Time and Clocks

- **Time**: we model the continuum of real-time as a directed timeline consisting of an infinite set {T} of instants with the following properties:
  - {T} is an ordered set, i.e., if p and q are any two instants, then either (1) p is simultaneous with q or (2) p precedes q or (3) q precedes p and these relations are mutually exclusive. The order of instants on the timeline is called temporal order.
  - {T} is a dense set.
- **Duration**: A section of the timeline.
- **Event**: A happening at an instant of time.
- **Causal order**: If event e1 is a cause of event e2, then a small variation in e1 is associated with a small variation in e2, whereas small variations in e2 are not necessarily associated with small variations in e1.

Time and Clocks

- **Clock**: A clock c is a device that contains a counter and increments this counter periodically according to some law of physics (microticks).
- **Reference clock**: Let us assume there exists an external observer with an atomic clock that has a granularity that is much smaller than the duration of any intervals of interest (reference clock z).
  Granularity: The granularity of a clock is the number of microticks of z between any two consecutive microticks of c.

Time

- **Precision**: Given an ensemble of clocks {1,2, ..., n}, the maximum offset between any two clocks of the ensemble is called the precision of the ensemble at microtick i.
- **Accuracy**: The offset of clock k with respect to the reference clock z at tick i is called the accuracy.

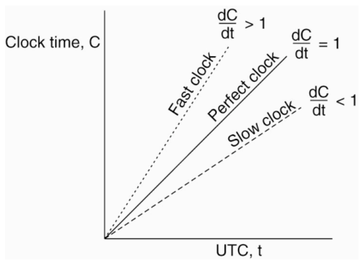
Time Standards

- **International Atomic Time (TAI):** TAI is a physical time standard that defines the second as the duration of 9 192 631 770 periods of the radiation of a specified transition of the cesium atom 133. TAI is a chronoscopic timescale, i.e., a timescale without any discontinuities. It defines the epoch, the origin of time measurement, as January 1, 1958 at 00:00:00 hours, and continuously increases the counter as time progresses.
- **Universal Time Coordinated (UTC):** UTC is an astronomical time standard that is the basis for the time on the "wall clock". In 1972 it was internationally agreed that the duration of the second should conform to the TAI standard, but that the number of seconds in an hour will have to be occasionally modified by inserting a leap second into UTC to maintain synchrony between the wall clock time and the astronomical phenomena, like day and night.

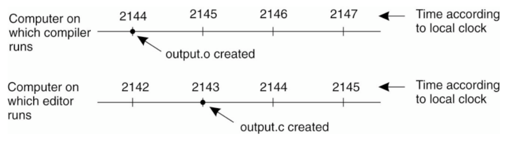
Clock Synchronization

- Why do we need to agree on time?
- Why do we need clock synchronization?
  - time initially set incorrectly
  - clock drifts over the course of time
- Clock synchronization:
  - internal vs external
  - master-slave vs distributed
  - hardware vs software
- Goals:
  - high precision: small deviation between clocks
  - high accuracy: small deviation to external time
  - monotonic and "continuous" time
- Clock drift:

Clock Drift



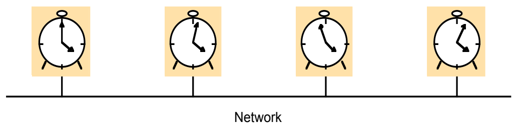
Why Clock Synchronization?


Computer on which compiler runs
2144
2145
2146
2147
Time according to local clock
output.o created
Computer on which editor runs
2142
2143
2144
2145
Time according to local clock
output.c created

Physical Clocks

- A high-frequency oscillator, counter, and holding register
  - Counter decrements on each oscillation, generates a “tick” and is reset from the holding register when it goes to 0.
- On a network, clocks will differ: “skew”.
- Two problems:
  - How do we synchronize multiple clocks to the “real” time?
  - How do we synchronize with each other?

