CLOCK SYNCHRONIZATION

Uday Acharya Audumbar Chormale

Overview

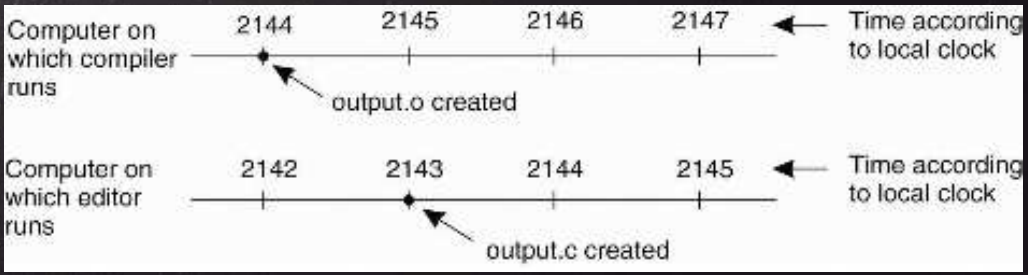
- Clock synchronization
    1. The Need of synchronization
    2. Physical Clocks
    3. Cristian's Algorithm
    4. Berkeley Algorithm
    5. Lamport logical clock

- Network Time Protocol
    1. Overview
    2. Architecture
    3. NTP Flow Diagrams
    4. NTP Security Model
    5. NTP Perfomance compared

Need Of Synchronization

Since each machine has it's own clock, it's difficult to time order the events occurred.

Example: make utility



Need Of Synchronization

- Distributed database transaction journalling and logging
- Stock market buy and sell orders
- Aviation traffic control and position reporting
- Multimedia synchronization for real-time teleconferencing
- Interactive simulation event synchronization and ordering
- Distributed network gaming and training

Clock Synchronization

- Physical clocks
- Logical clocks
- Vector clocks

Physical Clocks

Problem: Sometimes we simply need the exact time, not just an ordering.

Solution: Universal Coordinated Time (UTC):

- Based on the number of transitions per second of the cesium 133 atom (pretty accurate).
- At present, the real time is taken as the average of some 50 cesium-clocks around the world.
- Introduces a leap second from time to time to compensate that days are getting longer.
- UTC is broadcast through short wave radio and satellite.
- Satellites can give an accuracy of about ±0.5 ms.

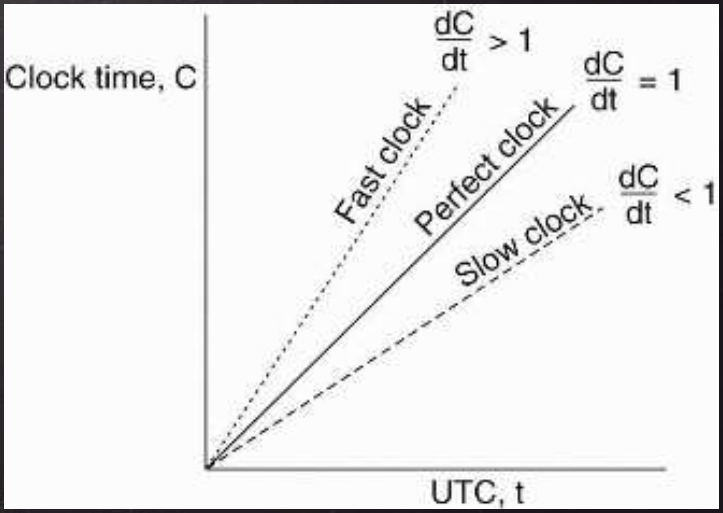
Physical Clocks

Problem: Suppose we have a distributed system with a UTC-receiver somewhere in it ⇒ we still have to distribute its time to each machine.

Basic principle:

- Every machine has a timer that generates an interrupt H times per second.
- There is a clock in machine p that ticks on each timer interrupt. Denote the value of that clock by $C_p(t)$, where t is UTC time.
- Ideally, we have that for each machine p, $C_p(t) = t$, or, in other words, $dC/dt = 1$.

