Networks 21: NTP

i.g.batten@bham.ac.uk

NTP

- Network Time Protocol
- Often used, rarely understood
- Clock alignment is necessary for log integrity, crypto replay prevention, filesystem caching
- It’s also vital if you want to do performance measurement on networks without assumptions about symmetric RTT.
- Time is hard. Knowing why it’s hard is good.

NTP is not about consensus

- NTP does not operate to average or take a central estimate of an ensemble of clocks

- NTP is about choosing exactly one clock and synchronising the local clock to that single remote reference

- Other clocks are used to cross-check and provide back-ups

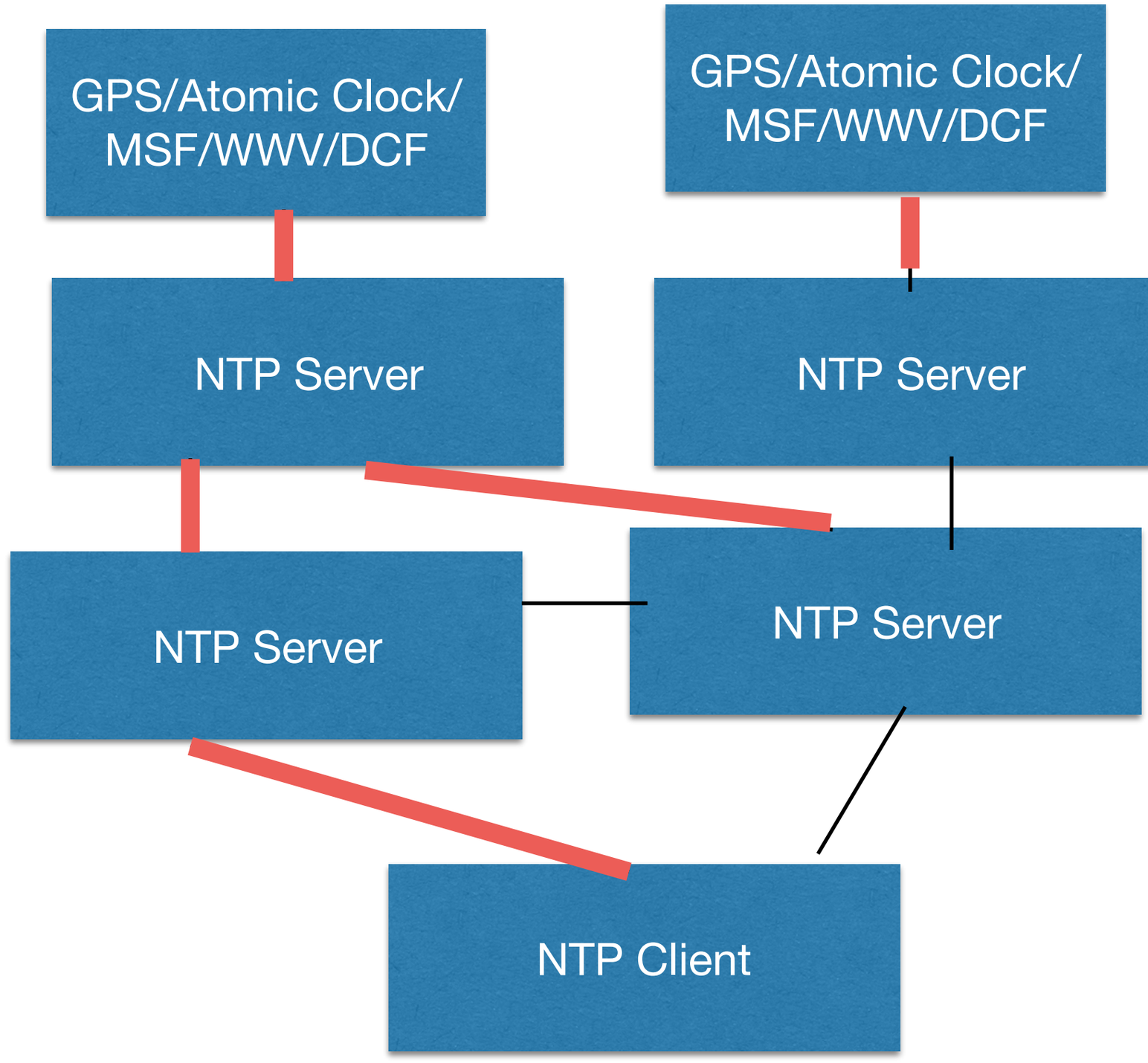
Stratums/Strata

Stratum 0

Stratum 1

Stratum 2

Stratum 3



Definitions

- **Rate**: the rate of a clock is the frequency of some repeating event, such as a rising edge

- So getting a clock correct for rate is about making each 1s tick equal (usually) 1 SI second. This can be done with a pendulum or a crystal or a tuning fork without external references.

- **Phase**: the phase of a clock is the alignment of it with some external timescale, usually UTC.

- So a clock which is “correct” in informal terms is correct in rate and phase. A clock which is correct in rate but wrong in phase will be consistently misaligned. A clock which is wrong in rate but initially correct in phase will drift away from correct time (linearly, quadratically, randomly).

- **Drift**: the error in rate for a given clock, which is hoped to be constant (but isn’t).

Computer Timekeeping

- It's rubbish. Next slide.

- When power is off, clock maintained by a "TOD Clock", which is vaguely accurate (<1s per day).

- When power is on, one of the onboard oscillators provides a clock reference for the computer, which periodically causes a counter to be incremented.

- Errors arise because that oscillator is inaccurate, not thermally stabilised, and sometimes the interrupt (or equivalent) gets dropped under load.

