Physical Clock Synchronization

Physical Clocks

- It is impossible to guarantee that crystals in different computers all run at exactly the same frequency. This difference in time values is **clock skew**.
- "Exact" time was computed by astronomers
  - The difference between two transits of the sun is termed a **solar day.** Divide a solar day by 24*60*60 yields a **solar second.**
- However, the earth is slowing! (35 days less in a year over 300 million years)
- There are also short-term variations caused by turbulence deep in the earth's core.
  - A large number of days **(n)** were used to calculate the average day length, then dividing by 86,400 to determine the **mean solar second.**

Physical Clocks

- Physicists take over from astronomers and count the transitions of cesium 133 atom
  - 9,192,631,770 cesium transitions = 1 solar second
  - 50 International labs have cesium 133 clocks.
  - The *Bureau Internationale de l'Heure (BIH) averages* reported clock ticks to produce the **International Atomic Time (*TAI*)**.
  - The TAI is mean number of ticks of cesium 133 clocks since midnight on January 1, 1958 divided by 9,192,631,770 .

Physical Clocks

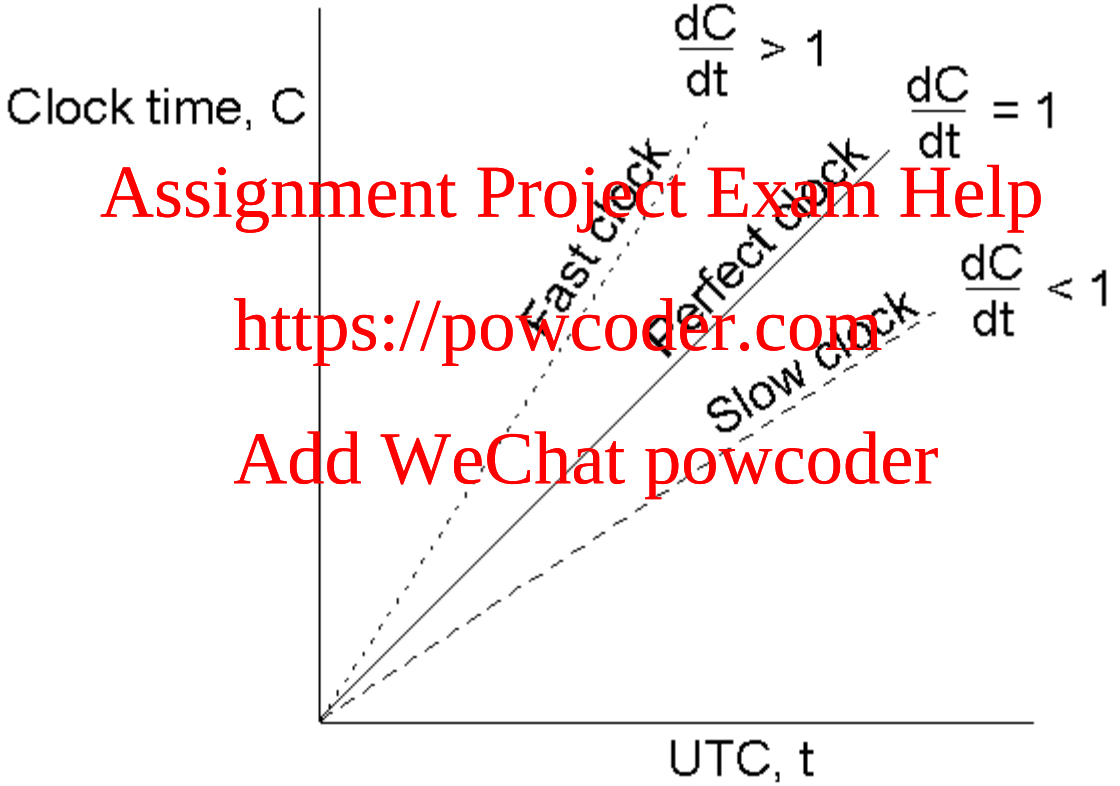
- To adjust for lengthening of mean solar day, **leap seconds** are used to translate TAI into **Universal Coordinated Time (*UTC*)**.
- UTC is broadcast by NIST from Fort Collins, Colorado over shortwave radio station WWV. WWV broadcasts a short pulses at the start of each UTC second.
- GEOS (Geostationary Environment Operational Satellite) also offer UTC service.