Physical Clocks

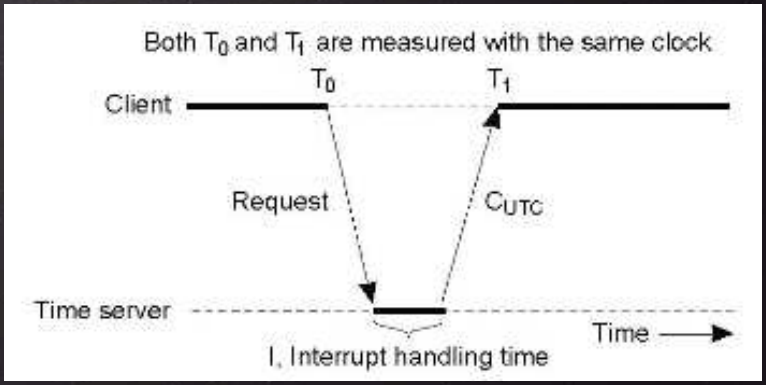


Never let two clocks in any system differ by more than d time units ⇒ synchronize at least every d / (2r) seconds.

Cristian's Algorithm

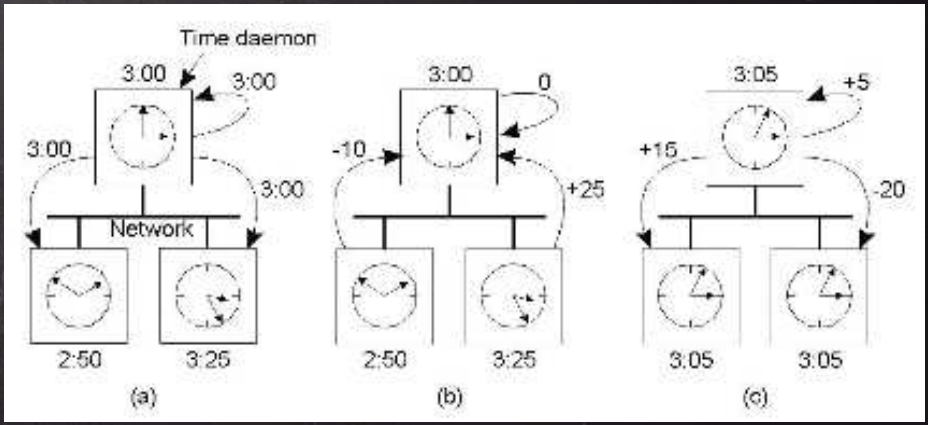
- Assume one machine (the time server) has a WWV receiver and all other machines are to stay synchronized with it.
- Every d/2r seconds, each machine sends a message to the time server asking for the current time.
- Time server responds with message containing current time, CUTC.

Cristian's Algorithm



- A major problem – the client clock is fast -> arriving value of CUTC will be smaller than client's current time, C.
- One needs to gradually slow down client clock by adding less time per tick.

The Berkeley Algorithm



- The time daemon asks all the other machines for their clock values.
- The machines answer and the time daemon computes the average.
- The time daemon tells everyone how to adjust their clock.

Lamport's Logical Clocks

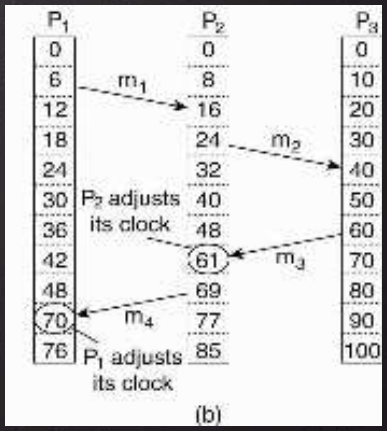
- All processes agree on the order in which events occur.

- Updating counter Ci for process Pi,
  Before executing an event Pi executes
  $Ci \leftarrow Ci + 1$.
- When process Pi sends a message m to Pj, it sets m's timestamp ts (m) equal to Ci after having executed the previous step.

