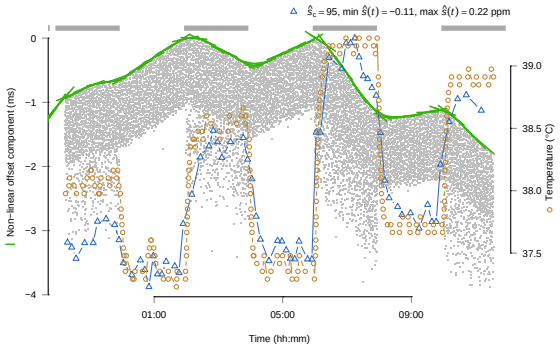
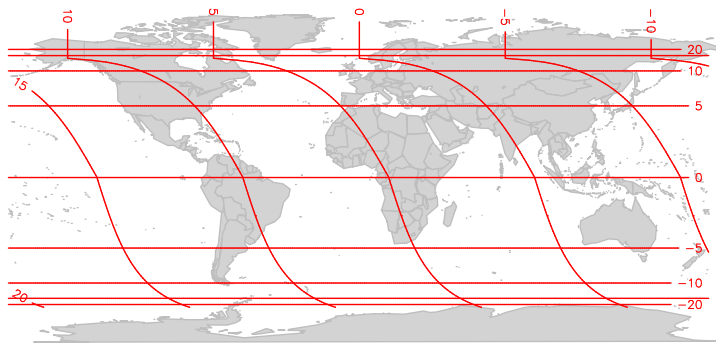
Hot or Not: Fingerprinting hosts through clock skew





Steven J. Murdoch

www.cl.cam.ac.uk/users/sjm217

Sebastian Zander

caia.swin.edu.au/cv/szander

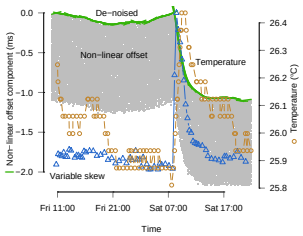
UNIVERSITY OF CAMBRIDGE

Computer Laboratory

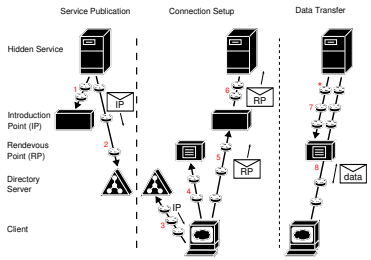
www.torproject.org

EuroBSDCon, 14–15 September 2007, Copenhagen, Denmark

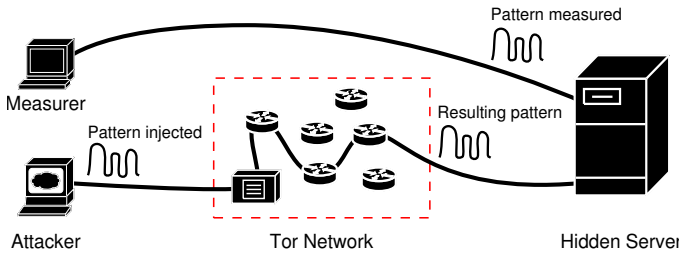
This presentation introduces clock skew and how it can introduce security vulnerabilities



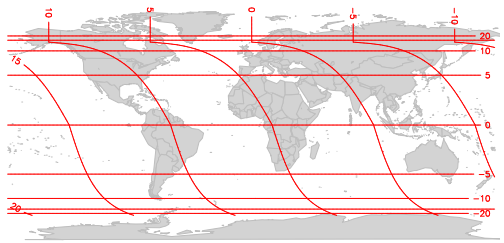
Clock skew, its link to temperature, and measurement