Clock Synchronization

- The internal timer causes an interrupt H times a second.
- When interrupt occurs, clock is incremented, and clock keeps time since it was initialized.
- The value of clock on machine p is $C_p(t)$.
- In reality, timers have drift ρ

  therefore: $1- \rho \leq dC_p(t)/dt \leq 1+ \rho$
- Two clocks can drift $2 \rho \Delta t$ after Δt.
- If we want clocks not to differ by more than δ, then resynch at least every $\delta/2\rho$ secs

Clock Synchronization Algorithms

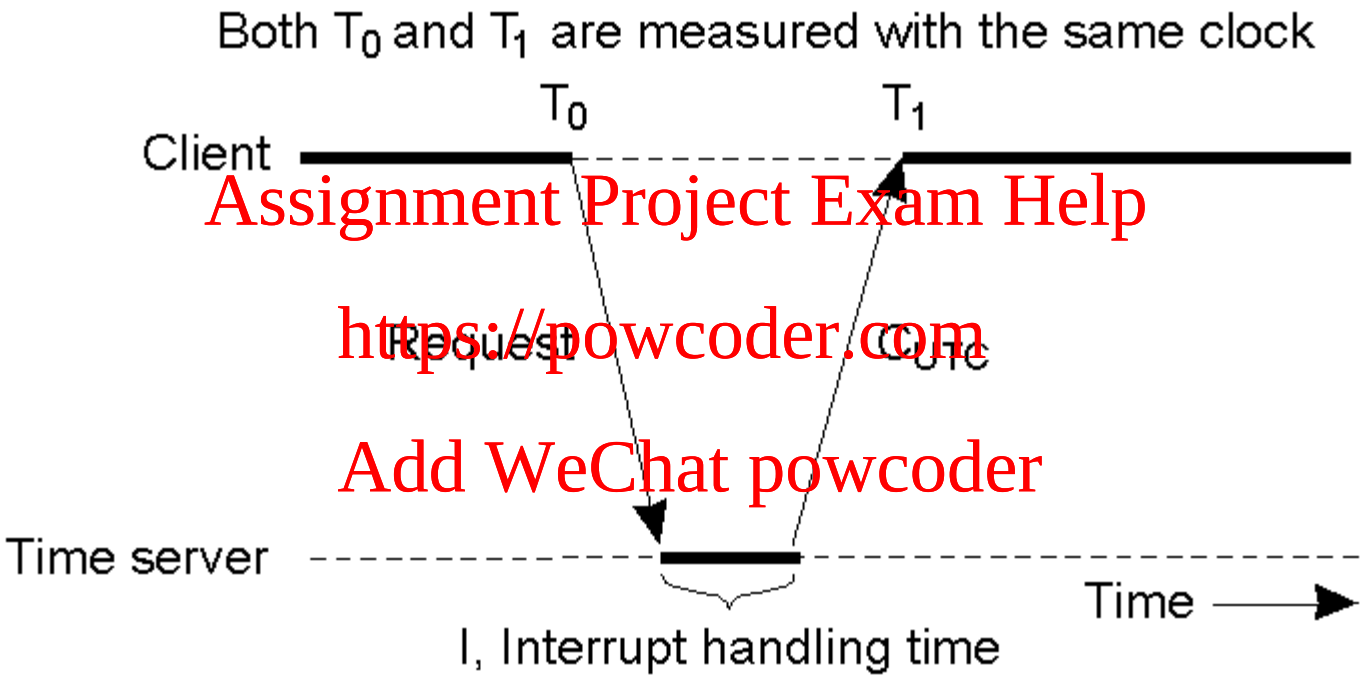


- The relation between clock time and UTC when clocks tick at different rates.