  Upon the receipt of a message m, process Pj adjusts its own local counter as

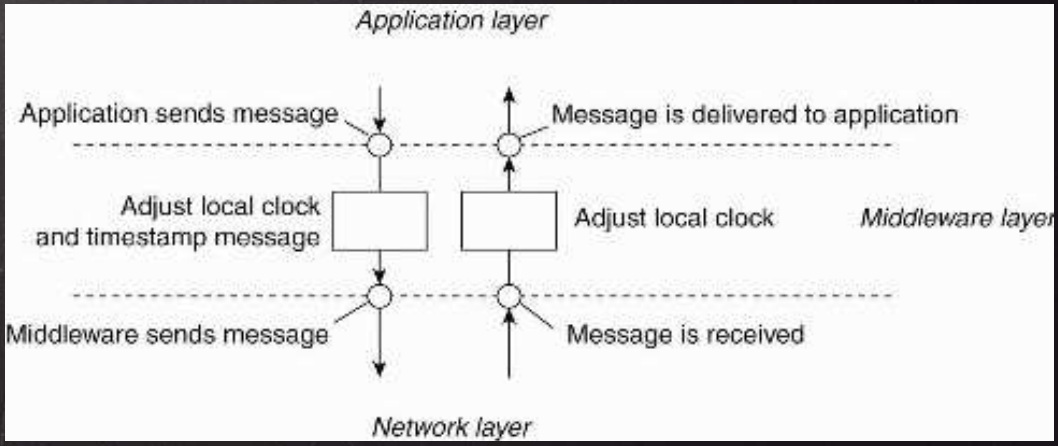
  $Cj \leftarrow \max\{Cj , ts (m)\}$, after which it then executes the first step and delivers the message to the application.

Lamport's Logical Clocks



- The "happens-before" relation → can be observed directly in two situations:
- If a and b are events in the same process, and a occurs before b then a → b is true.
- If a is the event of a message being sent by one process, and b the event of the message being received by another process, th a → b

Lamport's Logical Clocks



The positioning of Lamport's logical clocks in distributed systems.

Network Time Protocol

Overview:

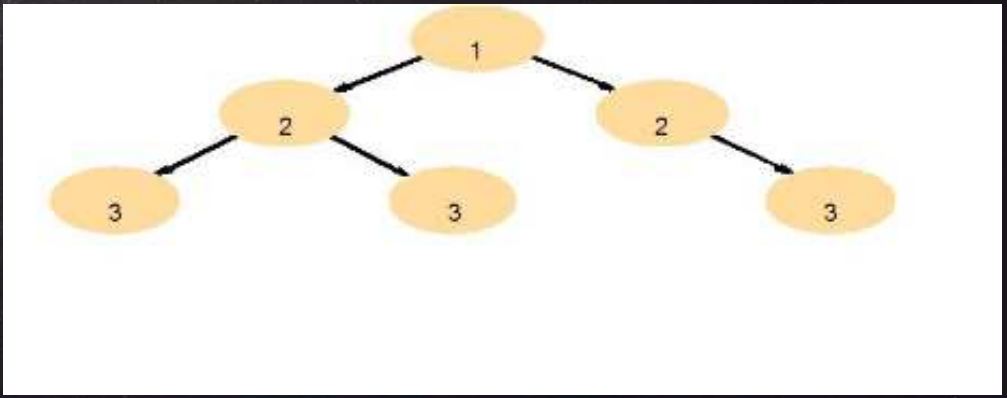
- The Network Time Protocol (NTP) is a time synchronization system for computer clocks through the Internet network
- It was mainly developed at the Delaware University in the United States.
- It is designed particularly to resist the effects of variable latency (jitter).
- NTP uses UDP port 123

Main characteristics

- Fully automatic, keeps continuously the synchronization.
- Suitable to synchronize one computer as well as a whole computer network
- Available on almost every type of computer
- Fault tolerant and dynamically autoconfiguring
- Carrying UTC time, independent of time zones and day-light saving time
- Synchronization accuracy can reach 1 millisecond.

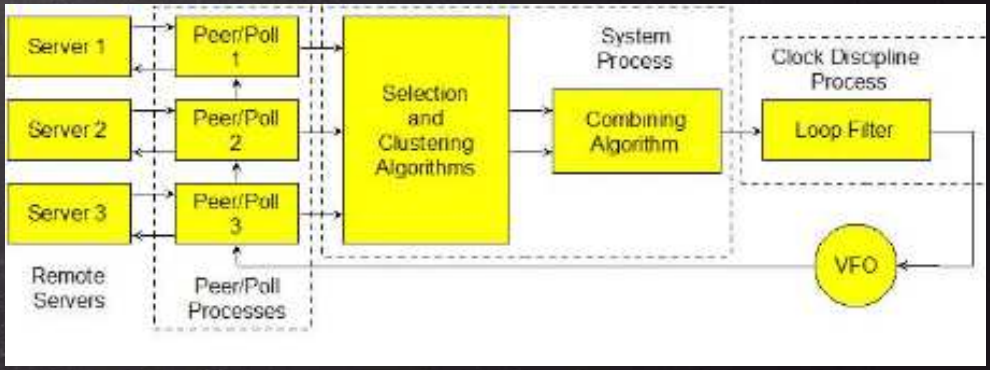
NTP mechanism:

- Stratum 1: Primary server (PS) sync directly with UTC sources.
- Stratum 2: Secondary servers (SS) sync directly to PS.
- Stratum 3: Lowest servers (LS) execute in user sites sync with SS.
- Accuracy: the number of levers (strata).



- Going away from stratum 1 toward lower stratums, the synchronization accuracy lowers.
- In such a framework, each computer can be at the same time a server for the computers belonging to the lower stratum and a client for the computers belonging to the upper stratum.
- Each server can have some hundreds of clients, so the number of computers that can be indirectly synchronized by one primary server is virtually unlimited.
- To make the system more reliable, each client can have more the one server in the upper stratum.
- In this case, the NTP software monitors continuously the figures of stability and accuracy of all configured servers, switching dynamically to the server with the best figures.

NTP Architecture Overview

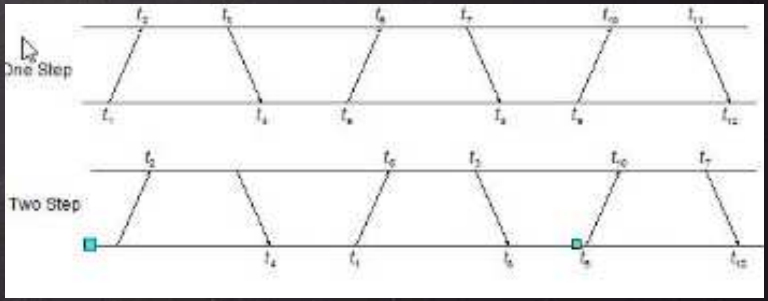


- Multiple servers/peers provide redundancy and diversity.
- Clock filters select best from a window of eight time offset samples.
- Intersection and clustering algorithms pick best truechimers and discard falsetickers.
- Combining algorithm computes weighted average of time offsets.
- Loop filter and variable frequency oscillator (VFO) implement hybrid phase/frequency-lock (P/F) feedback loop to minimize jitter and wander.

NTP peer protocol

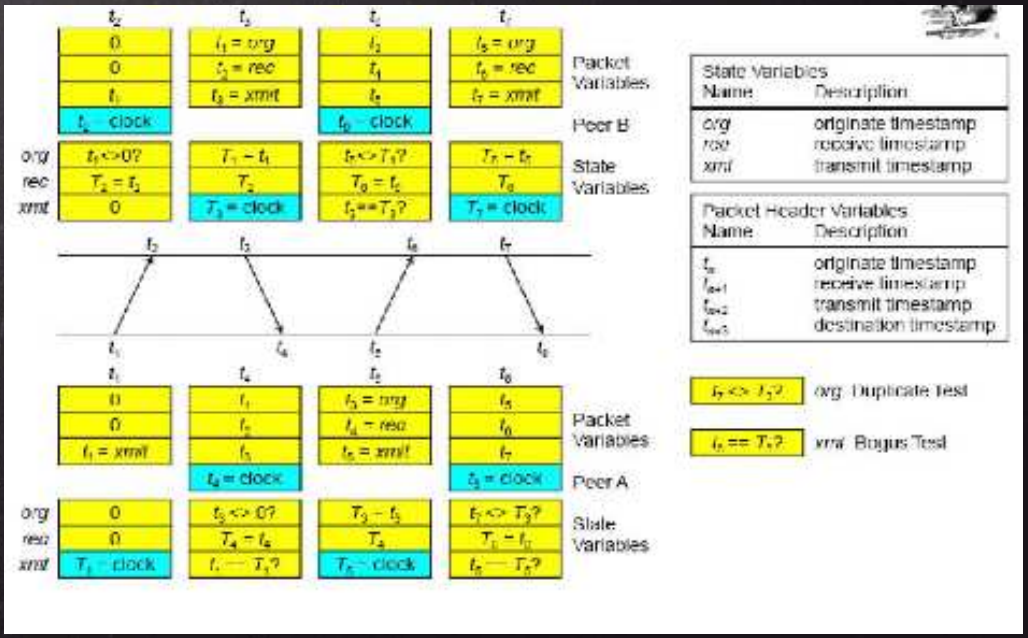
- Packet header includes T1, T2 and T3 timestamps.
- Peer state variables org, rec and xmt record the transmit and receive times of the most recent packet received.
- When a packet is transmitted
- Copy org to T1 and rec to T2.
- Copy the current time to xmt and to T3.
- When a packet received
- If T3 is the same as xmt, this is a duplicate packet.
- If T1 is not the same as org, this is a bogus packet.
- Otherwise, copy T3 to pkt and copy the current time to T4 and rec.
- Note that the protocol is symmetric and allows time values to flow both ways simultaneously and is resistant to replays and drops.
- Note the special conditions when either or both peers first start up.