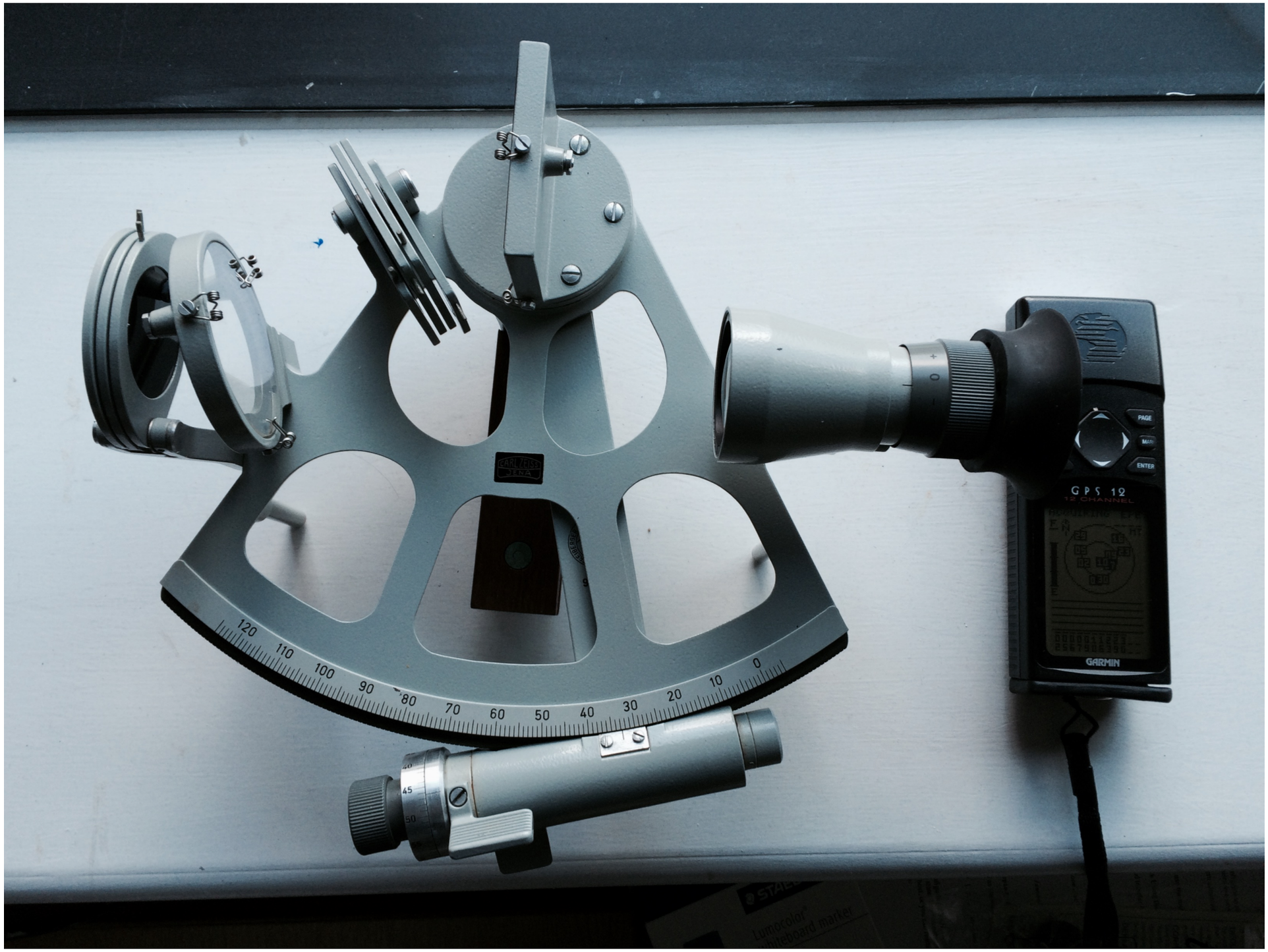
Old clocks can be better!

- If you have an accurate clock and can measure the angle between the horizon and the sun then, with some simple calculations, you can measure your longitude (how far east or west you are).
- Harrison's chronometers were temperature compensated (amongst other things)
- Computer clocks aren't compensated for anything.

Purpose of Clocks

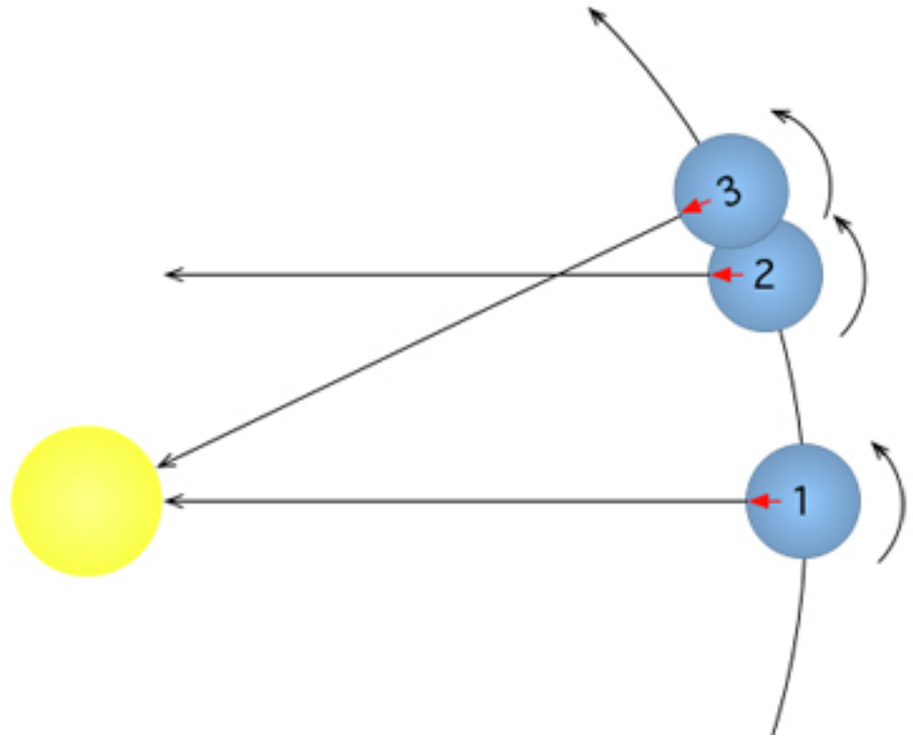
- Constant seconds?
- Labelling successive seconds uniquely?
- Making sure sun is at its zenith at 1200?
- Making sure winter is always in December?
- What?

