Universität Stuttgart

Physical Time

Marco Aiello and Ilche Georgievski

Physical clock's usage

- temporal ordering of events produced by concurrent processes
- synchronisation between senders and receivers of messages
- serialisation of concurrent access to shared objects
- to know the time an event occurred at a node
- but… *no global clock in a distributed system*

Logical vs. physical clocks

- **Logical Clock** keeps track of the ordering of events, used for causality

- **Physical Clocks** keep track of the time of day, used for timestamping

History of time keeping

Timekeeping tied to astronomical observation

Egyptian sundial
XIII Century BC

Greek water clock
V Century BC

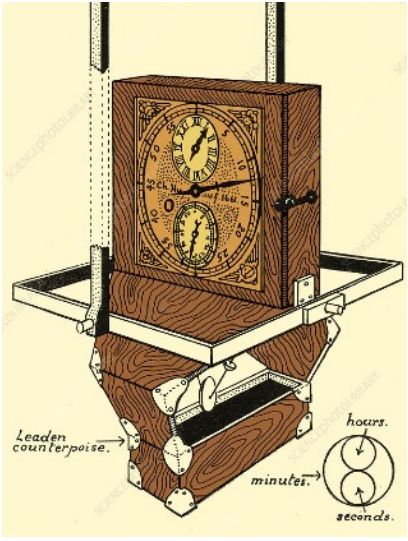
Chinese Incense/Fire Clock
VI Century

History of time keeping

Timekeeping tied to astronomical observation

Medieval sandclock

Salisbury Cathedral
verge and foliot
XV century

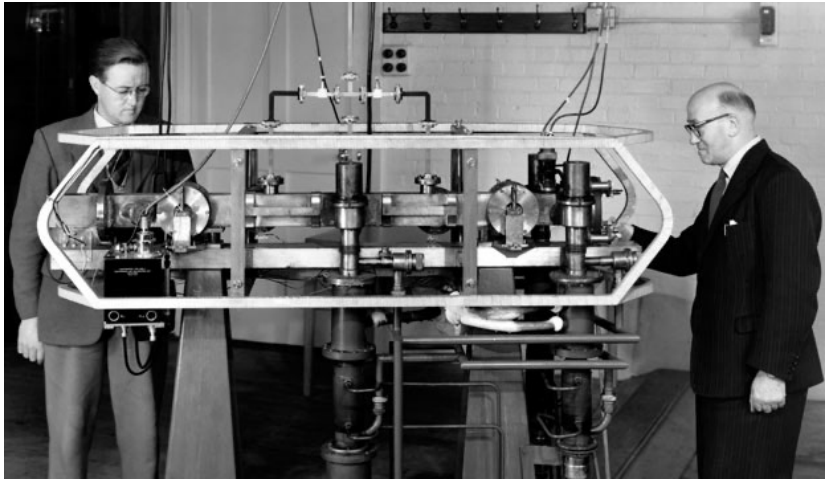


Huygens Pendulum clock
1656

History of time keeping

Timekeeping tied to astronomical observation

Horton and Marrison
Quartz clock
Bell Labs
1927



First caesium-133 atomic clock
National Physical Laboratory, London
1955

Clock characteristics

- **Resolution:** the minimal unit of time measured, ticking period
- **Drift rate:** is the relative amount that a computer clock differs from a perfect clock over time
- **Stability:** slope of drift rate curve (stable if flat)
- **Skew:** the difference in time between two clocks

	Age	Core technique	Resolution	Drift Rate
Sundial	XIII BC	Sun casting shadow	minutes (2)	None
Water clock	XII BC	Gravity on water	hour (1/2)	Depends on water temperature
Incense clock	VI	Burning of incense speed	minutes (5)	N.A.
Sand clock	VIII	Gravity on sand	minutes (1/2)	10% each run
Verge and foliot	XIII	oscillator under gravity force	minutes	15 min/day
Pendulum	1656	time taken by a pendulum to oscillate	seconds	15 seconds/day
Quartz	1927	quartz clock crystals vibrate at a frequency of 32768 Hz at 31°C	milli/microseconds 10^{-5} s	0.5 seconds/day
First Atomic	1955	transition between two energy levels of the caesium-133 atom, 9 192 631 770 cycles of radiation = 1 second	nano/picoseconds 10^{-10} s	1 second/6 million years
Photon	2020	time takes a photon to cross a hydrogen molecule	attoseconds 2.47×10^{-19} s	N.A.

Coordinated Universal Time (UTC)

1961

- Derived from International Atomic Time (TAI)
- Adjusted to UT1 by adding leap seconds

Broadcasted:

- land-based signals are accurate to about 0.1-10 milliseconds
- satellite-based signals are accurate to about one microsecond

Standards:

- **UT0** - obtained from astronomical observations
  UT1 - UT0 corrected for polar motion, i.e., the actual Earth's rotation with
- respect to the solar time
- **UT2** - UT1 corrected for seasonal variations in Earth's rotation

UT2 - Leap Seconds

The One-second War (What Time Will You Die?)