Tor and hidden services



Attacking Tor with clock skew



Applications and improvements

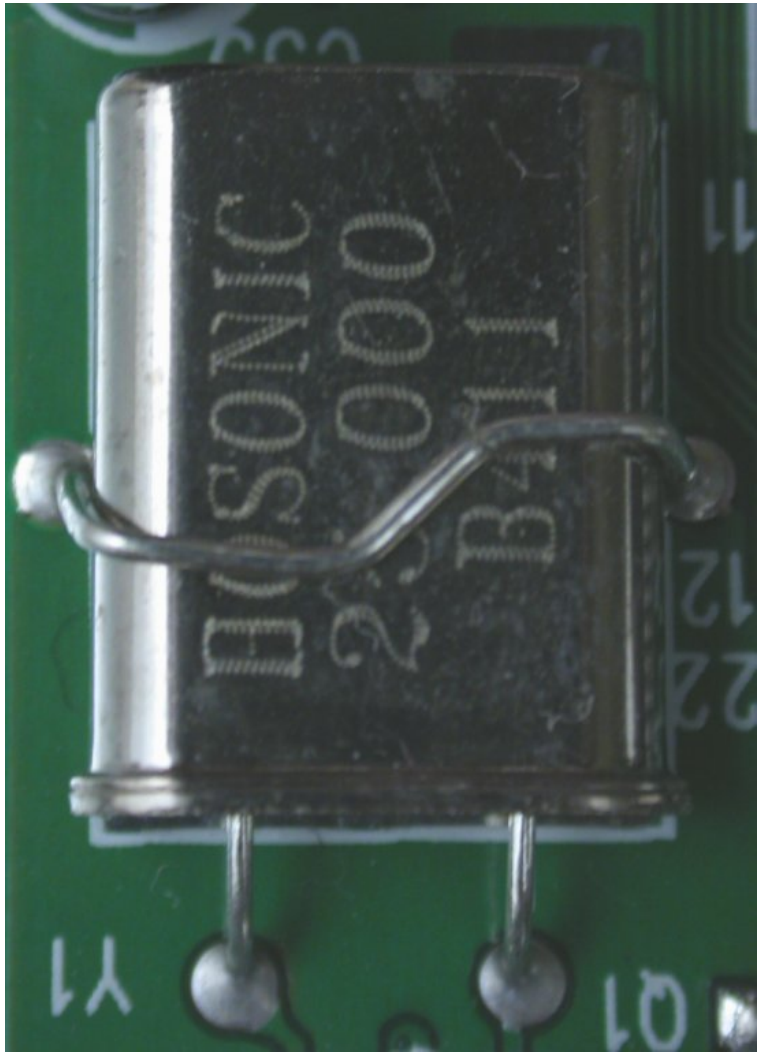
Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at a nominal frequency
- *Counter*, counts the number of ticks produced by the oscillator

On FreeBSD there are several clocks available (other OS are similar):

- *Tick counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off



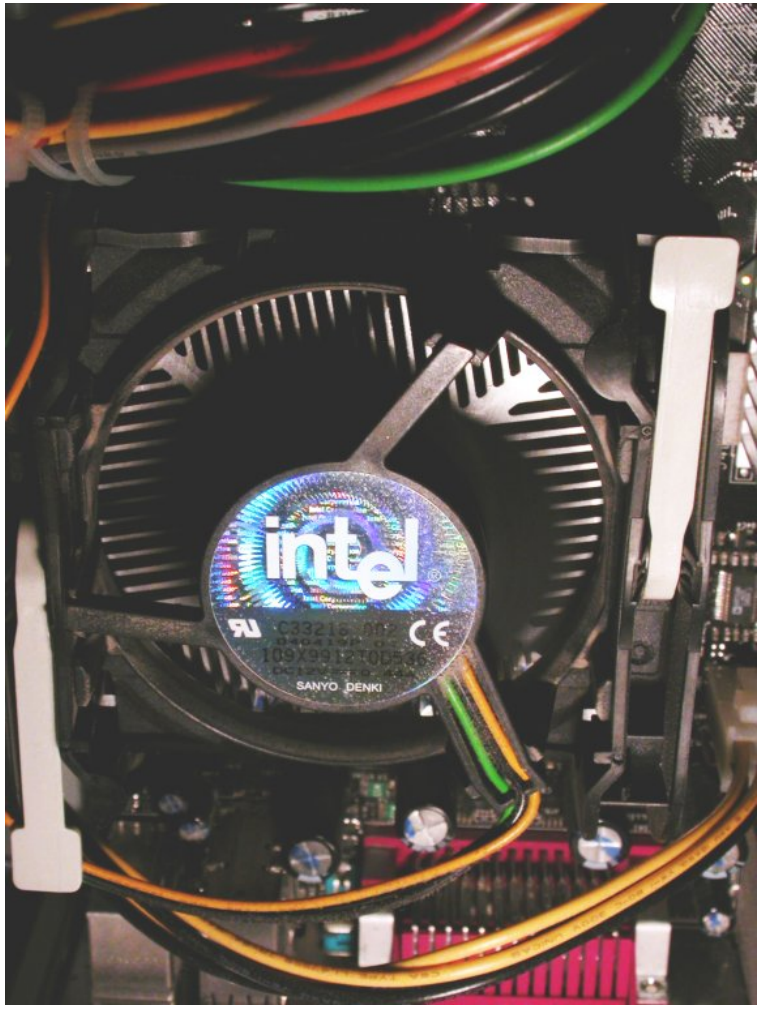
Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at a nominal frequency
- *Counter*, counts the number of ticks produced by the oscillator

