What’s the Time?

Geoff Huston

APNIC

Background

- All computers run with some kind of internal oscillator (mistakenly called a ‘clock’)
  - This clock manages the internal state changes each cycle of the central processing unit
  - Clock ‘ticks’ are fed to a digital counter
  - From this counter the computer can maintain a conventional clock and maintain the current time

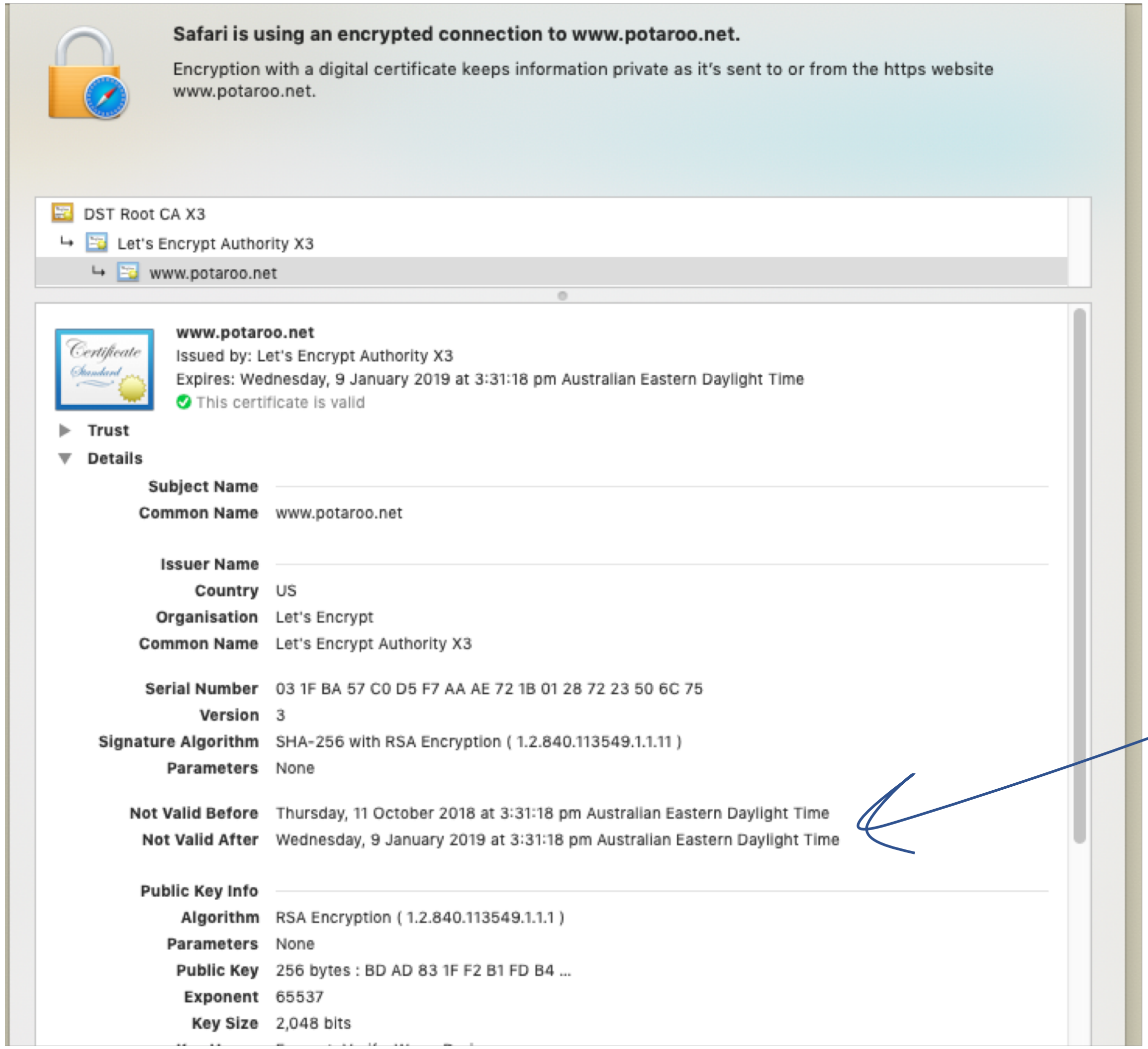
Why is Time useful?

- To stop everything happening at once!

Why is Time useful for a computer?