Leap Seconds Inserted into the UTC Time Scale

Date	MJD	Date	MJD	Date	MJD	Date	MJD
2016-12-31	57753	1998-12-31	51178	1989-12-31	47891	1979-12-31	44238
2015-06-30	57203	1997-06-30	50629	1987-12-31	47160	1978-12-31	43873
2012-06-30	56108	1995-12-31	50082	1985-06-30	46246	1977-12-31	43508
2008-12-31	54831	1994-06-30	49533	1983-06-30	45515	1976-12-31	43143
2005-12-31	53735	1993-06-30	49168	1982-06-30	45150	1975-12-31	42777
		1992-06-30	48803	1981-06-30	44785	1974-12-31	42412
		1990-12-31	48256			1973-12-31	42047
						1972-12-31	41682
						1972-06-30	41498

After many years of discussions by different standards bodies, in November 2022, at the 27th General Conference on Weights and Measures, it was decided to abandon the leap second by or before **2035**.


practice

DOI:10.1145/1941487.1941505

Article development led by queue.acm.org

Finding a lasting solution to the leap seconds problem has become increasingly urgent.

BY POUL-HENNING KAMP

The One-Second War

THANKS TO A secretive conspiracy working mostly below the public radar, your time of death may be a minute later than presently expected. But don't expect to live any longer, unless you happen to be responsible for time synchronization in a large network of computers, in which case this coup will lower your stress level a bit every other year or so.

We're talking about the abolishment of leap seconds, a crude hack added 40 years ago to paper over the fact that planets make lousy clocks compared with quantum mechanical phenomena.

Timekeeping used to be astronomers' work, and the trouble it caused was very academic. To the rural

population, sunrise, midday, and sun
set were plenty precise for all relevan
purposes.

Timekeeping became a problem fo
non-astronomers only when ships start
ed to navigate where they could not se
land. Finding your latitude is easy: mea
sure the height of the midday sun ove
the horizon, look at the table in your al
manac, done. Finding your longitude i
possible only if you know the time of da
precisely, and the sun will not tell yo
that unless you know your longitude.

If you know your longitude, however
the sun will tell you the time very pre
cisely. Using that time, you can maku
tables of other nonsolar astronomica
events—for example, the transits of the

Computer clocks

- **Hardware clock:** CMOS clock circuit that counts oscillations of a quartz, after a specified number of oscillations, the clock increments a register, thereby adding one clock-tick to a counter that represents the passing of time: $H_i(t)$ battery backup to continue measuring when computer power is off

- **Software clock:** OS reading of the hardware clock

$$C_i(t) = \alpha H_i(t) + \beta$$

- $C_i(t)$ approximates the physical clock t at process p_i
- e.g., a 64-bit number giving nanoseconds since some base time in UNIX, base time is January 1st, 1970, at midnight (UTC)

Clock correctness

- H_i is said to be **correct** if its drift rate is within a bound $\rho > 0$
  (e.g., 10^{-6} secs/sec)

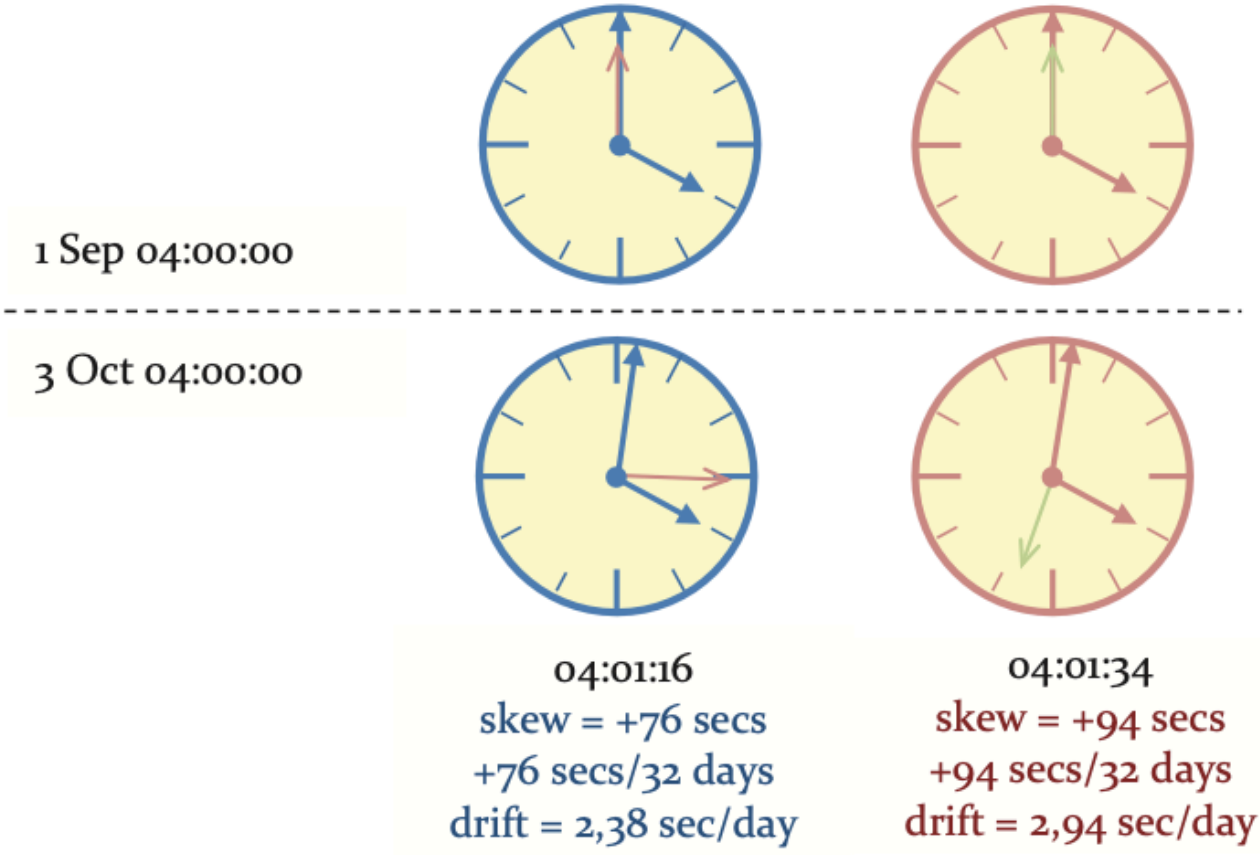
- The error in measuring the interval between real times t and t' is bounded:
  $$(1 - \rho)(t' - t) \leq H_i(t') - H_i(t) \leq (1 + \rho)(t' - t)$$
  forbids jumps in time readings of physical clocks