Timestamp interleaving

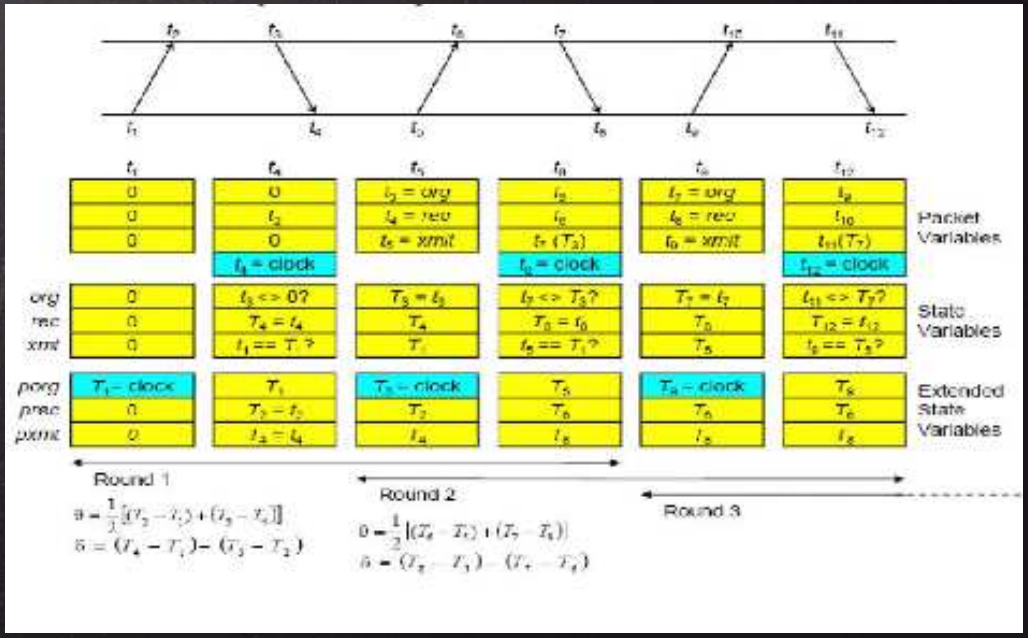


- In the diagrams, transmit timestamps carry odd numbers, receive timestamps carry even numbers.
- Receive timestamps are available immediately.
- In one-step mode, transmit timestamps are conveyed in the transmitted packet.
- In two-step mode, transmit timestamps are conveyed in the following transmitted packet.
- It takes two roundtrips to accumulate all four timestamps