- To understand when things happen
  - Crontab and event scheduling to ensure that a computer performs certain tasks at precise times
- To understand the relative age of things
  - For example, with NFS file systems its vital to understand which file is more recent
- To understand when things are valid

Security Certificates and Time



These security credentials are only usable in a defined window of time

The computer's local clock is compared to these dates to determine whether to trust this certificate or not

So we need to keep “time”

- But this can be challenging
- Computer clocks are based on quartz Crystal oscillation
  - Quartz crystal oscillation is only stable if the temperature and excitation voltage are kept stable. Changes in temperature or voltage will cause oscillation changes
- Computer time of day clocks rely on counting ticks in a register
  - Which is performed by software running in the processor at an elevated interrupt level
  - If the processor runs for extended times at an even higher interrupt level then clock ticks can be ‘lost’

So we need to keep "time"

- We actually want to keep **accurate** and **stable** time
  - **Accurate** in that every reference timekeeper keeps the same time
    (modulo the spacetime stretch factors of relativity)
  - **Stable** in that the duration of each measured interval is exactly the same
- We need to synchronize the internal computer clock to a reference time

What is reference “time”?

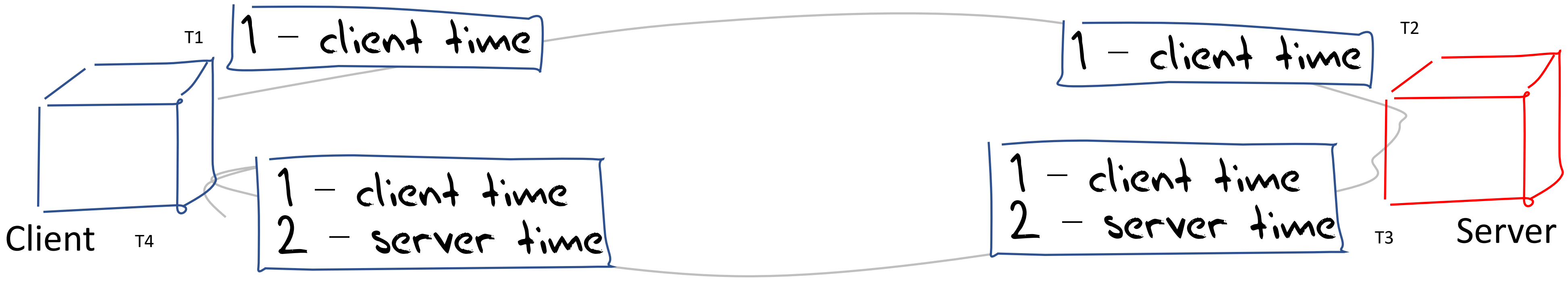
- We all know that time is divided into days, where a ‘day’ is defined as the duration between successive events when the sun is at precisely the same elevation in the sky
  - But we don’t do this any more because the earth and the sun are poor timekeepers
- We turned to distant quasars as the reference point
  - But we don’t do this any more because we needed even greater precision
- We turned to nuclear physics
  - Time is defined using Système International (SI) seconds, defined as the duration of 9,192,631,770 periods of the radiation emitted by a caesium-133 atom in the transition between the two hyperfine levels of its ground state at a temperature of 0K

Distributing Accurate Time

- Not every computer runs their own Cesium Clock or runs a GPS receiver to maintain accurate time
  - But some folk do
- So what we would like is a way to take this set of highly accurate reference time sources and provide a mechanism for others to synchronize their local clock against a reference source
- On the Internet we use the Network Time Protocol (NTP) to perform this time synchronization function

NTP Operation

- Time sources are classified by their accuracy
  - A Stratum 0 server is a reference clock (GPS or cesium)
  - A Stratum 1 server is directly connected to a reference clock source
  - A Stratum 2 server receives its time from a Stratum 1 server, and so on
- NTP is a simple clock exchange UDP protocol



Client Offset = ½ ((T2-T1) + (T3-T4))

NTP Operation

- In steady state the UDP clock packet exchange happens every 16 seconds
  - Faster clock exchanges happen when the client clock has lost synchronisation with the server, and it will burst 8 packets evenly spaced across a 16 second interval
- If the local clock needs to be adjusted the client time application will use adjtime() to slew the local clock. Clock correction is slow – 0.5ms per second
  - Jumping the clock can fatally confuse applications, so this gentle slew is far kinder
- NTP can normally maintain a client clock within a few hundredths of second of the server reference clock

So we all agree on the time?

- If everything supports NTP, and there is a well structured mesh of NTP reference clock servers then every connected Internet device that runs a clock should have the same value of time
  - “same” is within a tenth of a second or less

- But does the Internet agree on the time?