- Weaker condition of monotonicity:
  $$t' > t \implies C_i(t') > C_i(t)$$
  e.g., required by UNIX make operation
  achievable with a fast clock by adjusting α, β

- A **faulty** clock is one that does not obey its correctness condition
  - crash failure - a clock stops ticking
  - arbitrary failure - any other failure e.g., jumps in time

Clock correctness

Examples

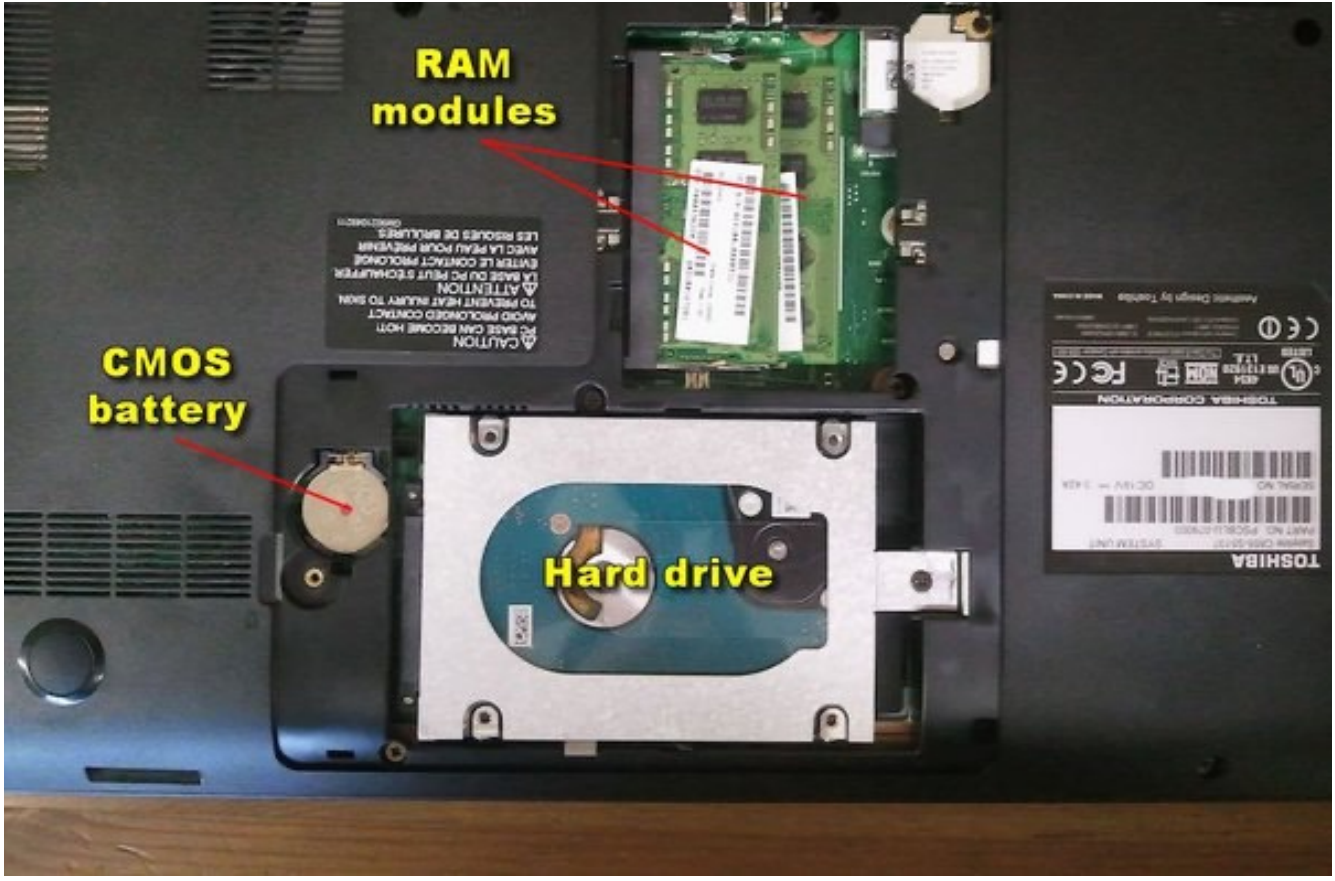


Clock correctness

Examples

Clock correctness

Examples



How to get accurate time?

Cost, power, reliability factors.

- GPS receiver per node (1 microsecond of UTC)
- Radio receiver (0.1-10 milliseconds of UTC)
- Synchronize with another machine with better time

GPS

- GPS can be used for clock synchronisation where there is reach.

Network accuracy: ± 20ms typical
Signal (GPS) accuracy: <1 µs, relative to GPS

Clock Synchronisation

External Synchronization

Process's clock is synchronised with an external authoritative time source S:

$$|S(t) - C_i(t)| < \delta$$
$$\delta > 0$$
$$\text{for } i = 1,2,\ldots,N$$

The clock $C_i(t)$ is accurate up to the bound δ

Internal Synchronization

Process's clock are synchronized with one another:

$$|C_i(t) - C_j(t)| < \delta$$
$$\delta > 0$$
$$\text{for } i, j = 1,2,\ldots,N$$

The clocks $C_i(t)$ and $C_i(t)$ agree within the bound δ