Modern Sextant and early 1990s GPS receiver

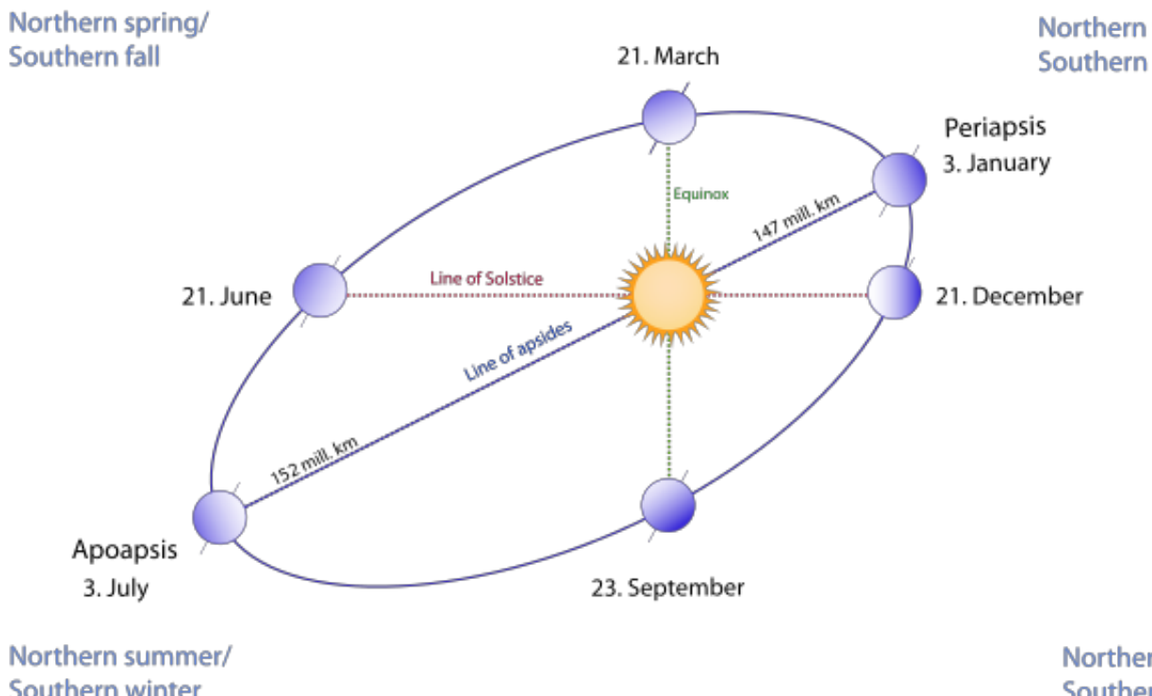


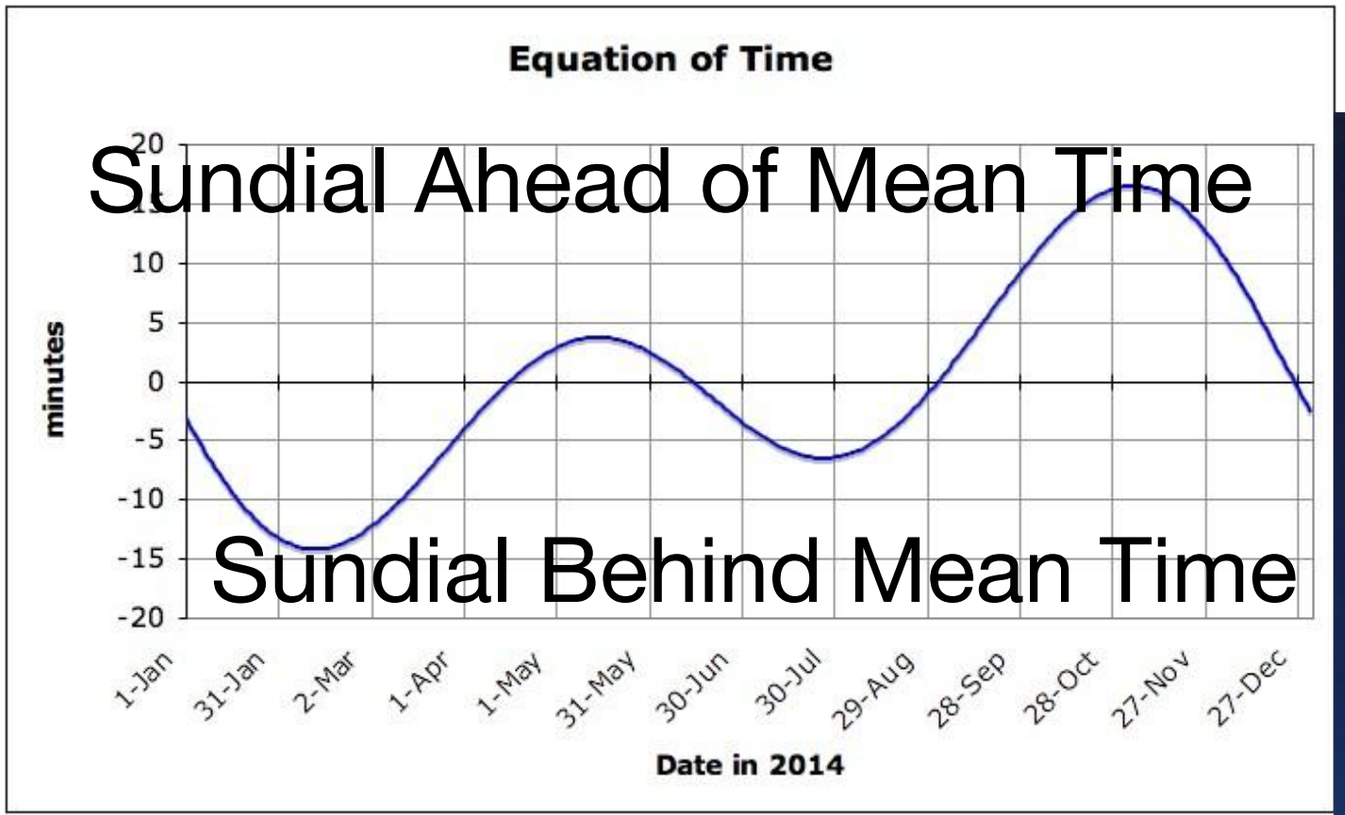
Solar Time v Sidereal Time

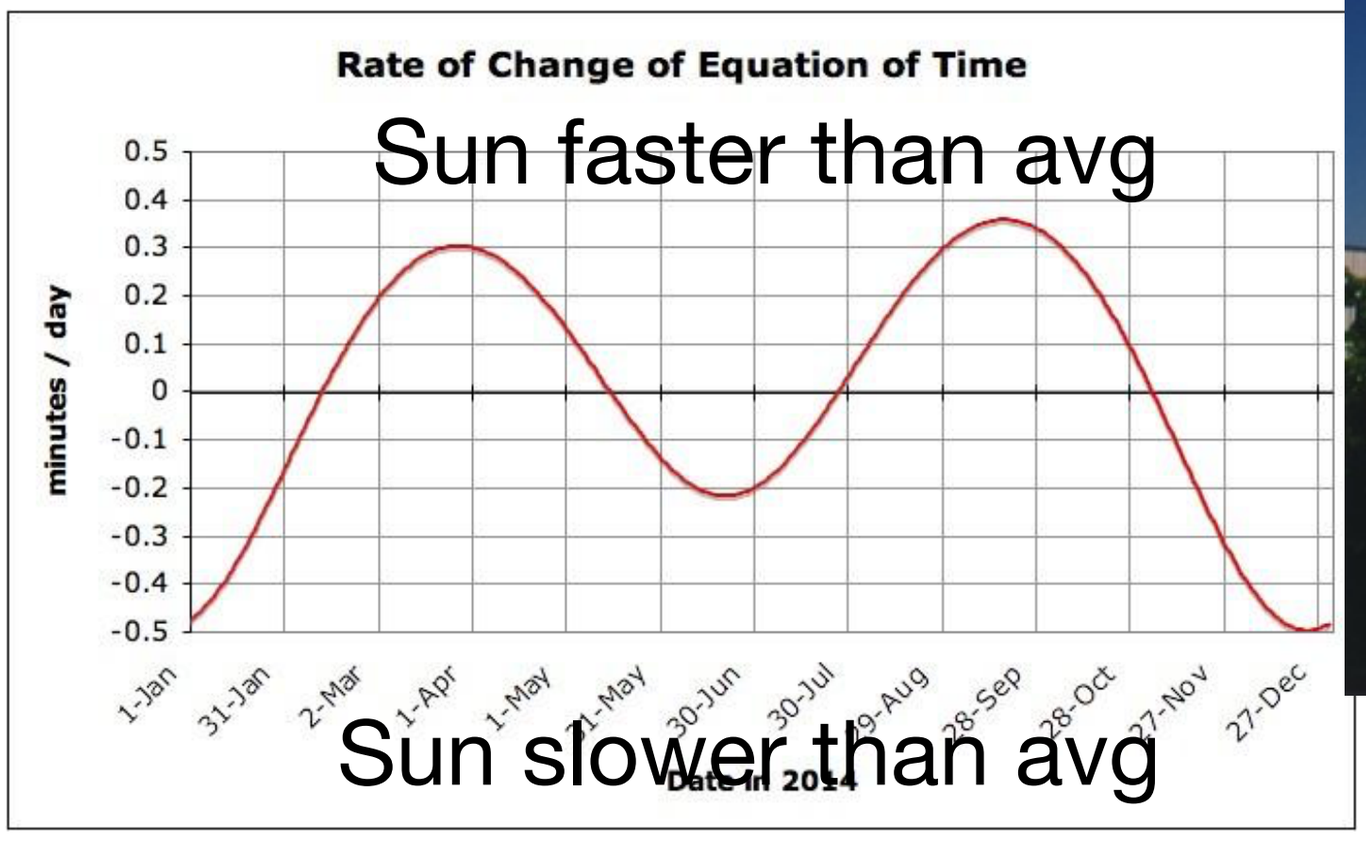
Earth spins on axis in 23h56m, relative to stars. Then it takes on average another four minutes for sun to be overhead, as earth has moved.