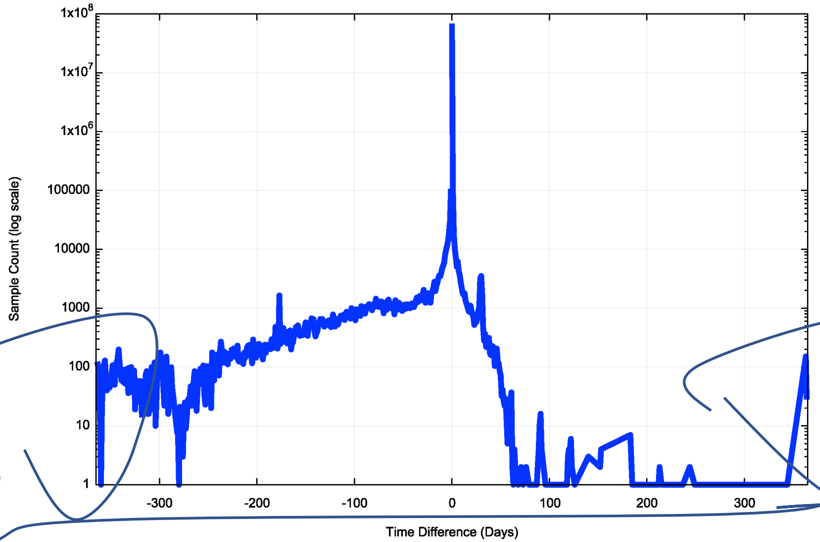
The Experiment

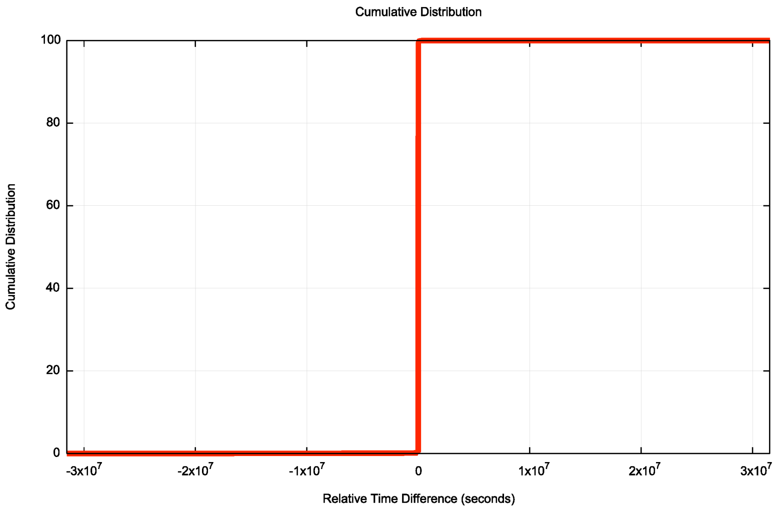
- Use a scripted online ad to direct a client report back on the time of day on the client system
  - Use the Javascript getTime() method to get the local UTC clock value
  - Pass this value to the server as an argument to a URL fetch operration
- Use NTP-managed clock on the server to maintain a stable reference clock
- Record the distribution of differences
  - Ignore the fine-grained differences due to local processing and network propagation time
  - Which means that we are looking at measurements of time within +/- 1 second as being equivalent

Results

We tested the clock of 202,460,921 clients over a 80 day period:

- 11% of clocks are more than 1 second fast
- 57% of clients are more than 1 second slow
- We observed clock slew values of up to 1 year both fast and slow
- 92% of clients are within 120 seconds of the reference clock