- Are internally synchronised clocks also externally synchronised?
- If a set of processes P is synchronised externally within a bound , is then P also internally synchronised? If yes, what is the bound?

Basic synchronization

- Server S sends its local time T_s to a client C in a message m
- C could set its clock to $T_s + T_t$, where T_t is the time to transmit m
- T_t is unknown, but in a synchronous system $T_{min} < T_t < T_{max}$
- Let u be the uncertainty in the message transmission time, so that $u = T_{max} - T_{min}$ Then, client sets the clock to $T_s + \frac{T_{min} + T_{max}}{2}$ so that $skew \leq \frac{u}{2}$

Basic synchronization

- But what if we are in an asynchronous system?

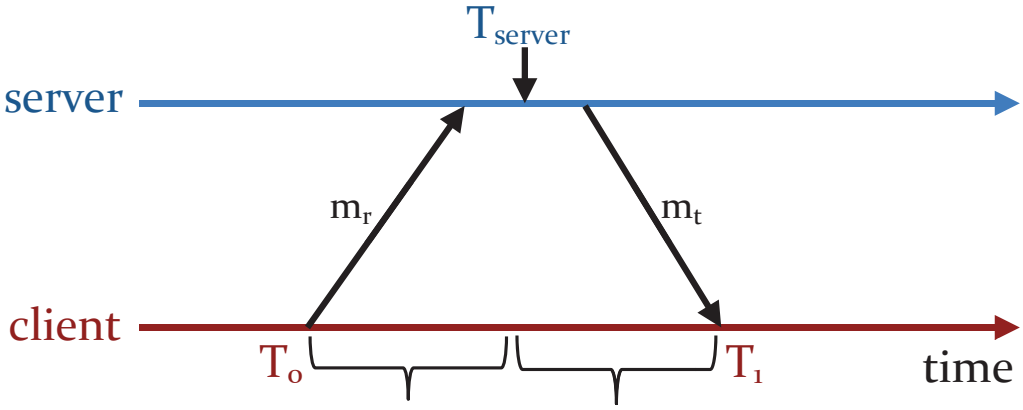
then we can only say $T_t = T_{min} + x$ where $x \geq x$

Synchronization

- Christian’s Algorithm, 1989
- Berkeley Algorithm, 1989
- Network Time Protocol, 1988
- Precision Time Protocol, 2002

Steps

1. A time server receives signals from a UTC source
2. Client requests time in m_r and receives T_{server} in m_t from the server
3. The client sets its clock to $T_{new} = T_{server} + \frac{T_{round}}{2}$

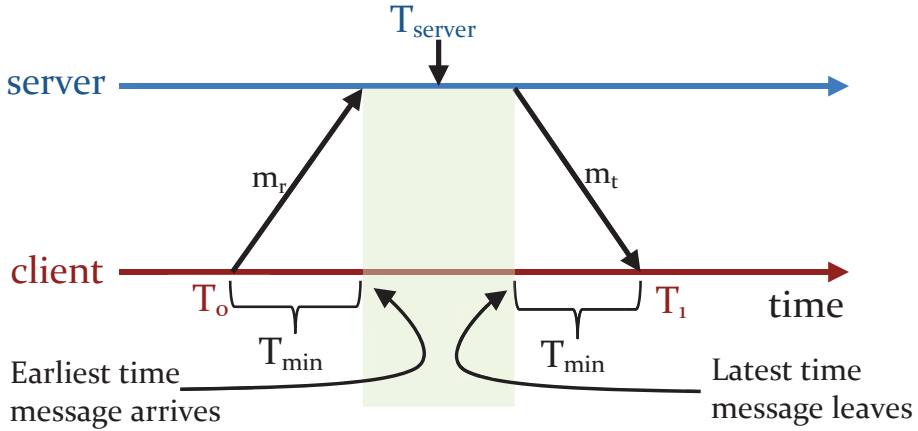


T_{round} is the round-trip time recorded by the client, i.e., $T_{round} = T_1 - T_0$