On FreeBSD there are several clocks available (other OS are similar):

- *Tick counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off



Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at a nominal frequency
- *Counter*, counts the number of ticks produced by the oscillator

On FreeBSD there are several clocks available (other OS are similar):

- *Tick counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off

```
/*
 * The real-time timer, interrupting hz
 */
void
hardclock(int usermode, uintfptr_t pc)
{
        int need_softclock = 0;

        hardclock_cpu(usermode);

        tc_ticktock();
        /*
         * If no separate statistics cl
         *
         * XXX: this only works for UP
         */
        if (stathz == 0) {
                profclock(usermode, pc)
                statclock(usermode);
        }

#ifdef DEVICE_POLLING
        hardclock_device_poll();
#endif /* DEVICE_POLLING */

        /*
         * Process callouts at a very l
         * relatively high clock interr
         */
```

Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at a nominal frequency
- *Counter*, counts the number of ticks produced by the oscillator

On FreeBSD there are several clocks available (other OS are similar):

- *Tick counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off

```
#ifndef _SYS_SYSPROTO_H_
struct gettimeofday_args {
        struct  timeval *tp;
        struct  timezone *tzp;
};
#endif
/* ARGSUSED */
int
gettimeofday(struct thread *td, struct
{
        struct timeval atv;
        struct timezone rtz;
        int error = 0;

        if (uap->tp) {
                microtime(&atv);
                error = copyout(&atv, u
        }
        if (error == 0 && uap->tzp != N
                rtz.tz_minuteswest = tz
                rtz.tz_dsttime = tz_dst
                error = copyout(&rtz, u
        }
        return (error);
}

#ifndef _SYS_SYSPROTO_H_
struct settimeofday_args {
        struct  timeval *tv;
```

Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

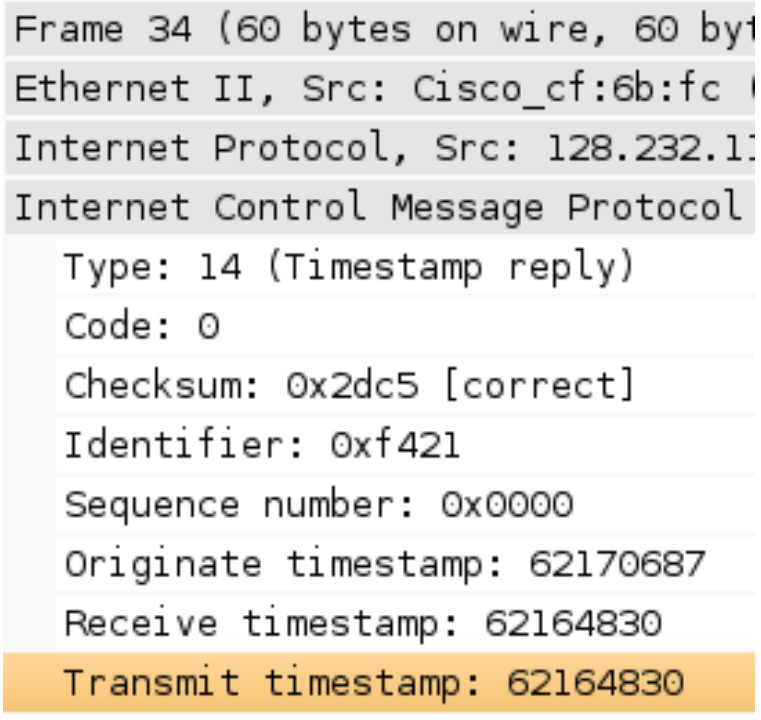
- *Oscillator*, controlled by a crystal, ticks at a nominal frequency
- *Counter*, counts the number of ticks produced by the oscillator

On FreeBSD there are several clocks available (other OS are similar):