Skew



- Computer clocks are not generally in perfect agreement
- *Skew*: the difference between the times on two clocks (at any instant)
- Computer clocks are subject to *clock drift* (they count time at different rates)
- Clock *drift rate*: the difference per unit of time from some ideal reference clock
- Ordinary quartz clocks drift by about 1 sec in 11-12 days. (10^{-6} secs/sec).
- High precision quartz clocks drift rate is about 10^{-7} or 10^{-8} secs/sec

Resynchronization

- ρ: rate of deviation from external time
- R: time between synchronizations
- Δs: precision after synchronization
- Skew increase by $R*2\rho$ between two synchronizations
- Choose R s.t. $\Delta s+R*2\rho <$ required precision
- Loosely coupled systems:
  - inaccurate, expensive clock readings: use high accuracy clocks, synchronize seldom
- Tightly coupled systems:
  - accurate clock readings: use low accuracy clocks and synchronize often
- Rate adjustment:
  - required adjustment: Δ
  - time between adjustments: R
  - adjust with rate Δ/R

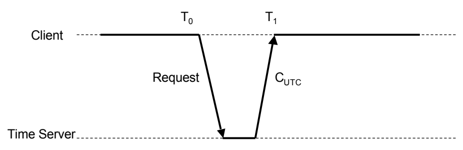
Synchronization in Synchronous System

- A synchronous distributed system is one in which the following bounds are defined:
  - the time to execute each step of a process has known lower and upper bounds
  - each message transmitted over a channel is received within a known bounded time
  - each process has a local clock whose drift rate from real time has a known bound

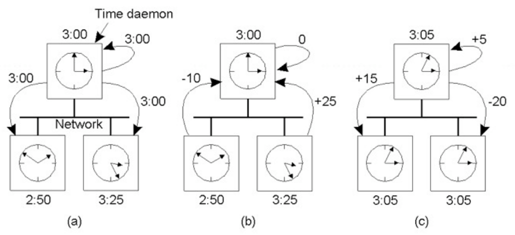
- Internal synchronization in a synchronous system
  - One process p_1 sends its local time *t* to process p_2 in a message *m*,
  - p_2 could set its clock to $t + T_{trans}$ where T_{trans} is the time to transmit *m*
  - T_{trans} is unknown but $min \le T_{trans} \le max$
  - uncertainty $u = max-min$. Set clock to $t + (max - min)/2$ then skew $\le u/2$

Asynchronous Systems: Cristian's Method

- A time server *S* receives signals from a UTC source
  - Process *p* requests time in m_r and receives *t* in m_t from *S*
  - *p* sets its clock to $t + T_{round}/2$



Berkeley Algorithm



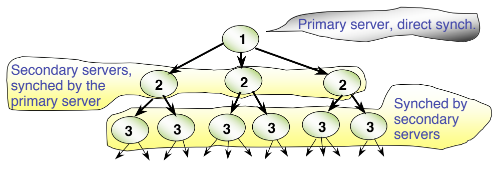
a) The time daemon asks all the other machines for their clock values.
b) The machines answer and the time daemon computes the average.
c) The time daemon tells everyone how to adjust their clock.

Averaging Algorithms

- Every *R* seconds, each machine broadcasts its current time.
- The local machine collects all other broadcast time samples during some time interval, *S*.
- **The simple algorithm**: the new local time is set as the average of the value received from all other machines.
- **Improved algorithm**: Correct each message by adding to the received time an estimate of the propagation time from the ith source (extra probe messages are needed to use this scheme).
- One of the most widely used algorithms in the Internet is the **Network Time Protocol (NTP).**
  - achieves worldwide accuracy in the range of 1-50 msec.

Network Time Protocol (NTP)

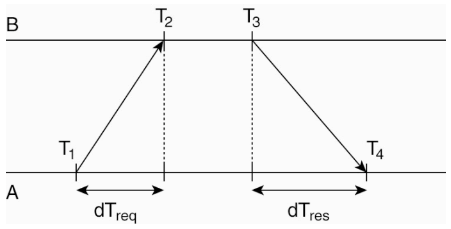
- Uses a network of time servers to synchronize all processors on a net.
- Time servers are connected by a synchronization subnet tree. The root is adjusted directly . Each node synchronizes its children nodes.