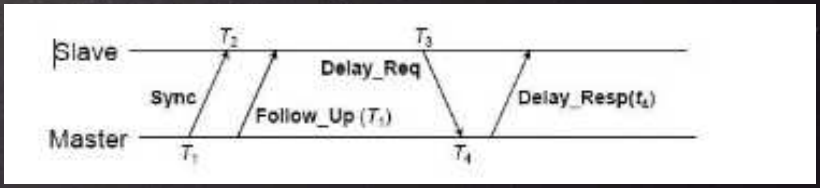
NTP one-step on-wire protocol



NTP two-step on-wire protocol

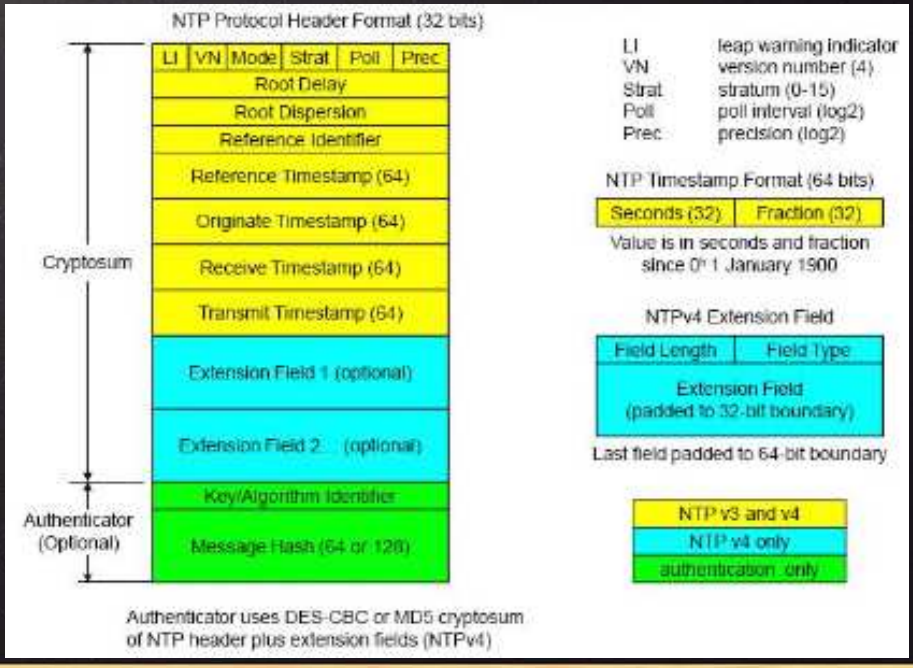


Master-slave protocol



- Ethernet NIC hardware strikes a timestamp after the preamble and before the data separately for transmit and receive.
- In each round master sends Sync message at T1; slave receives at T2.
- In one-step variant T1 is inserted just before the data in the Sync message; in two-step variant t1 is sent later in a Follow_Up message.
- Slave sends Delay_Req message at T3; master sends Delay_Resp message with T4. Compute master offset and roundtrip delay