- *Tick counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off

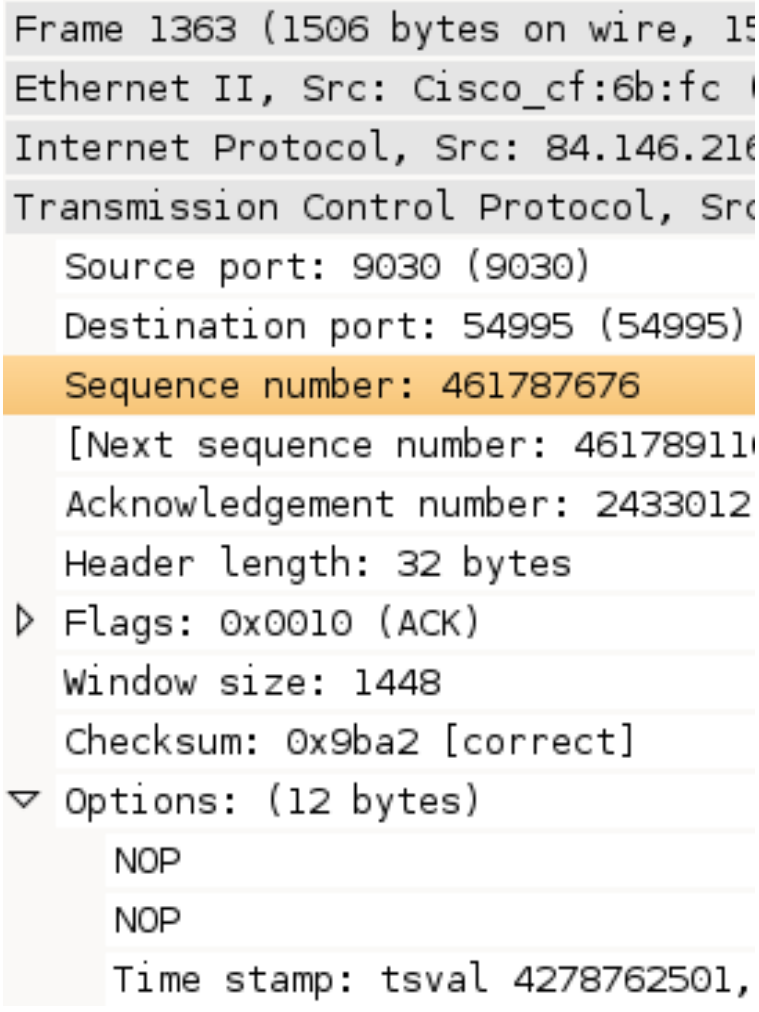
Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Works for Linux, 1 MHz, generated from system clock, rewriting needs state on firewalls, (more in my 22C3 talk)
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from tick counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



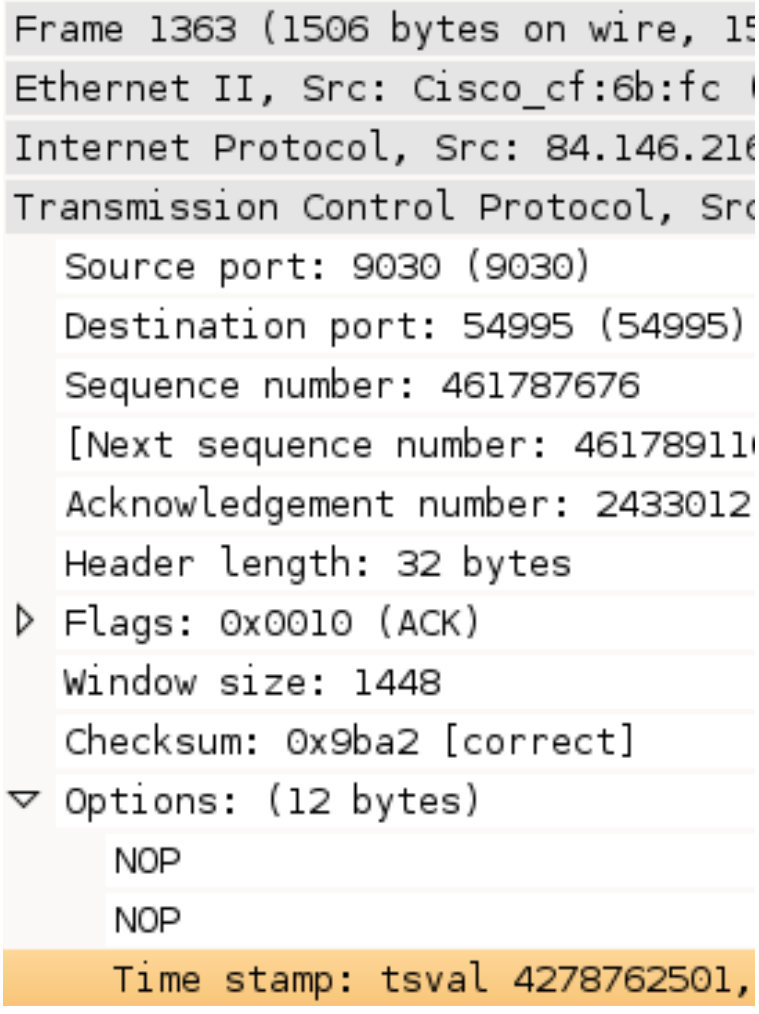
Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Works for Linux, 1 MHz, generated from system clock, rewriting needs state on firewalls, (more in my 22C3 talk)
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from tick counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Works for Linux, 1 MHz, generated from system clock, rewriting needs state on firewalls, (more in my 22C3 talk)
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from tick counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