Because the earth's orbit is not circular, this effect is not consistent. So sometimes the sun will be overhead slightly earlier, sometimes slightly later.




Equation of Time
Sundial Ahead of Mean Time
Sundial Behind Mean Time
minutes
20
10
5
0
-5
-10
-15
-20
1-Jan
31-Jan
2-Mar
1-Apr
1-May
31-May
30-Jun
30-Jul
29-Aug
28-Sep
28-Oct
27-Nov
27-Dec
Date in 2014


Rate of Change of Equation of Time
Sun faster than avg
Sun slower than avg
minutes / day
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
1-Jan
31-Jan
2-Mar
1-Apr
1-May
31-May
30-Jun
30-Jul
29-Aug
28-Sep
28-Oct
27-Nov
27-Dec
Date in 2014

Effect is not small

- The length of a solar day (from the sun at its highest point on two successive days, as seen from a fixed location) varies by +/- 7.9s over the course of a year from the average.

- If you try to use seconds as “noon to noon, divided by 86400” then the second is shortest in June and December, longest in March and September. The difference between the two is about a minute per week, which would be a pretty rubbish watch.

Leap Years

- Earth rotates around Sun in slightly less than 365.25 days
- 365 day calendar will see spring equinox (say) happening a day earlier every four years.
- Leap Years allow for the .25 days and keep calendar roughly aligned with seasons in the short term (“Julian Calendar”), but now spring equinox happens a day later every hundred years.
    - Spring equinox matters in the west as it relates to Easter, Passover and earlier pagan festivals.
- Skipping leap years that are divisible by 100 corrects by a day a century, but will now lose by one day every 400 years.
    - So years divisible by 400 **are** leap years (much confusion as 1900 and 2100 are not leap years, 2000 was).
- Sorting this out in the 17th and 18th century was pretty messy (“Gregorian Calendar”).