NTP protocol header and timestamp formats



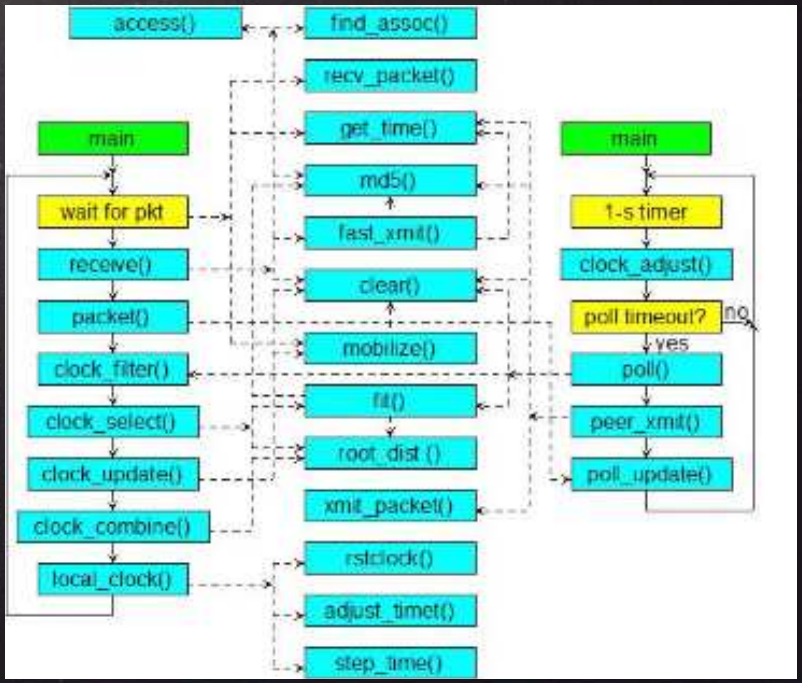
Flow Diagrams

1. Main Program
2. Peer Process
3. System Process
4. Clock Discipline Process
5. Clock Adjust Process
6. Poll Process

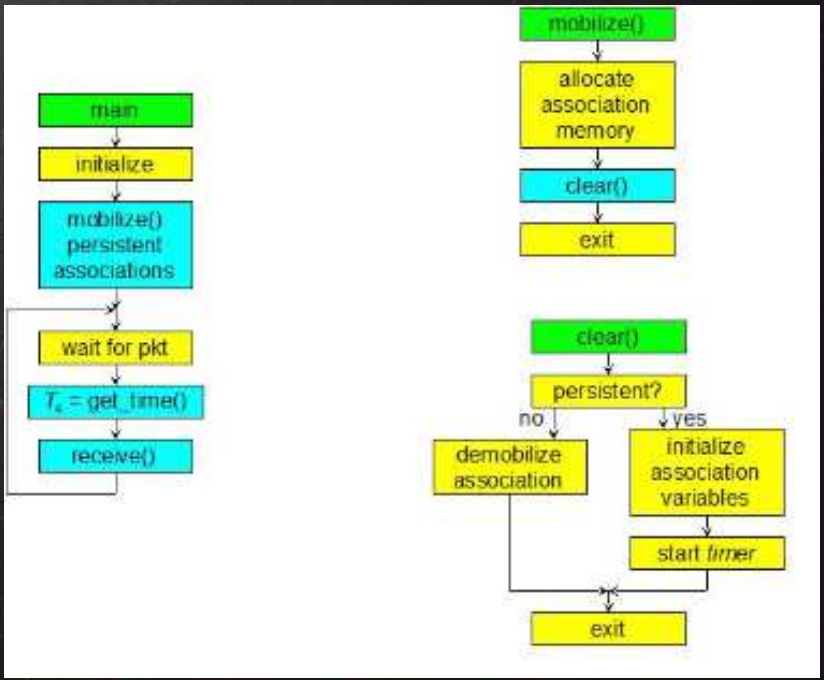
Control flow

- The main program waits for a packet arrival, then control flows by each of the procedures connected by solid arrows.
- A client request requires no persistent association; the server response is handled directly by fast_xmit.
- The packet procedure calls poll_update since it updates the packet poll variable.
- The main program waits for one second, then calls clock_adjust.
- At the poll timeout, control flows by each of the procedures connected by solid arrows.
- The peer_xmit procedure calls clock_filter when the server has not been heard for three poll intervals. It calls clear on timeout for ephemeral associaitons.
- The dotted arrows show which procdures are called by each procedure with control returning to the calling procedure.

Procedure flow



Main program



System process

- clock_select procedure
- select algorithm: classify available servers as truechimers or falsetickers.
- cluster algorithm: find and discard outlyers until no more than three survivors remain.
- clock_update procedure
- call clock_combine procedure to combine weighted server offsets.
- Call local_clock procedure to discipline the system clock.
- Update system variables
- rootdist function
- Return synchronization distance to the primary reference source.
- fit function
- Return TRUE if selected server is acceptable and root distance less than 1s

Clock discipline process

- local_clock() function
- Discipline system clock using adaptiver-parameter, phase/frequency-lock loop.
- rstclock procedure
- Transition to new state and initialize variables.
- adjust_freq segment
- Adjust oscillator frequency using PLL/FLL feedback loop.
- step_freq segment
- Step oscillator frequency when first starting and no previous information.
- tc segment
- Adjust time constant as a function of prevailing jitter and oscillator stability.

Clock adjust process: clock_adjust() procedure

- clock_adjust() procedure
- Called by kernel timer routines once each second.
- Adjusts system clock frequency as computed by PLL/FLL.
- system process computes initial system clock offset.
- Reduce residual clock offset as exponential decay.
- This procedure can also be implemented in the kernel for reduced sawtooth error.

Poll process

- poll() procedure
- Determine when to transmit a packet according to poll and burst schedules.
- peer_xmit() and fast_xmit() procedures
- Format and transmit an NTP packet.
- poll update() procedure
- Mitigate the poll interval as a function of the host and peer poll intervals and defined lower and upper limits.