Measured clock skew acts as a fingerprint of a computer (Kohno *et al.*, 2005)

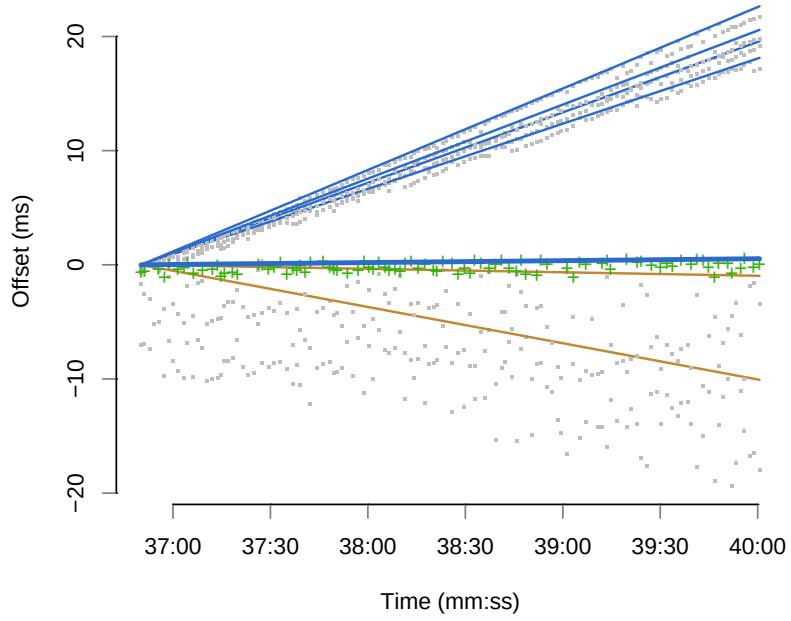
Offset:

- The difference between two clocks (ms)

↓

Skew:

- The rate of change of offset (ppm)
- Stable on one machine ($\pm$1–2 ppm), but varies over different machines (up to $\pm$50 ppm)
- Can give 4–6 bits of information on machine identity