Gregorian Shift

```
ians-macbook-air:~ igb$ cal 9 1752
   September 1752
Su Mo Tu We Th Fr Sa
          1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

England (and this being pre-1776, America and other Colonies) switched in 1752 due to *Calendar (New Style) Act 1750*. But it had already happened in 1582 in Catholic Europe.

The 1750 act also explains why our tax year starts on April 6th. Because 11 days were “lost” in 1752, the tax year 1752–3 was counted as 365 days from the 25th of March 1752 to 4th April 1753, with 1753–4 starting on 5th April 1753. 25th of March “Lady Day” was the historic start of the financial year and farm rentals. The same thing was done to deal with the “missing” 29th of February in 1800, moving the financial year to start on 6th April 1800. This was not done in 1900, hence our 6th April tax year start is the historic Lady Day plus some, but not all, of the Gregorian days.

Mean Time

- Clock ticks constant seconds

- On average (over a year or several years), sun is directly overhead at noon.

- It turns out that we can measure time more accurately than the earth keeps it, particularly rotation (slowed by frictional losses, nudged by earthquakes)

- Days much shorter in the past: 200 million years ago in the Jurassic, probably 23 hour days and 385 day years.

SI Second

- 9 192 631 770 transitions of Caesium 133.
- Based on movements of the earth as measured during, and with the technology of, the early 20th century, incorporating observations from the late 19th.
- It turns out to be slightly too short: if you count 86400 of them per day for a few years, the sun doesn’t quite return to the same place overhead.
- Solutions are messy, arguments are long.

Handwave over Timescales

- **TAI**: atomic clocks ticking SI seconds, with no claim to physical reality.
- **UT1**: Intervals between successive noons are averaged along a meridian and over a year and divided into 86400 seconds. These seconds aren't SI, and worse aren't constant (earth rotation is not constant in short or long term). 86400 * SI-second < mean day, and getting worse.
- **UTC**: TAI, with leap seconds added (or in principle subtracted) to keep it within 0.9s of UT1 ($|DUT1| < 0.9$). UTC is "good enough" for astro-navigation.
- **GMT**: UK legal time, probably UT1. In reality, UK time is UTC, which is transmitted by MSF and the "Greenwich" pips on the radio. DUT1 is available if you need it. US legal time is UTC. For other countries it's not quite clear, hence the debate over leapseconds.
- NTP is about distributing UTC, but other time scales (such as TAI, UTC(GPS) are used as references).

Leap Seconds

- Add extra seconds at 23:59:60 UTC to allow for the preceding seconds being slightly too short
- Allows clocks to tick constant seconds, but be sharply corrected at a fixed time.
  - Provision also for double leap seconds (23:59:61) and skipping 23:59:59: completely untested in both cases
- Happens at midnight 30 June or 31 December, London Time
  - June leaps not funny for far-East stock markets
  - Whole thing complex and messy for manufacturing, network management…
- Posix/Unix/computer time knows nothing of leap seconds, seeing them as corrections to be make, nothing more
- Serious proposals to abandon leap seconds for this reason, leaving issue to changing time zones in a few thousand years' time.
- Google Google's "leap second smear" for one implementation.

NTP Objectives

- Correct rate: compute the instantaneous drift which can be applied to the computer clock to make it tick at the same rate as the reference clock

- Correct phase: apply corrects to the system clock to make its phase align with the reference clock

- Monotonicity: the computer clock shouldn’t step sharply forward, and should never go backwards. If the clock is “fast”, you run it slow until reality catches up.

GPS is King

- These days, most accurate source of time is GPS, which flies atomic clocks which transmit their time (handwave)
- Ground segment knows where the space segment satellites are, observes the transmitted time, calculates the range and solves for position
- At speed of light, 1’ (1 foot, 30cm) is roughly 1 nanosecond
- GPS has to deliver ~10ns precision timing to ground-based equipment in order to produce <1m ranging accuracy.
- In passing, GPS proves both special relativity and general relativity
  - GPS clocks fall behind ground clocks by 7 microseconds per day because moving faster than us