A view of Whole of Internet Time

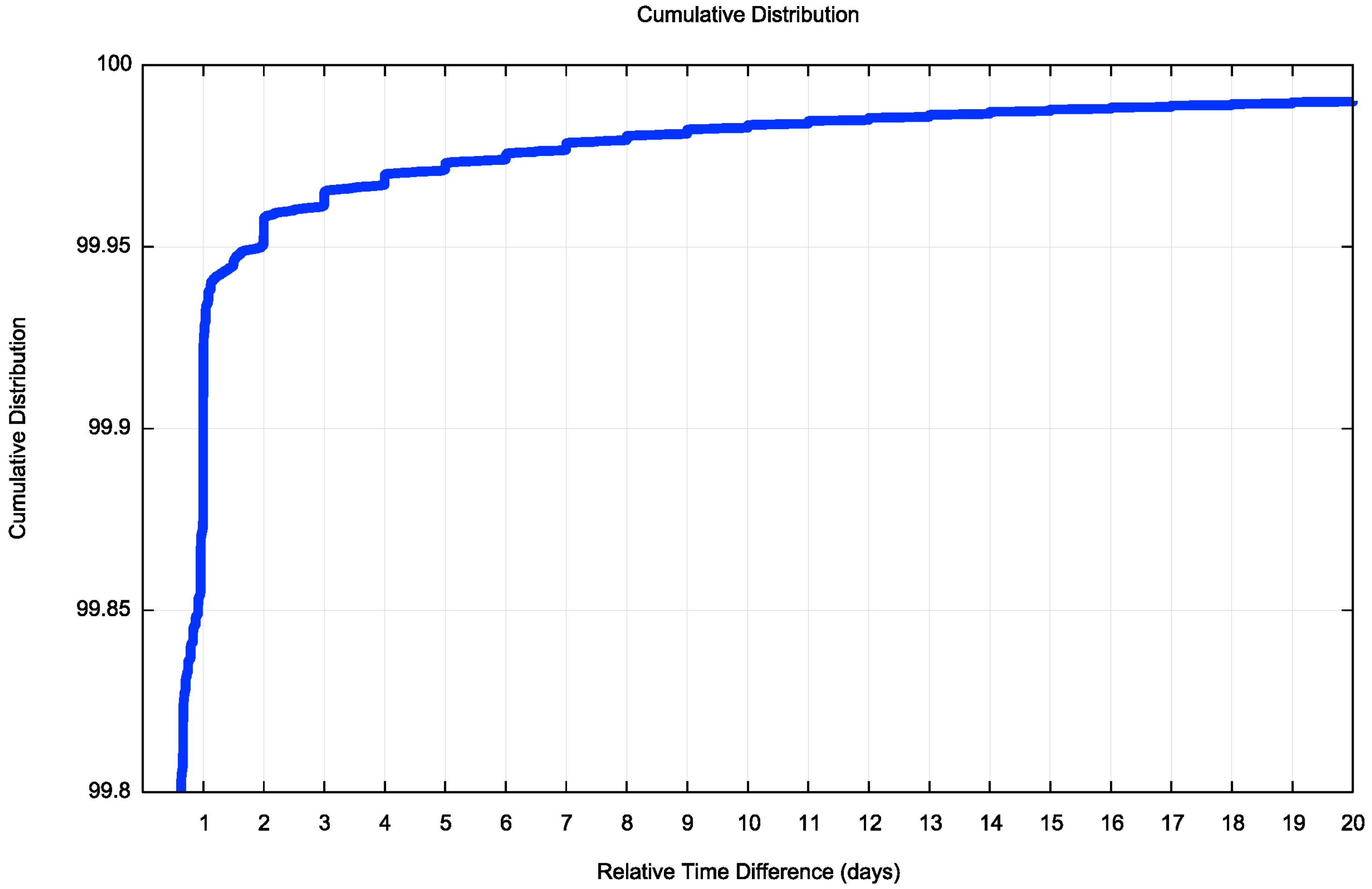
- Only 58% of visible clients run their clock with 2 seconds of UTC time
- 92% of visible clients run a clock that is within 60 seconds of UTC time
- 98% of clients are within 1 hour of UTC time

Fast Clocks

0.05% of all clocks are ahead by more than 2 days

There is a clear step function in this distribution that is aligned quite precisely to whole days

How can a client clock maintain a stable per-second clock, yet report a time value that is off by a number of whole days?