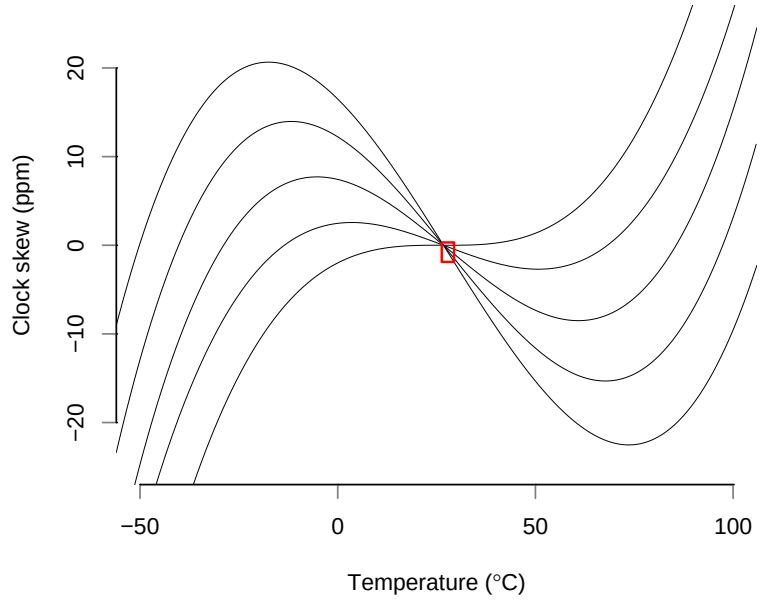


Fingerprinting computers allows identification of hosts and virtual machines

- Identify machines, as they change IP address, ISP and even physical location
- De-anonymise network traces
- Detecting whether a host is running on a virtual machine
- Confirming whether a group of hosts are running on the same hardware (e.g. a honeynet)
  - Honeyd has now been modified to produce different clock-skew fingerprints for virtual hosts
- Counting number of hosts behind a NAT
- The paper did note that clock skew can be affected by temperature, but did not explore the full potential

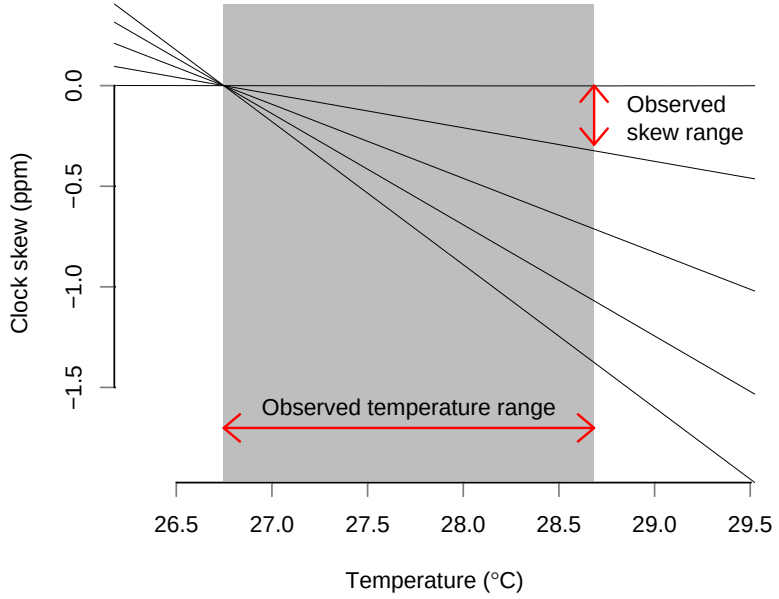
Temperature has a small, but remotely measurable, effect on clock skew

- Skew of typical clock crystal will change by ± 20 ppm over 150 °C operational range
- In typical PC temperatures, only around ± 1 ppm
- By requesting timestamps and measuring skews, an estimate of temperature changes can be derived
- Even in a well-insulated building, changes in temperature over the day become apparent



Temperature has a small, but remotely measurable, effect on clock skew

- Skew of typical clock crystal will change by ± 20 ppm over 150 ° C operational range
- In typical PC temperatures, only around ± 1 ppm
- By requesting timestamps and measuring skews, an estimate of temperature changes can be derived
- Even in a well-insulated building, changes in temperature over the day become apparent