Cristian's Algorithm [Distributed Computing 1989]

- *Assume one machine (the time server) has a WWV receiver and all other machines are to stay synchronized with it.*
- Every $\delta/2\rho$ seconds, each machine sends a message to the time server asking for the current time.
- Time server responds with message containing current time, C_{UTC}.

Cristian's Algorithm



Getting the current time from a time server

Cristian's Algorithm

- Problem – the one-way delay from the server to client is "significant" and may vary considerably.
  - What to do?
    - ***Measure this delay and add it to C_{UTC}.***
    - The best estimate of delay is $(T_1 - T_0)/2$.
  - Can subtract off **I** (server interrupt handling time).

Cristian's Clock Synch Algorithm

- Asynchronous System
- What does p set local clock to?
- Client sets its clock to halfway between C_{UTC} and C_{UTC} + Round Trip Time

$$C_{UTC} + T_{round}/2$$

- $T_{round} = T_1 - T_0$

Cristian's Algorithm: Analysis

- Assume
  - min = minimum client-server one-way transmission time
  - the server time stamped the message at the last possible instant before sending it back
- Then, the actual time could be between **$[C_{UTC} + min, C_{UTC} + T_{round} - min]$**
- Error: +/- $[T_{round}/2 - min]$

Cristian's Algorithm

- Problem – the client clock is fast ➔ arriving value of C_{UTC} will be smaller than client's current time, C.
  - What to do?
    - ***One needs to gradually slow down client clock by adding less time per tick.***
- Time must be monotonic.

Leap Seconds and Google

- Fluctuations in Earth's rotational speed mean that even very accurate clocks have to be adjusted slightly to bring them in line with "solar time." There have been 24 such adjustments, called "leap seconds," since they were introduced in 1972.
- Having accurate time is critical to everything we do at Google. Keeping replicas of data up to date, correctly reporting the order of searches and clicks, and determining which data-affecting operation came last....
- Computers traditionally accommodate leap seconds by setting their clock backwards by one second at the very end of the day. But this "repeated" second can be a problem. For example, what happens to write operations that happen during that second? Does email that comes in during that second get stored correctly?
- https://googleblog.blogspot.com/2011/09/time-technology-and-leaping-seconds.html

Google's Smear

Unsmeared UTC	Smeared time
2016-12-31 13:59:59.000000	2016-12-31 13:59:59.000000
2016-12-31 14:00:00.000000	2016-12-31 14:00:00.000000
2016-12-31 14:00:01.000014	2016-12-31 14:00:01.000000
2016-12-31 23:59:58.499972	2016-12-31 23:59:58.000000
2016-12-31 23:59:59.499986	2016-12-31 23:59:59.000000
2016-12-31 23:59:60.000000	2016-12-31 23:59:59.500007
2016-12-31 23:59:60.500000	2017-01-01 00:00:00.000000
2017-01-01 00:00:00.000000	2017-01-01 00:00:00.499993
2017-01-01 00:00:00.500014	2017-01-01 00:00:01.000000
2017-01-01 00:00:01.500028	2017-01-01 00:00:02.000000
2017-01-01 09:59:58.999986	2017-01-01 09:59:59.000000
2017-01-01 10:00:00.000000	2017-01-01 10:00:00.000000
2017-01-01 10:00:01.000000	2017-01-01 10:00:01.000000

For the leap second #37, on December 31, 2016, we used a 20-hour linear smear.

The smear period started at 2016-12-31 14:00:00 UTC. Continued through 2017-01-01 10:00:00 UTC.

Before and after this period, our clocks and time service agreed with servers that apply leap seconds.

https://developers.google.com/time/smear