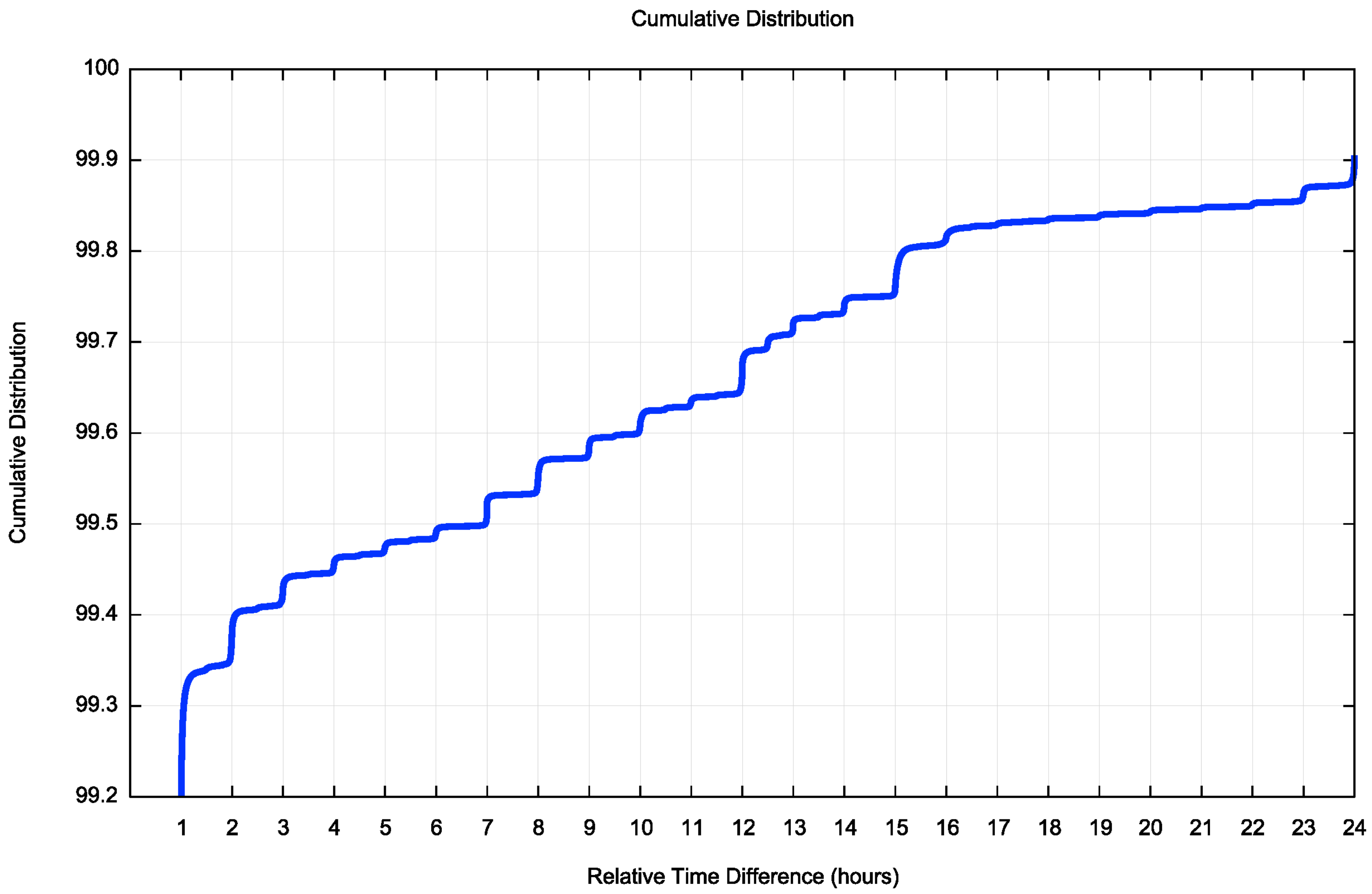


Fast Clocks

0.7% of all clocks are ahead by more than 1 hour

As with the day distribution, there is a marked clustering of the clock offsets into units of hours, and a slightly smaller clustering into half-hours

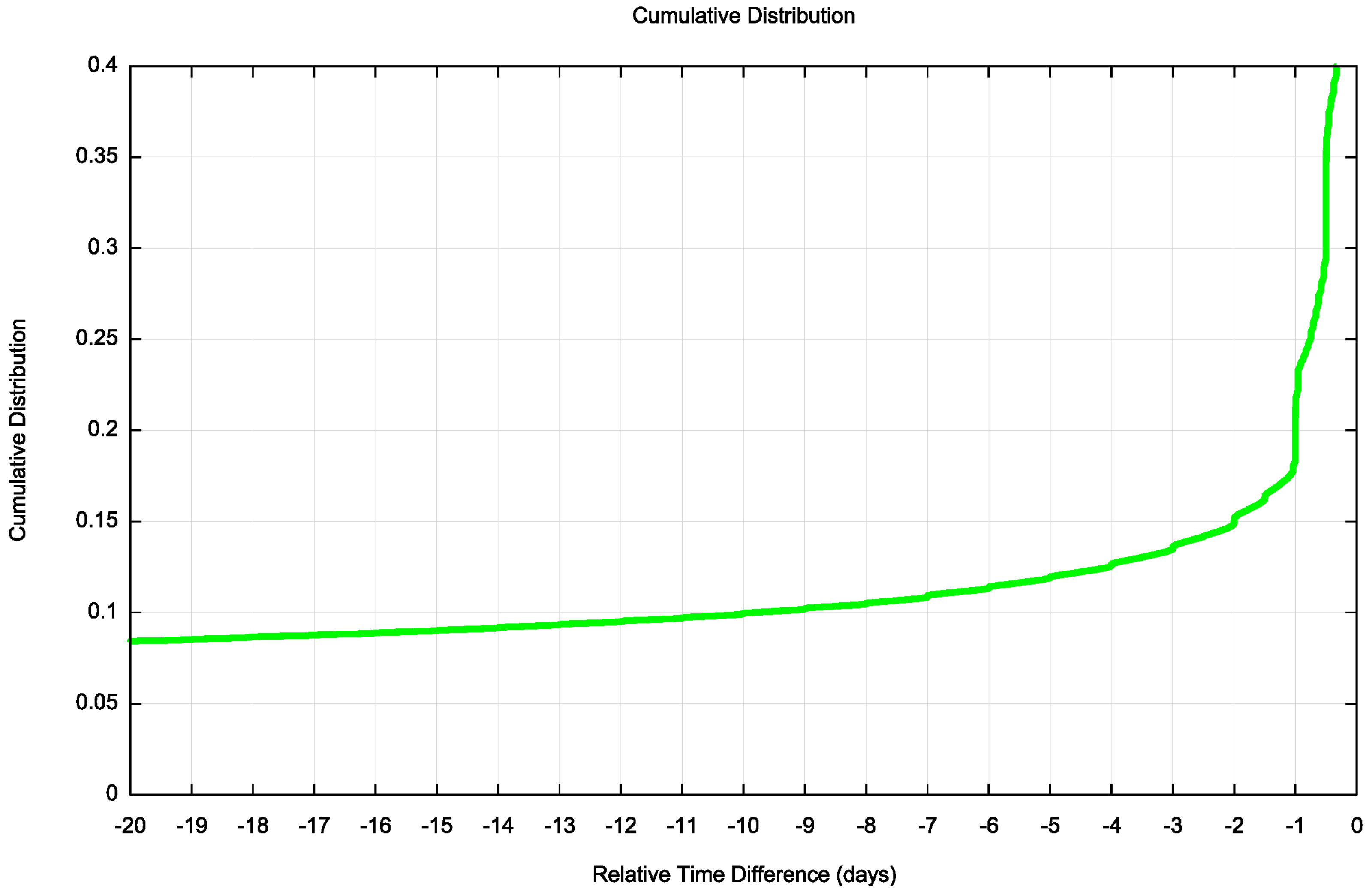
Similar question: How can a client clock maintain a stable per-second clock, yet report a time value that is off by a number of whole hours?