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

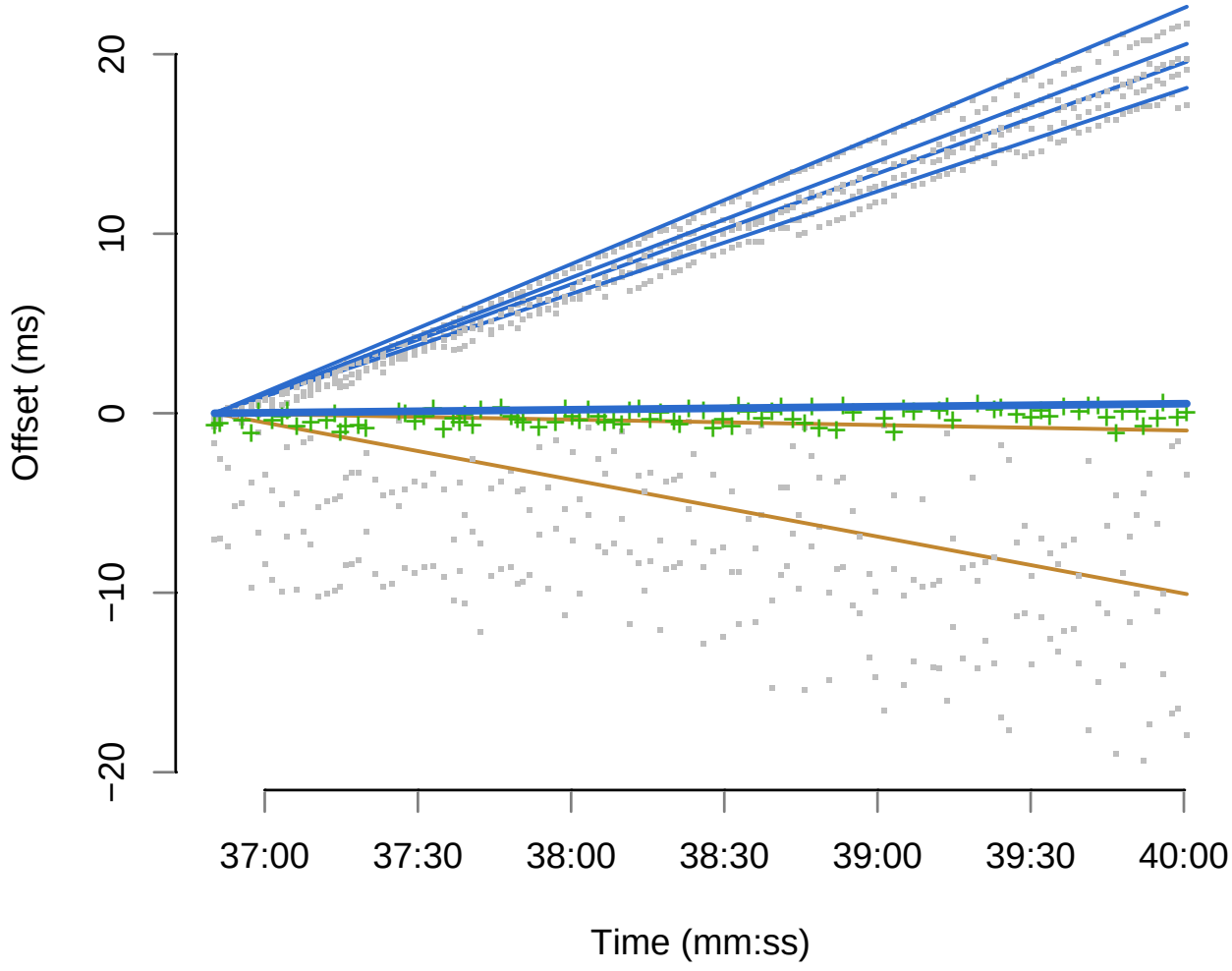
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

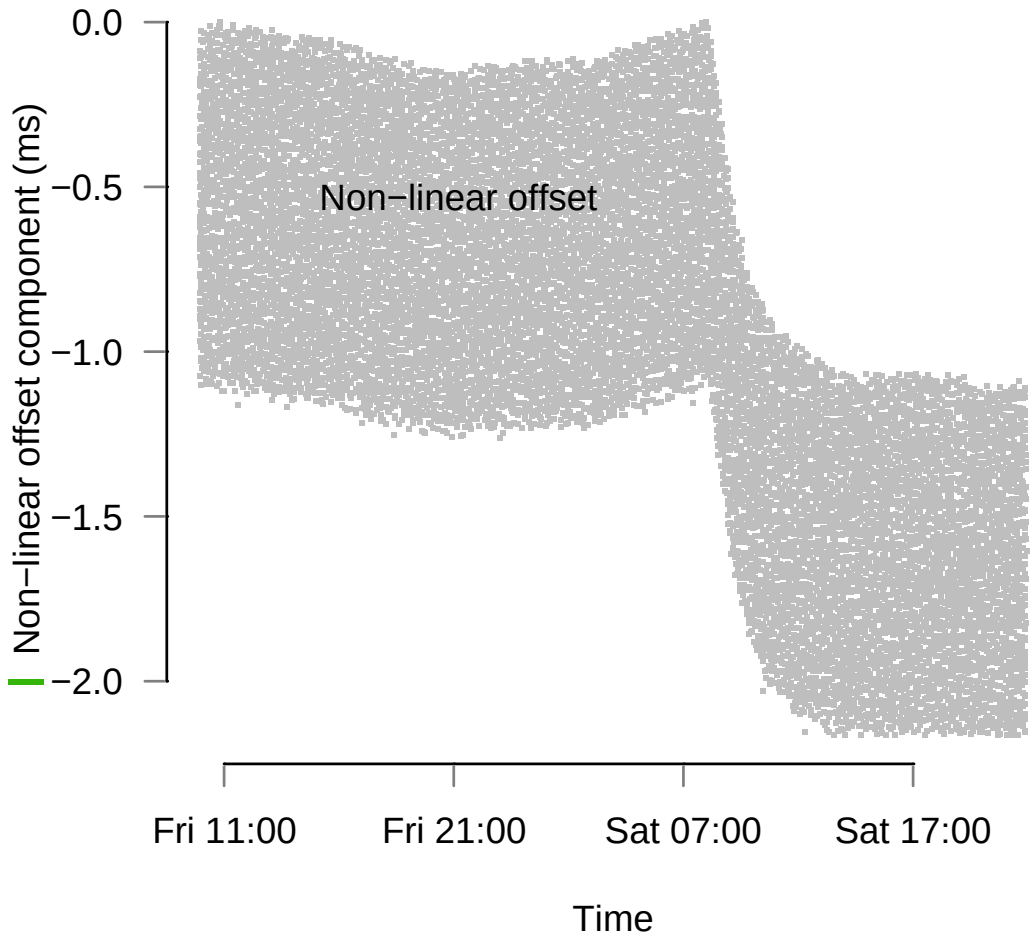
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