  - GPS clocks gain on ground clocks by 45 microseconds per day because in lower gravitational potential
  - Together, 38 microseconds per day means errors building by 10km per day unless corrected for.

Alternatives to GPS

- Although the distinctions are legal rather than practical, you might want to also need to align a national standard
  - MSF (“Rugby” although now actually in Cumbria)
  - DCF77 (“Braunschweig” although actually in Mainflingen)
  - WWV (Colorado)
- Actually getting accurate timing off these (ie, better than a few milliseconds) is quite tricky, and involves PLLs locked to the carrier. You can’t do it just by looking at the pulses.

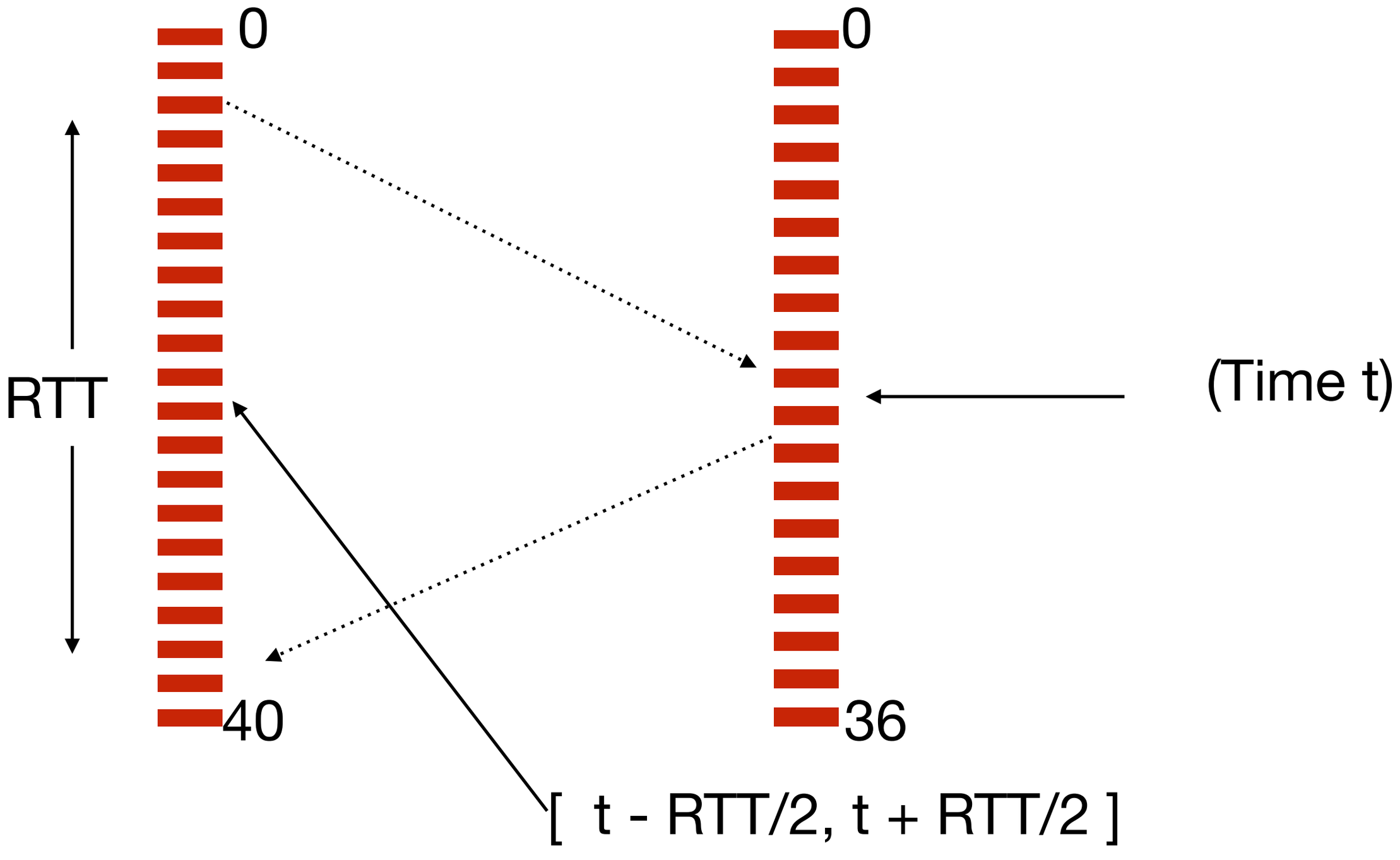
OS issues

- Multics really, really, needed timestamps to be unique and monotonic, as the current time (microseconds since 00:00 01/01/1901) was used for purposes like naming process directories where the stack lived.

- Unix isn't quite as fussy, but timestamps going backwards will potentially cause problems (deep assumption that seconds || pid is unique) and timestamps going forward with large gaps can break *cron(1)*, *at(1)* and *select(2)*.

- Thought experiment: if I set a 60s timeout on an operation, and after 1s the clock steps by 60s, what should happen?

Basic Operation: Phase



NTP Clock Selection

- NTP has a complex algorithm to select clocks
- It prefers clocks that are higher (numerically lower) stratum, close (low RTT) and stable (low variation in RTT).
- It finds the intersection of a set of clocks to exclude outliers (“false tickers”)
- But once it has found a clock it trusts, it aligns to that, not to the intersection

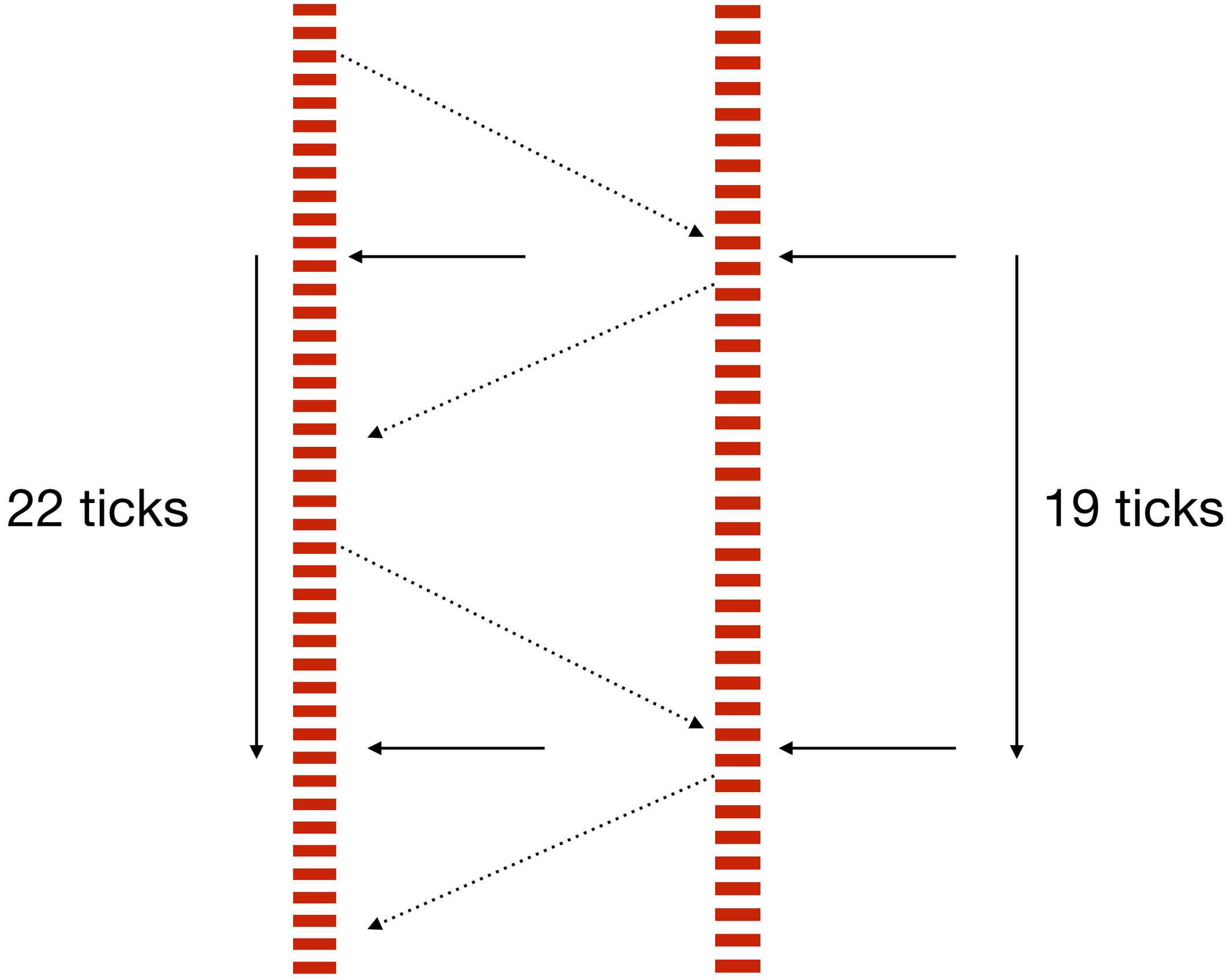
SNTP v NTP

- SNTP (“Simple” NTP) doesn’t try to condition the local clock, it just resets the system clock every time it talks to a reference server. In this case, there will be savage drift between adjustments.

- Full NTP computes the drift of the local clock and attempts to correct the rate as well.

Basic Operation: Rate

- Get the time (phase) twice
- You know the interval between the time at the reference clock, because you have two successive timestamps from it
- You know the interval in terms of your local clock, because you can count ticks
- You can therefore compute the drift between the two clocks, as the ratio between their rates (in our case the local clock is ticking 22s for every 19s on the reference clock)


22 ticks
19 ticks

Dispersion / Jitter

- Obviously, all these computations have error bars based on things like asymmetric network paths, interrupt latency, local clock drift, etc.

- Dispersion is a measure of how likely it is that the time is accurate (error bars on result)

- Jitter is a measure of the rate at which the computed time changes

Drift

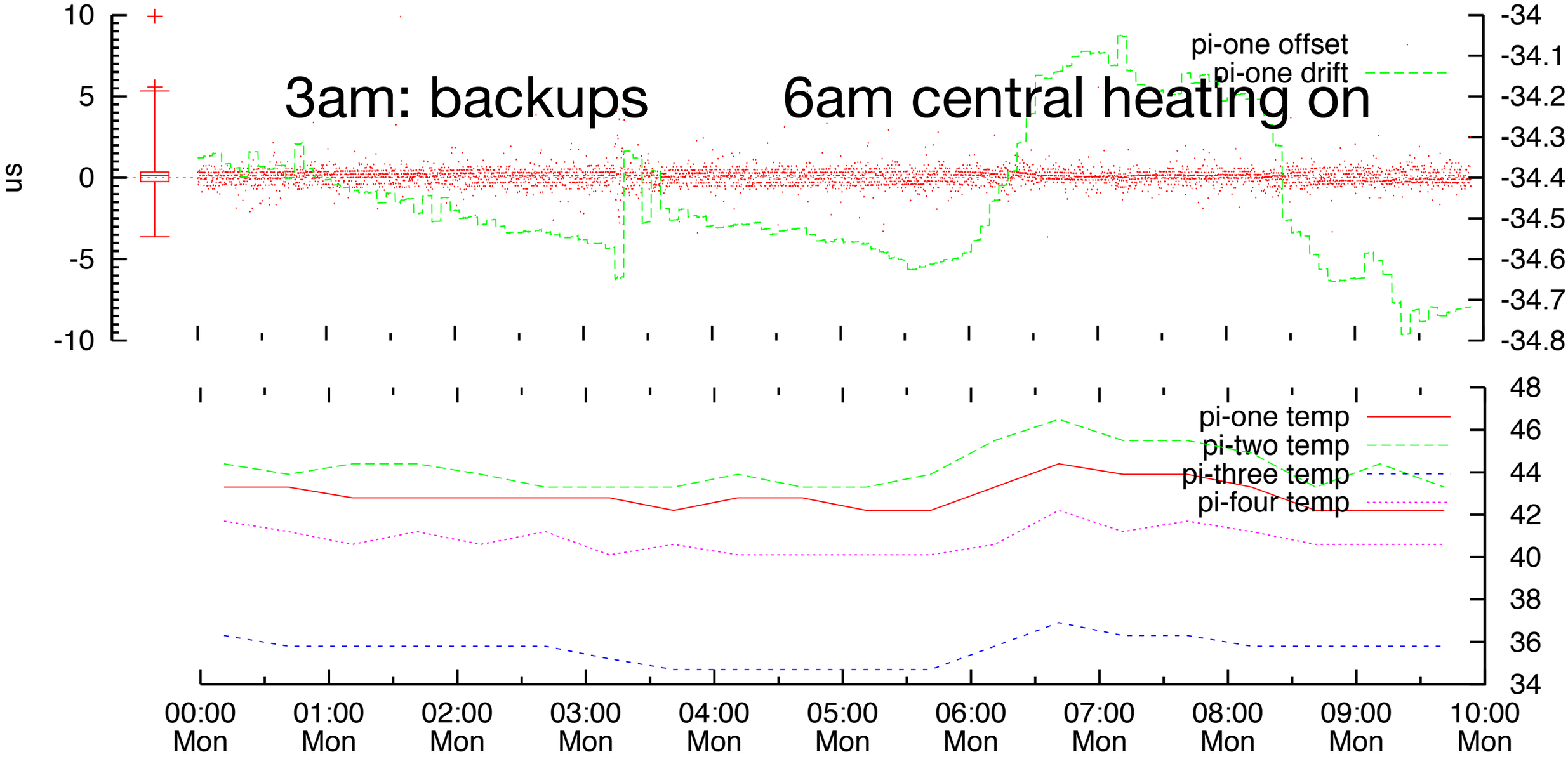
- Once you know the ratio between your local clock and the reference clock (typically of the order of 50ppm) you can add slightly more or less to the time each time there is a tick.
  - 50HZ tick = 20ms per tick, 20.001ms per tick if local clock is 50ppm slow, 19.999ms per tick if local clock is 50ppm fast
- Obviously requires timestamps in higher precision than previously

Drift Stability

- If drift is constant, you can predict the behaviour of the clock over the long term and keep accurate time without an external reference (cf. naval chronometers, where consistency is more important than absolute accuracy).

- For a lightly loaded machine in a thermally stable room, drift can be consistent to <0.1ppm (<3s per year).

Real World Performance



Offset between a GPS “pulse per second” output and system clock.

Oscillator Stability

- So, here a 2C change in system temperature equates to 0.5ppm change in system clock.

- 1ppm is nearly 1 second per fortnight: you can buy better clocks than that in petrol stations and military surplus places.

  - F91W: the bombmaker’s favourite, allegedly

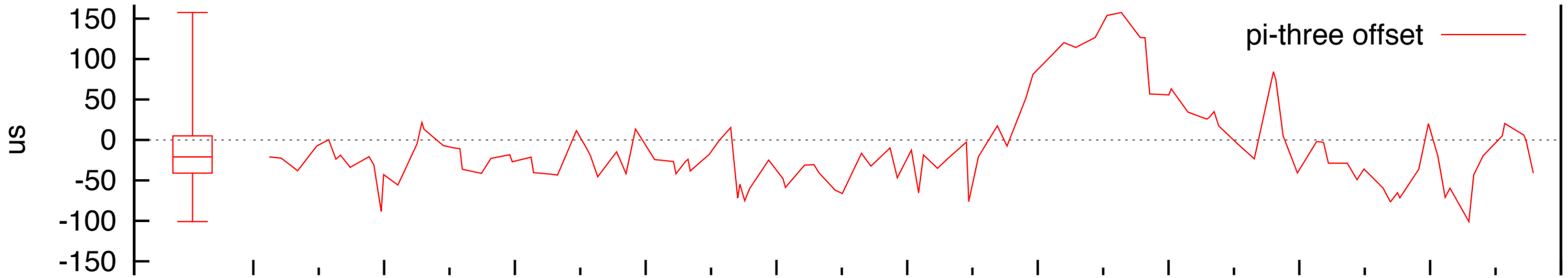
  - G10: UK military issue, plus me and (more recently) Nick Hawes.

System Load

- A smaller but sharper additional drift is imposed by system load

- This is exacerbated by the device being a Raspberry Pi, which has the ethernet on a USB port and a rather crude disk controller.

Nearby machine (ethernet)



Central heating, again, both
on local oscillator and remote time

Behaviour on Internet

seconds milliseconds