Slow Clocks

0.15% of all clocks lag by more than 2 days (3 x the number of fast clocks)

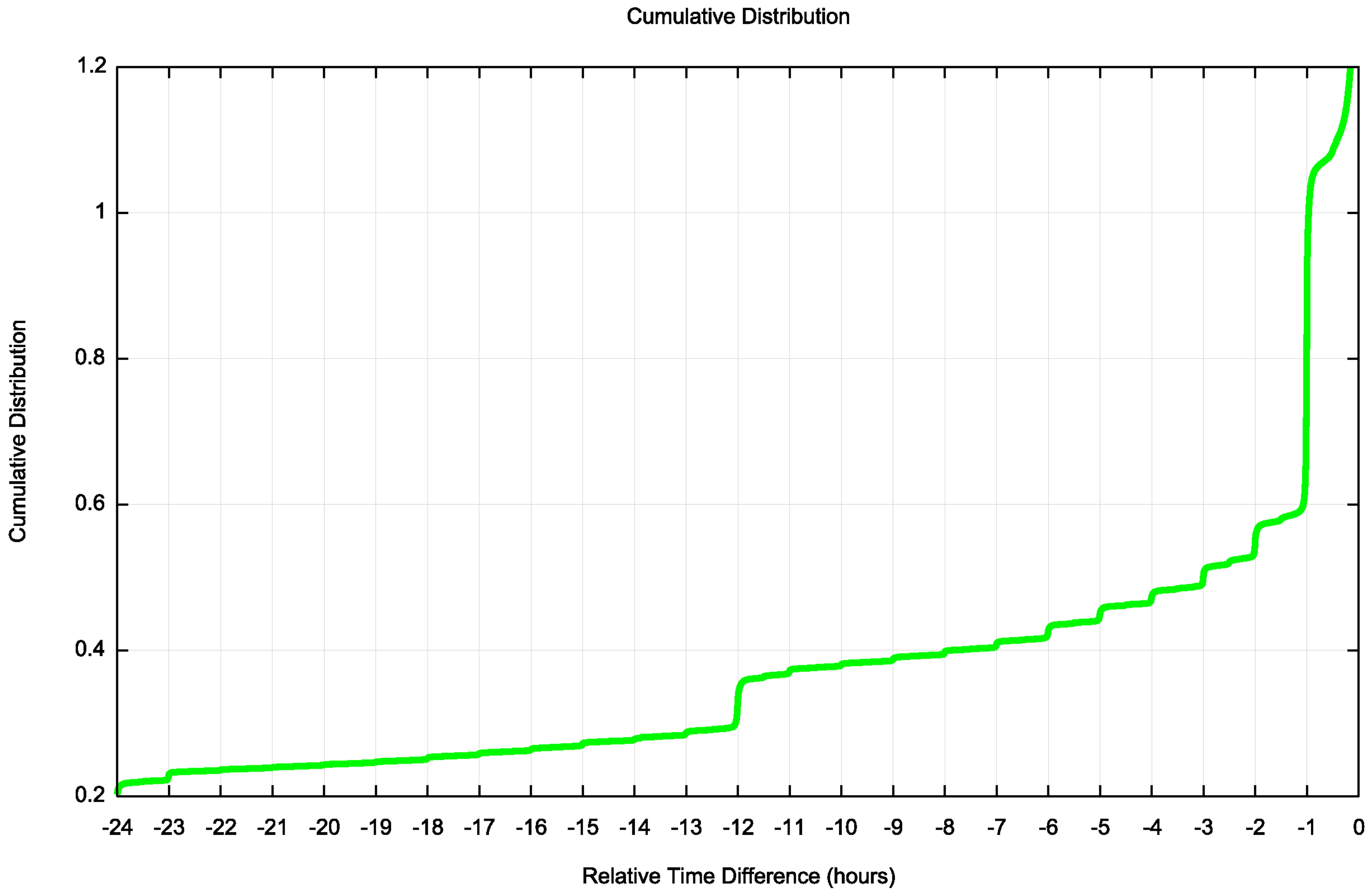
The per-day clustering is not so clear for slow clocks with a lag of greater than 2 days.