Accuracy

Let T_{min} be the minimum message delay (transition time). Then,

- T_{min} is the earliest time the server puts T_{server} in message m_t after the client sent m_r, and
- T_{min} is the latest time before m_t arrived at the client, and
- $[T_{server} + T_{min}, T_{server} + Tround - T_{min}]$ is the range of the time by the server's clock when m_t arrives.



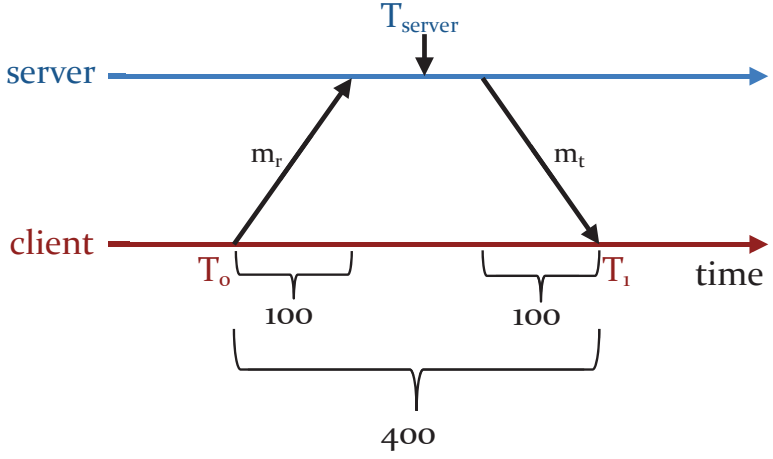
$$\text{Accuracy} = \pm(T_{round}/2 - T_{min})$$

What is the potential problem in using a single time server?

Example

1. request sent at (T_0): 04:01:16.200
2. response received at (T_1): 04:01:16.600
   ➠ response contains time (T_{server}): 04:02:30.200

- round time: $T_1 - T_0$
  ➠ 04:01:16.600-04:01:16.200 = 400 msecs
- best guess: the timestamp was generated 200 msecs ago
- set time (T_{new}): 04:02:30.200+200 = 04:02:30.400

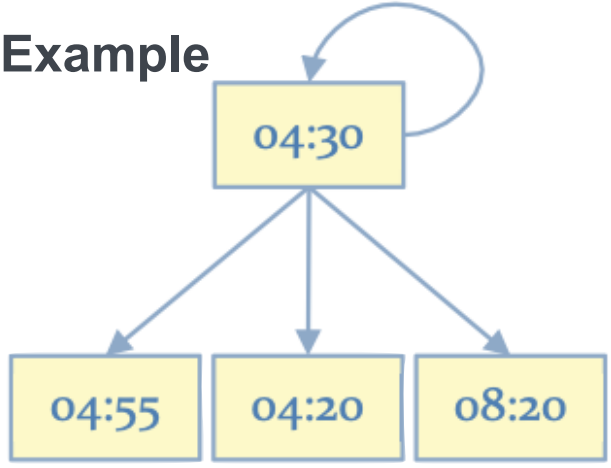


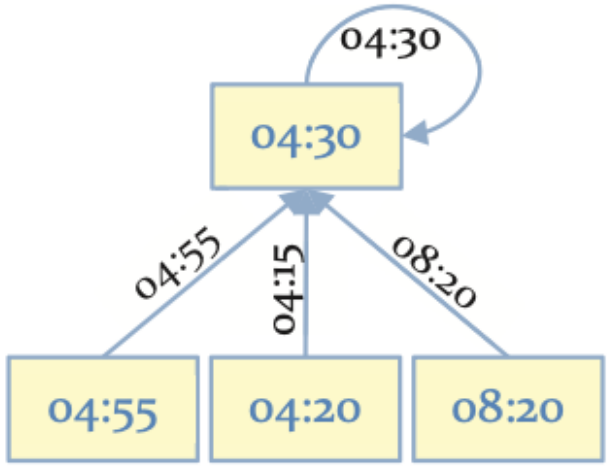
$$accuracy = \pm\frac{600-200}{2} - 100 = \pm 200 - 100 = \pm 100$$