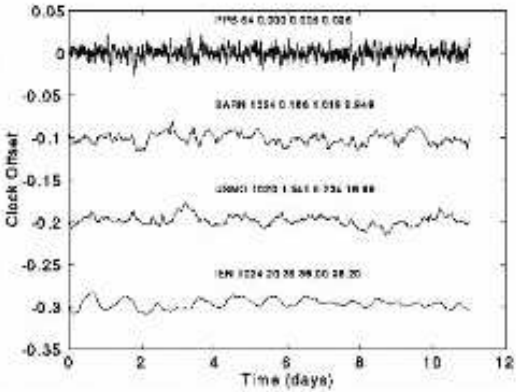
NTP secure group principles

- A NTP secure group is a subnet using a common security model, authentication protocol and identity scheme based on symmetric key or public key cryptography.
- Each group host has
  - For public key cryptography, a public/private host key pair and self-signed host certificate.
  - Optional password-encrypted identity parameters.
- Each group has one or more trusted hosts that
  - Provide cryptographic redundancy and diversity.
  - Operate at the lowest stratum of the group.
  - For public key cryptography, the host certificate must have a trusted extension field.
- A trusted agent acting for the group generates the identity parameters, which are distributed to other group hosts by secure means..

NTP security model

- NTP operates in a mixed, multi-level security environment including symmetric key cryptography, public key cryptography and unsecured.
- NTP timestamps and related data are considered public values and never encrypted.
- Time synchronization is maintained on a master-slave basis where synchronization flows from trusted servers to dependent clients possibly via intermediate servers operating at successively higher stratum levels.
- A client is authentic if it can reliably verify the credentials of at least one server and that server messages have not been modified in transit.
- A client is proventic if by induction each server on at least one path to a trusted server is authentic.

NTP performance compared



Legend

IEN	IEN Torino, Italy
USNO	US Naval Observatory, Wash DC
BARN	local time server on Dcnet
PPS	PPS signal (64-s poll clamp)

Label Format

1	server name
2	mean poll interval (s)
3	mean error (ms)
4	RMS error (ms)
5	max error (ms)

All servers are synchronized to GPS
All NTP algorithms are operative

Typical performance of stratum-2 servers synchronized to remote primary servers

Performance of typical NTP servers in the global Internet

NTP Server	Location	Days	Mean	RMS	Max	>1	>5	>10	>50
Austron GPS	DCnet	91	0.0	0.012	1.000	0	0	0	0
rackaty	DCnet	95	0.066	0.053	2.054	11	0	0	0
mizbeaver	DCnet	17	0.150	0.171	1.141	2	0	0	0
churchy	DCnet	42	0.185	0.227	3.150	15	0	0	0
pogo	DCNet	88	0.091	0.057	1.588	8	0	0	0
beauregard	DCnet	187	0.016	0.108	2.688	30	0	0	0
umd1	U Maryland	78	4.266	2.669	35.89	29	29	28	0
swifty	Australia	84	2.364	56.70	3944	27	27	27	13
ntps1	Germany	70	0.810	10.86	490.9	12	12	12	6
time_a	NIST Boulder	85	1.511	1.686	80.56	28	19	11	2
fuzz	San Diego	77	3.889	2.632	47.59	27	27	23	0
la	Los Angeles	83	0.650	0.771	17.84	28	8	3	0
enss136	NSFnet WashDC	88	0.657	1.203	32.65	38	23	10	0

- Table shows number days surveyed, mean absolute offsets (ms), RMS and maximum absolute error (ms) and number of days on which the maximum error exceeded 1, 5, 10 and 50 ms at least once
- Servers represent LANs, domestic WANs and worldwide Internet
- Results show all causes, including software upgrades and reboots

Present Status

A major milestone has been passed with new support for the IPv6 addressing family in addition to the original IPv4 addressing family.

Manycast mode support has been refined and tested in a production environment

Autokey public key cryptography support has been completed

References:

- Continuous clock amortization need not affect the precision of a clock synchronization algorithm, August 1990- Frank Schmuck,Flaviu Cristian
- Fault-tolerant clock synchronization in distributed systems- Ramanathan, P.; Shin, K.G.; Butler, R.W.
- Clock Synchronization in a Local Area Network, R. Gusella and S. Zatti, University of California, Berkeley
- Byzantine Clock Synchronization - Leslie Lamport,P. M. Melliar-Smith
- http://www.ntp.org/
- http://toi.iriti.cnr.it/uk/ntp.html
- http://www.eecis.udel.edu/~mills/ntp.html