```
igb@research-1:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ntp0.linx.net   .PPS.           1 u  682 1024  377    7.226    0.019   0.498
+ntp1.linx.net   .PPS.           1 u   32 1024  357    6.714   -0.441   2.120
+ntp2.linx.net   .GPS.           1 u  660 1024  377    6.326   -0.297   0.351
igb@research-1:~$
```

```
igb@research-1:~$ ntpq -c rv
associd=0 status=0619 leap_none, sync_ntp, 1 event, leap_armed,
version="ntpd 4.2.7p381@1.2483-o Tue Jun 17 19:07:01 UTC 2014 (1)",
processor="i86pc", system="SunOS/5.11", leap=00, stratum=2,
precision=-21, rootdelay=7.226, rootdisp=34.036, refid=195.66.241.2,
reftime=d8ba6c45.dde4d76a  Mon, Mar 23 2015 10:41:09.866,
clock=d8ba6eeb.359c0f67  Mon, Mar 23 2015 10:52:27.209, peer=27373,
tc=10, mintc=3, offset=0.090644, frequency=-64.524, sys_jitter=0.310031,
clk_jitter=0.372, clk_wander=0.001, tai=35, leapsec=201507010000,
expire=201512280000
```

Behaviour on LAN

```
igb@pi-three:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ff05::101       .ACST.           16 u    -   64    0    0.000    0.000   0.001
*pi-one.home.bat .PPS.            1 u   42   64  377    0.823   -0.060   0.067
-ntp2.warwicknet 195.66.241.10    2 u   18   64  377   13.825    0.486   0.345
```

```
igb@pi-three:~$ ntpq -c rv
associd=0 status=0615 leap_none, sync_ntp, 1 event, clock_sync,
version="ntpd 4.2.6p5@1.2349-o Mon Feb  9 03:34:42 UTC 2015 (1)",
processor="armv7l", system="Linux/3.18.7-v7+", leap=00, stratum=2,
precision=-20, rootdelay=0.823, rootdisp=5.620, refid=241.95.198.250,
reftime=d8ba6f42.79dc3c24  Mon, Mar 23 2015 10:53:54.476,
clock=d8ba6faa.b657ab3d  Mon, Mar 23 2015 10:55:38.712, peer=19693, tc=6,
mintc=3, offset=-0.035, frequency=-5.314, sys_jitter=0.048,
clk_jitter=0.032, clk_wander=0.010
```

Behaviour on WLAN

```
ians-macbook-air:~ igb$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 usnyc3-ntp-002. .STEP.          16 u 195m 1024    0    0.000    0.000   0.001
 ntp-anycast     .ACST.          16 u    -   64    0    0.000    0.000   0.001
*kvm1.websters-c 193.62.22.82     2 u   85  512  377    7.893  -77.799  66.956
+eggburt.positiv 178.79.160.57    3 u  186  512  377   24.768  -111.82  79.432
+ntp1.exa-networ 33.117.170.50    2 u  131  512  377    5.144  -111.54  79.444
-dns2-ha.uk.syra 45.162.145.187   3 u  439  512   77    9.024  -166.21 126.933
ians-macbook-air:~ igb$
```