Berkeley Algorithm

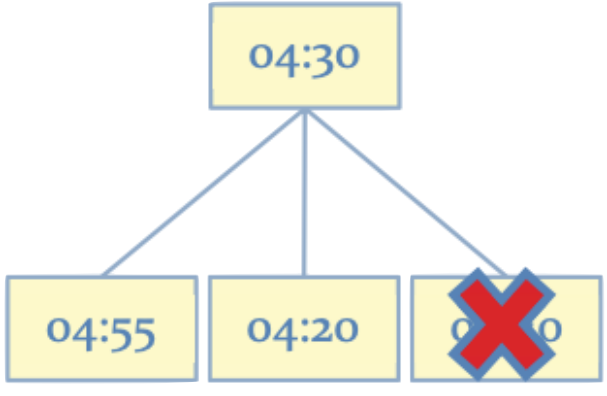
- Internal synchronisation of a group of computers
- Assumes no machine has an accurate time source
- Each machine runs time dæmon
- One machine represents a server (master), others are slaves

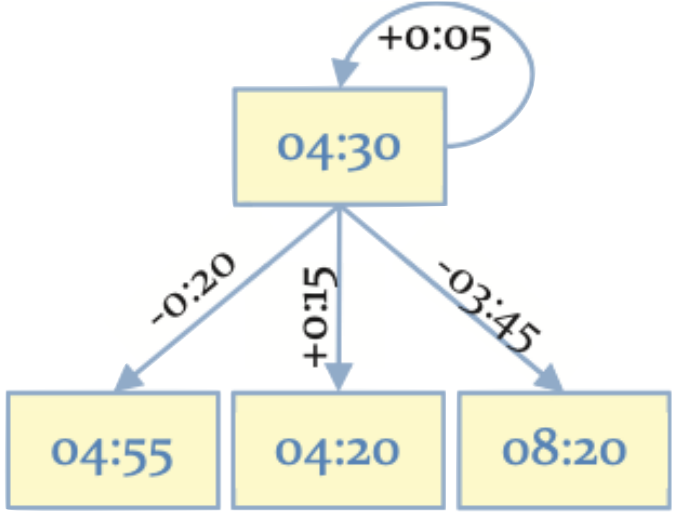
Example





$$\text{fault tolerant average} = \frac{04:30 + 04:55 + 04:20}{3} = 04:35$$





Network Time Protocol

History

A time service that played a large role in time synchronisation by keeping networked computer clocks synchronised to within milliseconds of each other.

- 1988 (v1): RFC 1059, clock filter, selection and discipline algorithms, client/server and symmetric modes
- 1989 (v2): RFC 1119, formal model, pseudo-code, Control Message Protocol, cryptographic authentication,
- 1992 (v3): RFC 1305, formal error analysis, timekeeping quality, broadcast mode, reference clock drivers
- 2010 (v4): RFC 5905-5908,IPv4, IPv6 and OSI support, improved accuracy to tens of microseconds, dynamic server discovery, manycast mode

Network Time Protocol

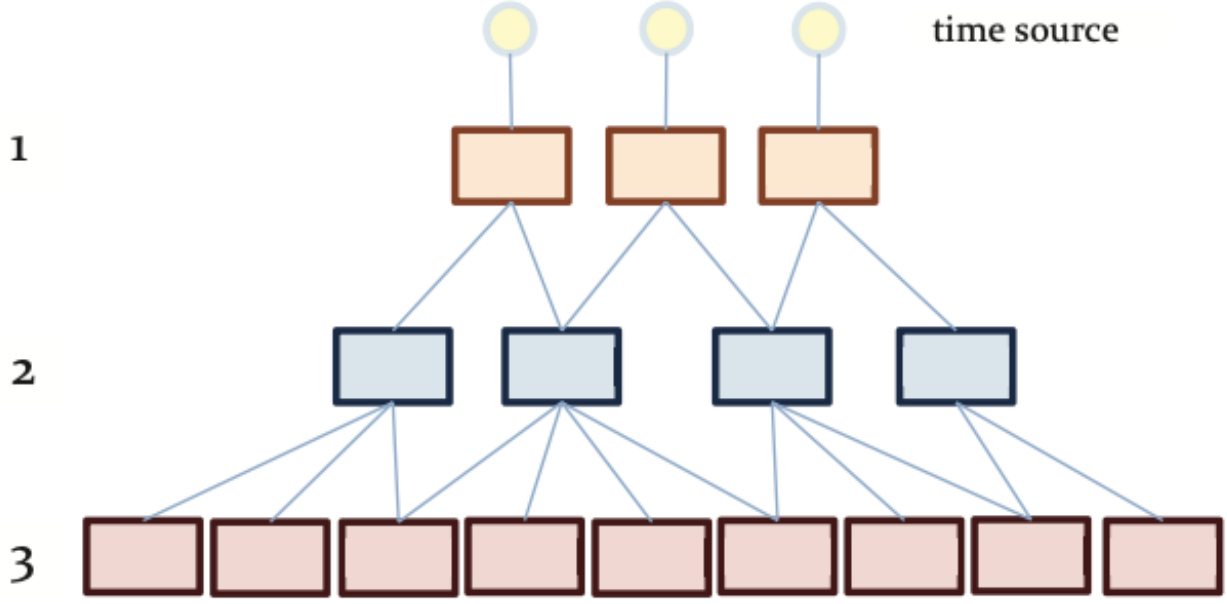
Design goals

- Enable clients across the Internet to be synchronised accurately to UTC
- Provide a reliable service that can handle extensive losses of connectivity
- Enable clients to resynchronise frequently enough
- Protect against interference with the time service