Slow Clocks

1.05% of all clocks lag by 1 hour or more

Here there is a marked clustering of the clock offsets into units of hours



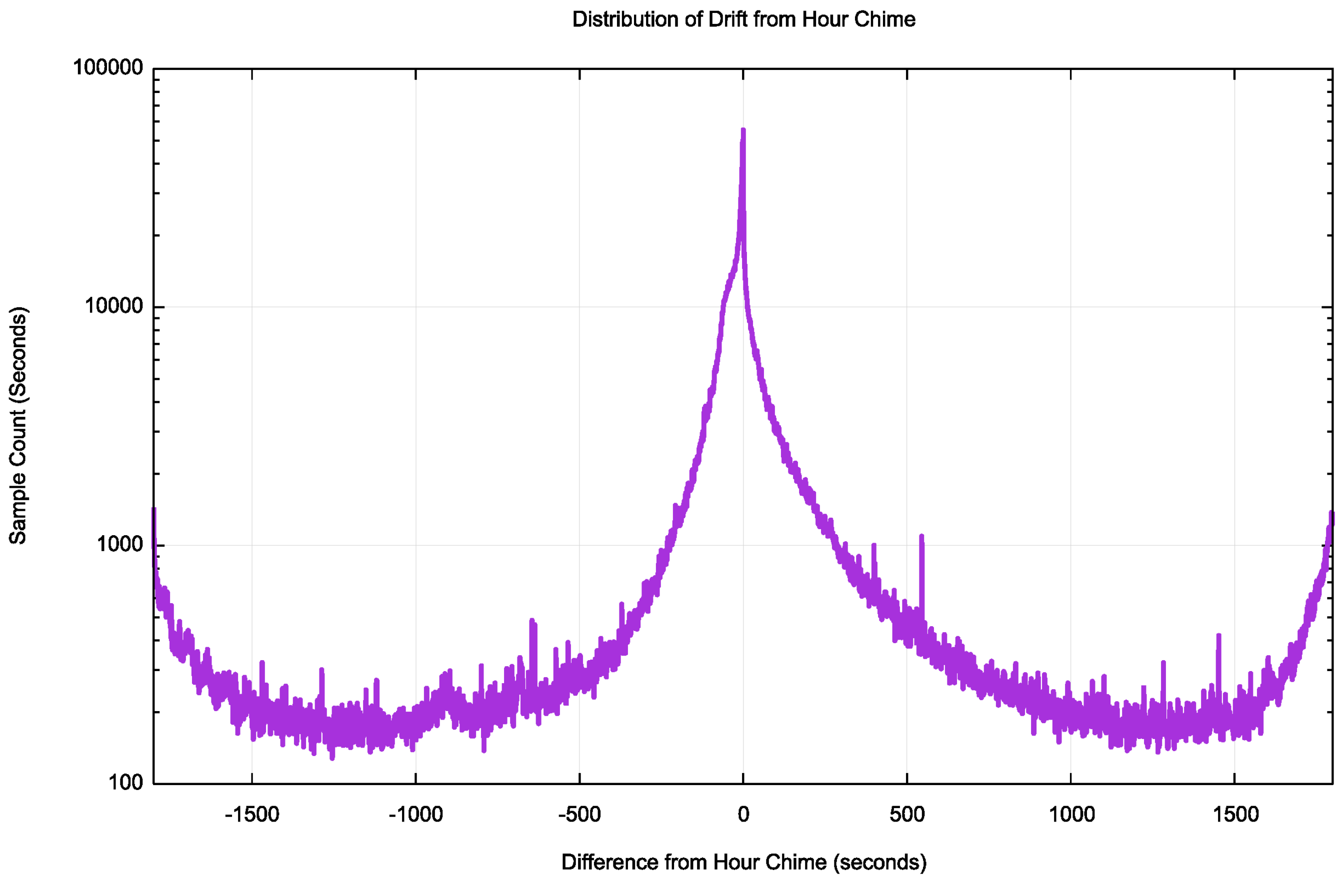
Clustering of Clock Slew Values

This is a distribution of the clock slew values when the whole hours are removed

There is a very strong signal that when a clock has slewed from UTC time it does so in units of hours (and less so in units of half-hours)

NTP does not stablilze a local clock into a slew value of a whole number of hours, so this distribution is not an artefact of NTP.