```
ians-macbook-air:~ igb$ ntpq -c rv
associd=0 status=0618 leap_none, sync_ntp, 1 event, no_sys_peer,
version="ntpd 4.2.6@1.2089-o Fri May 28 01:20:53 UTC 2010 (1)",
processor="x86_64", system="Darwin/14.3.0", leap=00, stratum=3,
precision=-20, rootdelay=21.810, rootdisp=227.604, refid=176.126.242.239,
reftime=d8ba6de3.8408e74e  Mon, Mar 23 2015 10:48:03.515,
clock=d8ba72b4.fd970317  Mon, Mar 23 2015 11:08:36.990, peer=58963, tc=9,
mintc=3, offset=-84.423, frequency=8.617, sys_jitter=68.255,
clk_jitter=27.933, clk_wander=0.057
ians-macbook-air:~ igb$
```

Networking Issues

- Likes fast, consistent routes
- On a LAN, you can use broadcast, multicast and anycast to avoid needing detailed configuration
  - Broadcast and Multicast measure RTT once, then assume it doesn’t change
  - Anycast uses broadcast to find servers, then does full NTP exchanges with the servers it finds (and periodically re-finds).
- If it’s supported and working (requires modern NTP and modern networking), Anycast is far and away the best approach

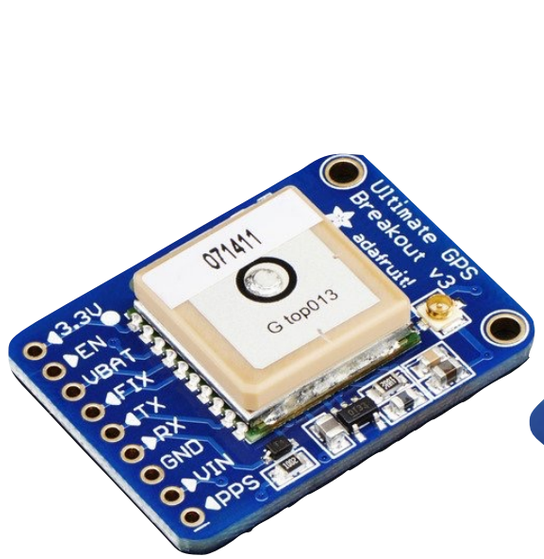
NTP Authentication

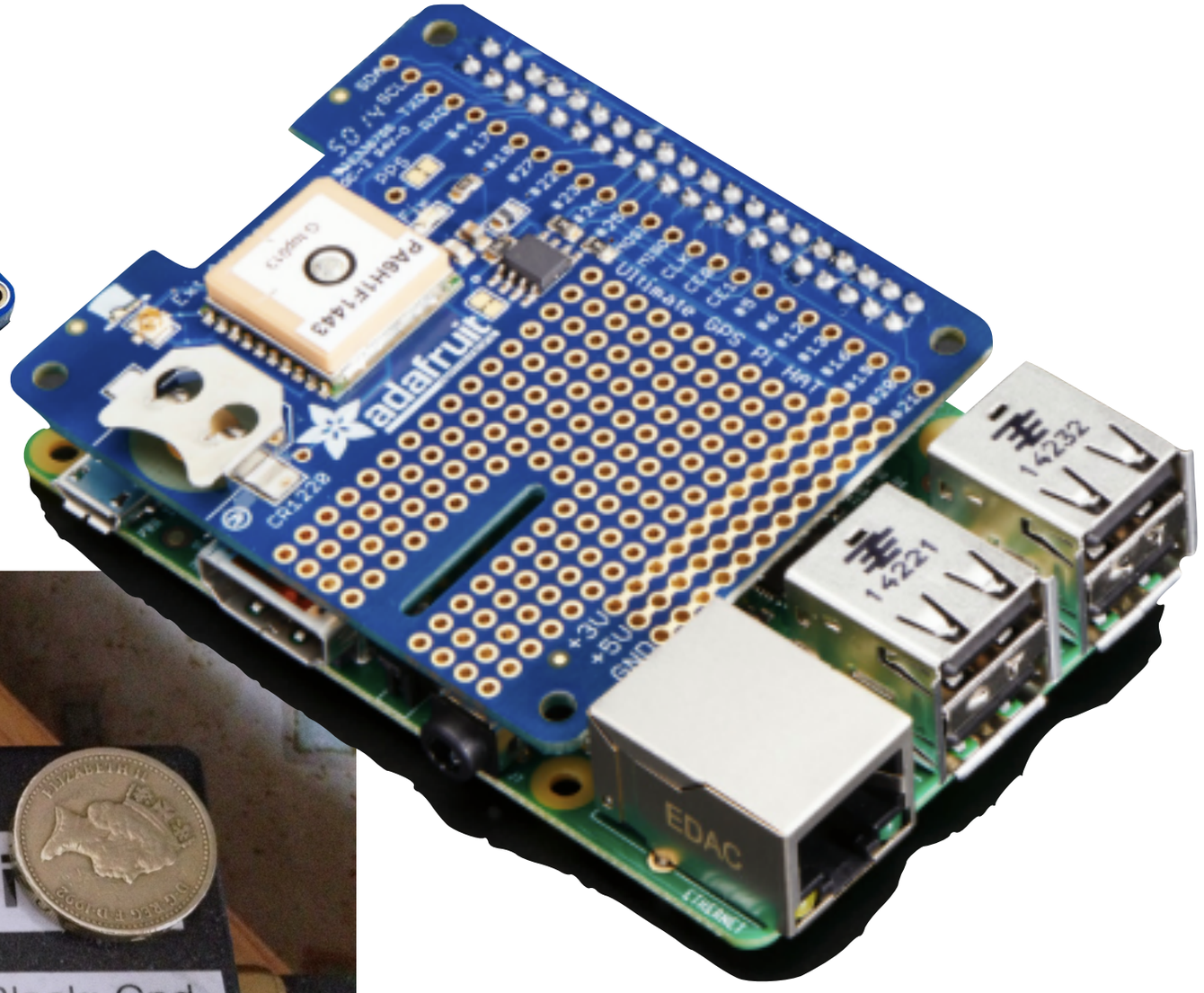
- Pretty much essential for multi/broad/anycast.
- “Old style” is like SNMP: insert a string, hash, replace location of string with result
- Requires all parties to know secret, so every client is equipped to forge packets from server: not scalable.
- “New style” uses public key to generate session keys: complex, but works much better
- Other option is IPSec or some other VPN.

Pro Tip

- It costs about £50 to build a “good enough” Stratum Zero clock reference (Raspberry Pi plus an Adafruit GPS module).
    - If placed in a machine room, my problems with central heating don’t arise.
    - If dedicated to the job, backups aren’t an issue.
- Pre-built boxes from Meinberg are of the order of £2000.
- In exchange a whole suite of problems with logging, modification times, replay-protection, transactions, etc, go away.

£20 GPS Modules





NTP Issues

- No way to (accurately) compensate for asymmetric networks
- Authentication model is complete but not used as much as it should be (“rogue clock” attacks)
- ntpd is a very complex codebase, with lots of risks of amplification attacks
- Dave Mills is 80 and reportedly blind, Harlan Stern is 63 and nearly bankrupt. But the whole internet uses the code they maintain.
- OpenNTPD and chrony are niche (main problem is lack of wide refclock support) but worth investigating.