Network Time Protocol

Architecture

- Stratum 1: primary servers connected directly to UTC
- Stratum 2: secondary servers synchronised to primary servers



Network Time Protocol

Modes

- Multicast mode
  - a server within a high-speed LAN multicasts time
  - not very accurate, but efficient
- Procedure-call mode
  - similar to Cristiain's algorithm
  - higher accuracy, useful if no hardware multicast
- Symmetric mode
  - pairs of servers exchange messages containing time information
  - very high accuracy

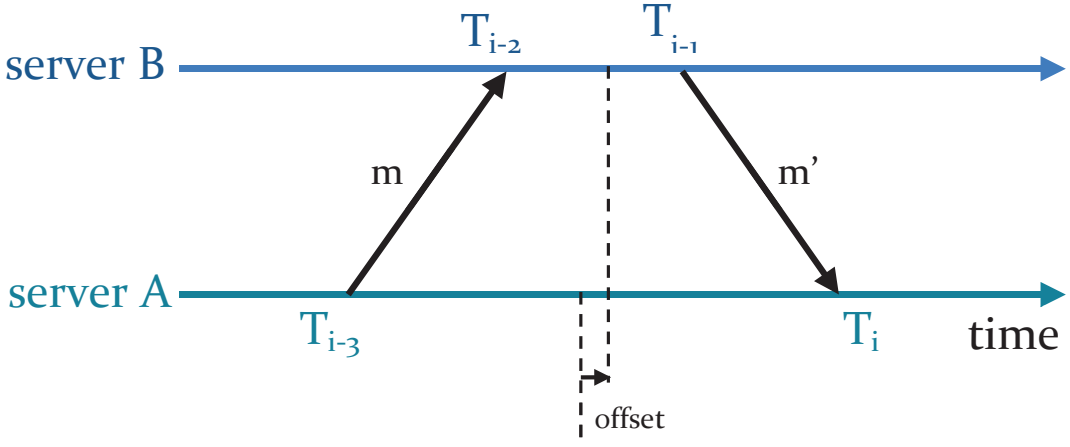
In all modes, UDP is used!

Synchronisation strategy (message exchange)

- Each message bears timestamps of recent events:

T_{i-3}: local time when the previous message was sent
T_{i-2}: local time when the previous message was received
T_{i-1}: local time when the current message was sent

- Recipient notes the time of receiving the message, T_i
- In the symmetric mode, there can be a non-negligible delay between messages



Accuracy

- For each pair of messages between two servers, NTP calculates an *estimated offset* Θ_i and a *delay* δ_i.

Let two servers exchange a pair of messages, and let Θ be the *actual offset* between the servers' clocks and t, t' the transmission times of the messages. Then,

$$T_{i-2} = T_{i-3} + t + \Theta \text{ and } T_i = T_{i-1} + t' - \Theta.$$

The *delay* equals the total transmission time

$$\delta_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}.$$

The *estimated offset* is

$$\Theta_i = \frac{(T_{i-2} - T_{i-3}) + (T_{i-1} - T_i)}{2}.$$

The *actual offset* is $\Theta = \Theta_i + \frac{(t' - t)}{2}$.
Since $t > 0$ and $t' > 0$, the value of $t' - t$ must be between δ_i and $-\delta_i$. Therefore,

$$\Theta_i - \frac{\delta_i}{2} \leq \Theta \leq \Theta_i + \frac{\delta_i}{2},$$

So, Θ_i is an estimate of the offset and δ_i is a measure of the accuracy.

Network Time Protocol

Design goals

- NTP servers filter pairs $< \Theta_i, \delta_i >$ by estimating reliability from variation, which allows them to select peers. The eight most recent pairs are saved. It selects the Θ_i, with the smallest δ_i (the smaller delay, the better accuracy).

- Accuracy of tens of milliseconds over Internet paths and 1 millisecond on LANs (v3)

Precision Time Protocol

History

A packet-based synchronisation mechanism able to synchronise LAN-networked computer clocks within tens of nanoseconds of each other.

- 2002 (v1): local networks
- 2008 (v2): improved accuracy, precision and performance

Precision Time Protocol

Design goals

- Provide sub-microsecond synchronisation of real-time clocks in distributed (measurement and control) systems
- Applicable to LANs supporting multicast communication
- Provide a simple, administration-free installation
- Support heterogeneous systems of clocks
- Impose minimal resource requirements on networks and hosts

Precision Time Protocol

Approach

- Master-slave hierarchy
- Master
  - time reference for one or more slaves
  - selected by Best Master Clock (BMC) algorithm
- Devices: ordinary clock, boundary clock, transparent clocks, etc.

