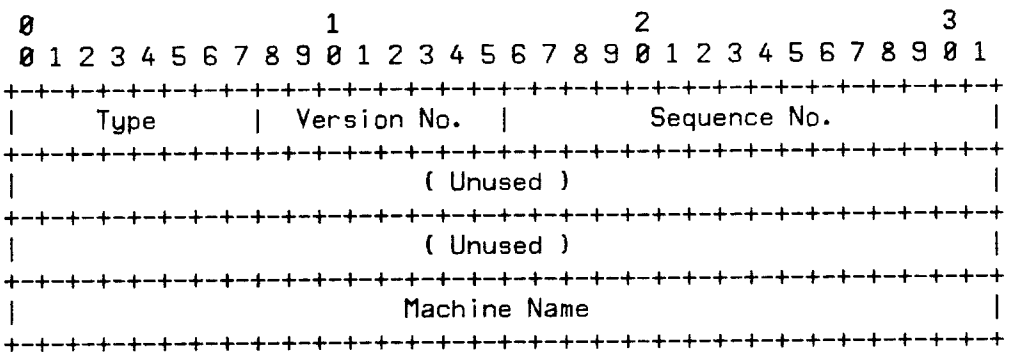


Type: TSP_DATEACK (16)

Description:

The master sends this message to a slave in acknowledgment of a Set Date Request Message. The same message is sent by the local time daemon to the program *date(1)* to confirm that the network date has been set by the master.

Start Tracing Message

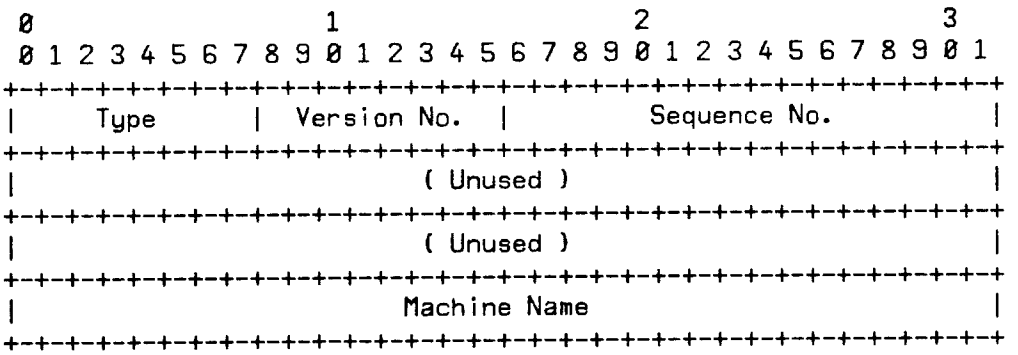


Type: TSP_TRACEON (17)

Description:

The controlling program *timedc* sends this message to the local time daemon to start the recording in a system file of all messages received.

Stop Tracing Message

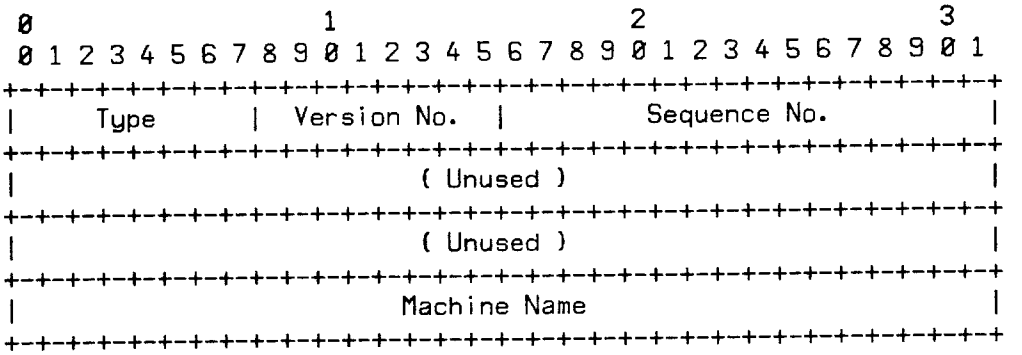


Type: TSP_TRACEOFF (18)

Description:

Timedc sends this message to the local time daemon to stop the recording of messages received.

Master Site Message

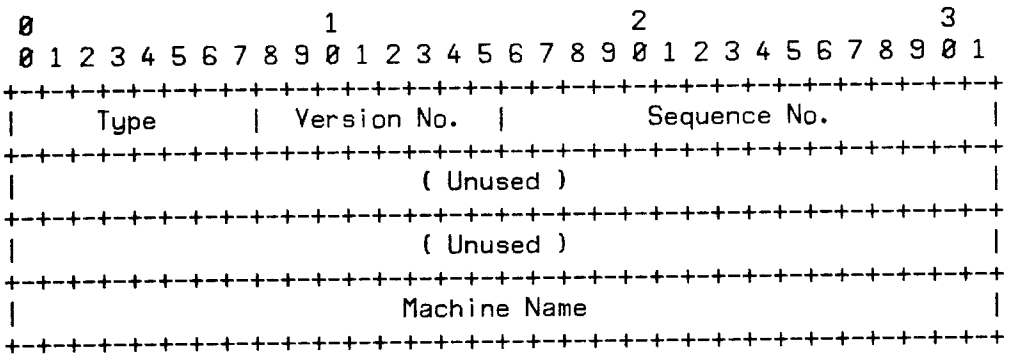


Type: TSP_MSITE (19)

Description:

Timedc sends this message to the local time daemon to find out where the master is running.

Remote Master Site Message

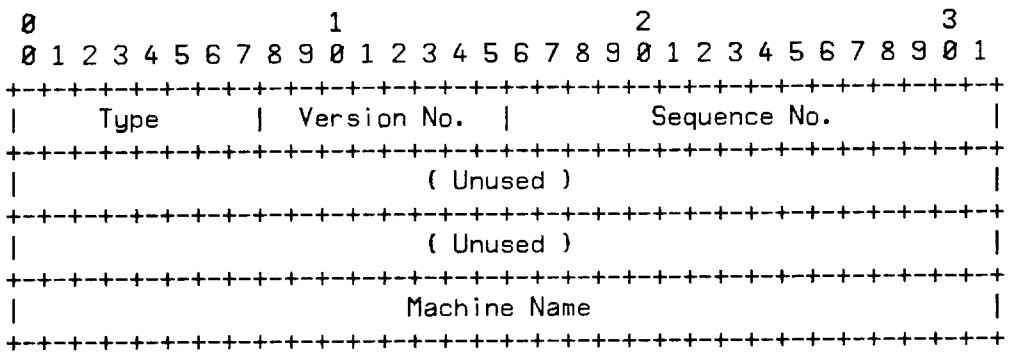


Type: TSP_MSITEREM (20)

Description:

A local time daemon broadcasts this message to find the location of the master. It then uses the Acknowledgement message to communicate this location to *timedc*.

Test Message



Type: TSP_TEST (21)

Description:

For testing purposes, *timedc* sends this message to a set of slaves to cause their election timers expire simultaneously.

References

1. R. Gusella and S. Zatti, *Clock Synchronization in a Local Area Network*, University of California, Berkeley, Technical Report, *to appear*.
2. R. Gusella and S. Zatti, *An Election Algorithm for a Distributed Clock Synchronization Program*, University of California, Berkeley, Technical Report, *to appear*.