Primary server, direct synch.
1
Secondary servers, synched by the primary server
2
2
2
Synched by secondary servers
3
3
3
3
3
3

Network Time Protocol

- Contact a time server to ask the time.
  - What is the major issue here?
- Trick is to get a good estimate of the message delays.

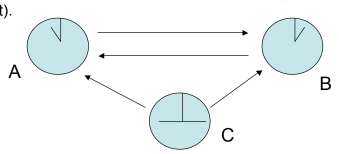


NTP Time Corrections

- NTP is about making clocks correspond to the real time.
- Clock servers are divided into stratums, starting with stratum-0 for the reference physical clock.
- If a stratum-2 clock is running faster than a stratum-1 clock, the stratum-2 clock will adjust.
- How should it adjust if it is running fast?
  - Time should never go backwards, but should just slow down.

Fault-Tolerant Clock Synchronization

- Central Master Algorithm: master clock sends periodically its time value to all other clocks (not fault-tolerant).
- Average Algorithm: each clock gets the time from all others and average difference and a local correction factor (almost fault-tolerant).



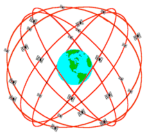
- Fault-Tolerant Average Algorithm: sort times and remove k smallest and k largest values, compute average difference from rest (tolerates k Byzantine clocks).

Quick Quiz

- Consider the client of a time server, using Cristian's method for clock synchronization. The local (client) time is 18 when a request is sent to the time server. The UTC time is returned as 23 and arrives at time 24 local (client) time. What will the client do?
- Consider the Berkeley algorithm with four clocks. The master clock shows 3:00, the other clocks show 2:50, 3:04, and 3:10. What do the clocks show after the algorithm is executed.

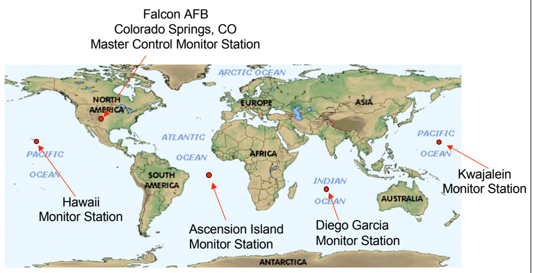
Very Common Now: GPS

- Satellite system launched by military in early 1990's, became public and inexpensive
- 24 active satellites, with atomic clocks, at 20,000 km
- Each satellite continuously broadcasts its position and time
- Can think of satellites as broadcasting the time
- Small radio receiver picks up signals from three satellites and triangulates to determine position
- Same computation also yields extremely accurate clock (accurate to a few milliseconds)



- Put two GPS receivers (or more) on a network
- Periodically, receivers broadcast the "true time"
- Other machines only need to adjust their clocks to overcome propagation delays for clock sync messages through the network!
- Well matched to the a-posteriori clock synchronization approach

GPS



GPS

- These stations are the eyes and ears of GPS, monitoring satellites as they pass overhead by measuring distances to them every 1.5 seconds
- This data is then smoothed using ionospheric and meteorological information and sent to Master Control Station at Colorado Springs
- The ionospheric and meteorological data is needed to get more accurate delay measurements, which in turn improve location estimation
- Master control station estimates parameters describing satellites' orbit and clock performance,. It also assesses health status of the satellites and determines if any re-positioning may be required
- This information is then returned to three uplink stations (collocated at the Ascension Island, Diego Garcia and Kwajalein monitor stations) which transmit the information to satellites

Basic Idea

- GPS receiver broadcasts “the time is now 10:00” on a broadcast network (ethernet)
- Receivers note the time when they receive the message: [10:01, 9:58, ...] and reply with values
- GPS receiver subtracts the median value
- Differences [1, -2, ...] now give the drift of the clock of the destination relative to the median clock
- Now we distribute these drift values back to the processors, which compensate for the rate of drift over the time period to the next synchronization
- Can show that this yields clocks that are optimal both in accuracy and precision
- A processor with a broken clock has a good chance of discovering it during synchronization