Precision Time Protocol

Devices

- **Ordinary clock**
  - * a device with a single network connection
  - * either a source or destination for a synchronisation reference

- **Boundary clock**
  - * a device with multiple network connections
  - * accurately bridges synchronisation from one network segment to another – distributes a master clock to different parts of the network
  - * contains a timekeeper and multiple ports

- **Transparent clock**
  - * a device that connects a group of devices without segmenting the PTP network exposes its slave devices to the PTP master
  - * improves distribution accuracy

Precision Time Protocol

Configurations and modes

- **Software-only configuration** – ordinary clocks

- **Hardware timestamping configuration** – ordinary clocks, boundary clocks, transparent clocks

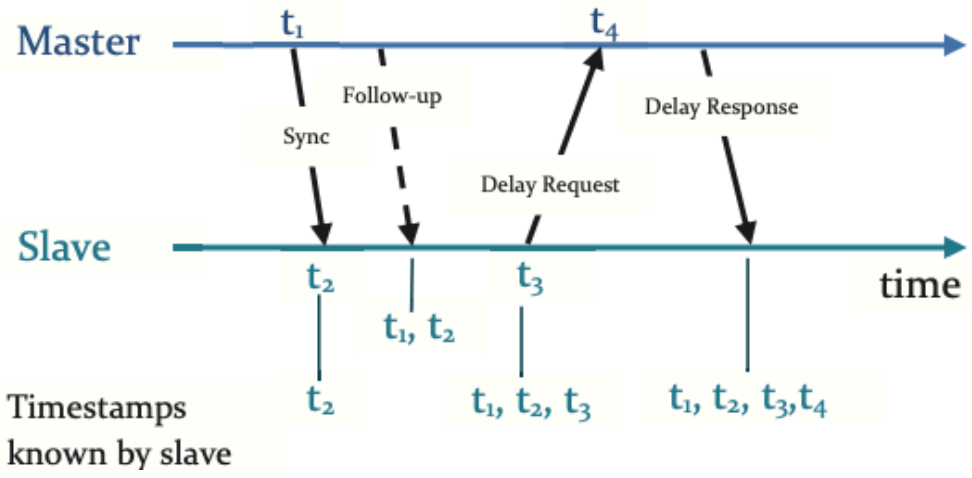
- **End-to-end mode** – a slave issues a Delay Request message and a master responds with a Delay Response

- **Peer-to-peer mode** – a device issues a Peer Delay Request message to an immediate neighbour (not necessarily master) which responds with Peer Delay Response
    * better performance when network traffic
    * better accuracy

Precision Time Protocol

Software only configuration in end-to-end mode

- Master: periodically transmits a *Sync* message by UDP multicast
  - $t1$: time when the *Sync* message is sent by the master
  - $t2$: time when the *Sync* message is received by the slave
  - $t1$: actual time when the *Sync* left the master contained in the *Follow-up* message
  - $t3$: time when the *Delay Request* message is sent by the slave
  - $t4$: time when the *Delay Request* message is received by the master
- Master: sends a *Delay Response* to the slave containing $t4$



GPS Excercise, from an exam

Time adjustments

- Satellites emit messages at a constant rate of 50bit/second of 1500 bits in size. Divided into five 6s subframes of ten 30-bit words each. Each subframe has the GPS time in 6-second increments.
- Satellites are equipped with atomic clocks that tick every nanosecond.
- Due to relativity theory, satellite clocks will tick more rapidly by about 45.9 μs/day day because they have a higher gravitational potential, but they will also tick more slowly by about 7.2 μs/day due to their relative speed to the earth.

Questions:

- How much time does a satellite need to emit one message (in seconds)?
- What is the satellite clock resolution (seconds)?
- What is the clock's drift rate due to the relativistic effect with respect to time on earth (seconds/second)?
- If with a 25 nanosecond error, a GPS receiver will provide its location with an accuracy of 6.46 meters, what will be the accuracy after one day if the relativistic effects are not compensated for (meters)?

Answers

- How much time does a satellite need to emit one message (in seconds)?

$$\frac{1.500\,b}{50\,b/s} = 30s \quad 6s \text{ per GPS timestamp}$$

- What is the satellite clock resolution (seconds)? $10^{-9}\,Hz$

more precisely, caesium 133 atom, 9.192.631.770 oscillations per second, i.e., $1{,}09 \cdot 10^{-10}\,Hz$

- What is the clock's drift rate due to the relativistic effect (seconds/second)?

$$45.9 - 7.2\,\mu s/\text{day} = 38.7\,\mu s/\text{day} = \frac{38.7}{24 * 60 * 60}\,\mu s/s = \frac{38.7 * 10^{-6}}{86.4 * 10^{3}}\,s/s = 0.45 * 10^{-9}\,s/s$$

- If with a 25 nanosecond error, a GPS receiver will provide its location with an accuracy of 6.46 meters, what will be the accuracy after one day if the relativistic effects are not compensated for (meters)?

$$25 * 10^{-9}\,s : 6.46\,m = 38.7 * 10^{-6}\,s : d$$

$$d = \frac{38.7 * 10^{-6} * 6.46}{25 * 10^{-9}}$$

$$d \approx 10 * 10^{3}\,m \quad = \quad 10\,Km$$