What is going on here?



Clustering of Clock Slew Values

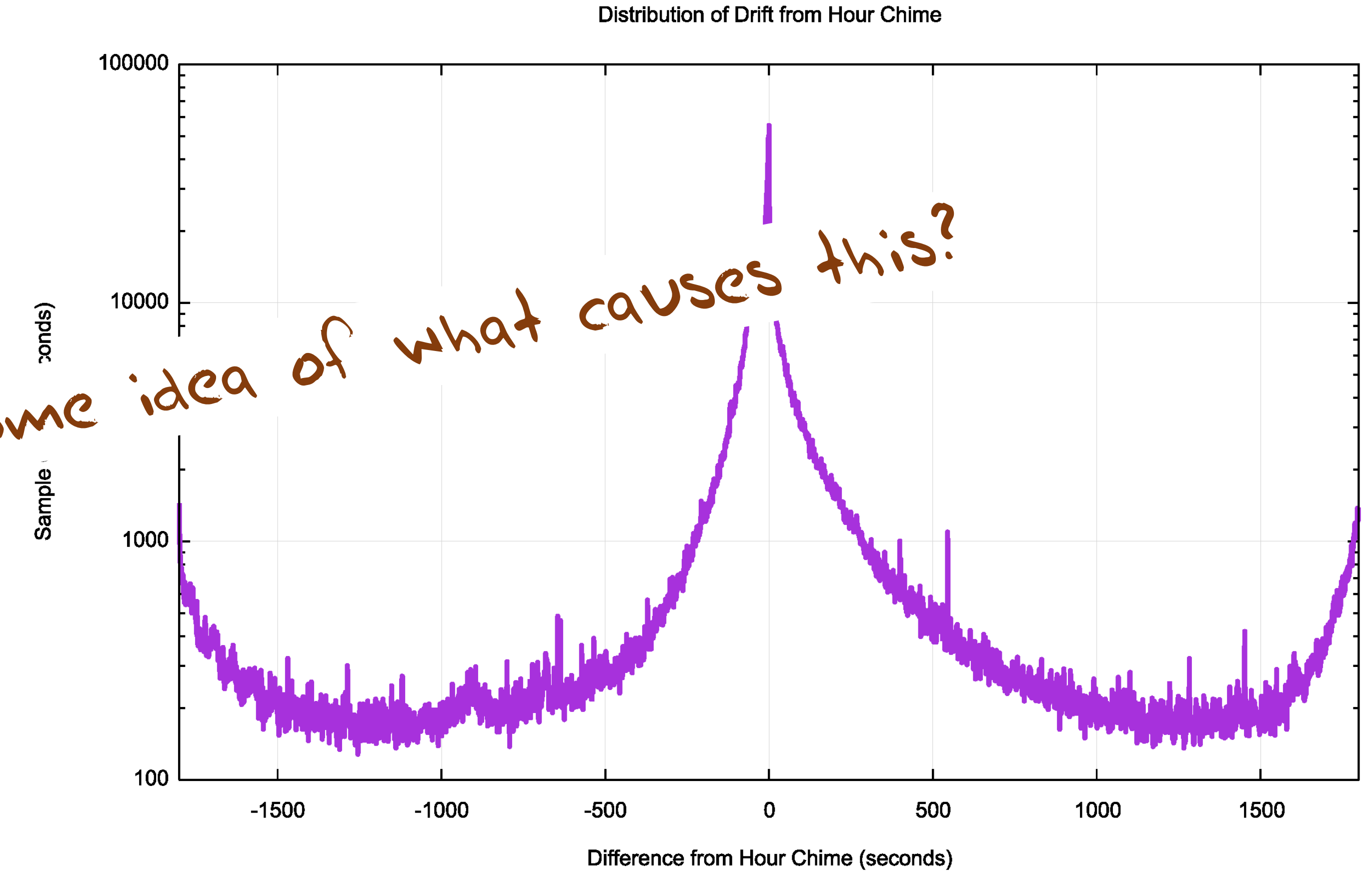
This is a distribution of the clock slew values when the whole hours are removed

There is a very strong signal that when a clock has slewed from UTC time it does so in units of hours (and less so in units of half-hours)

NTP does not stablilze a loc… value of a wh… so this dist… artefact of NTP.

What is going on here?

Does anyone here have some idea of what causes this?



Thanks!