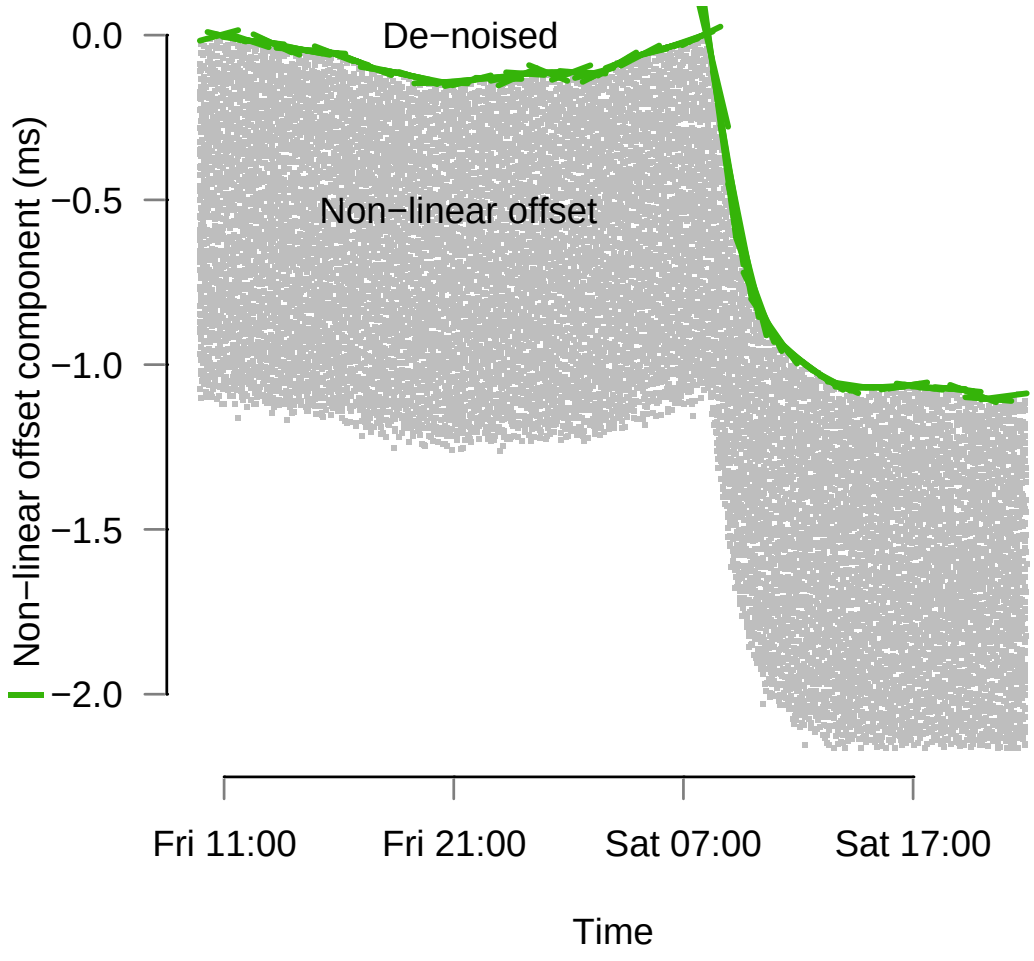
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

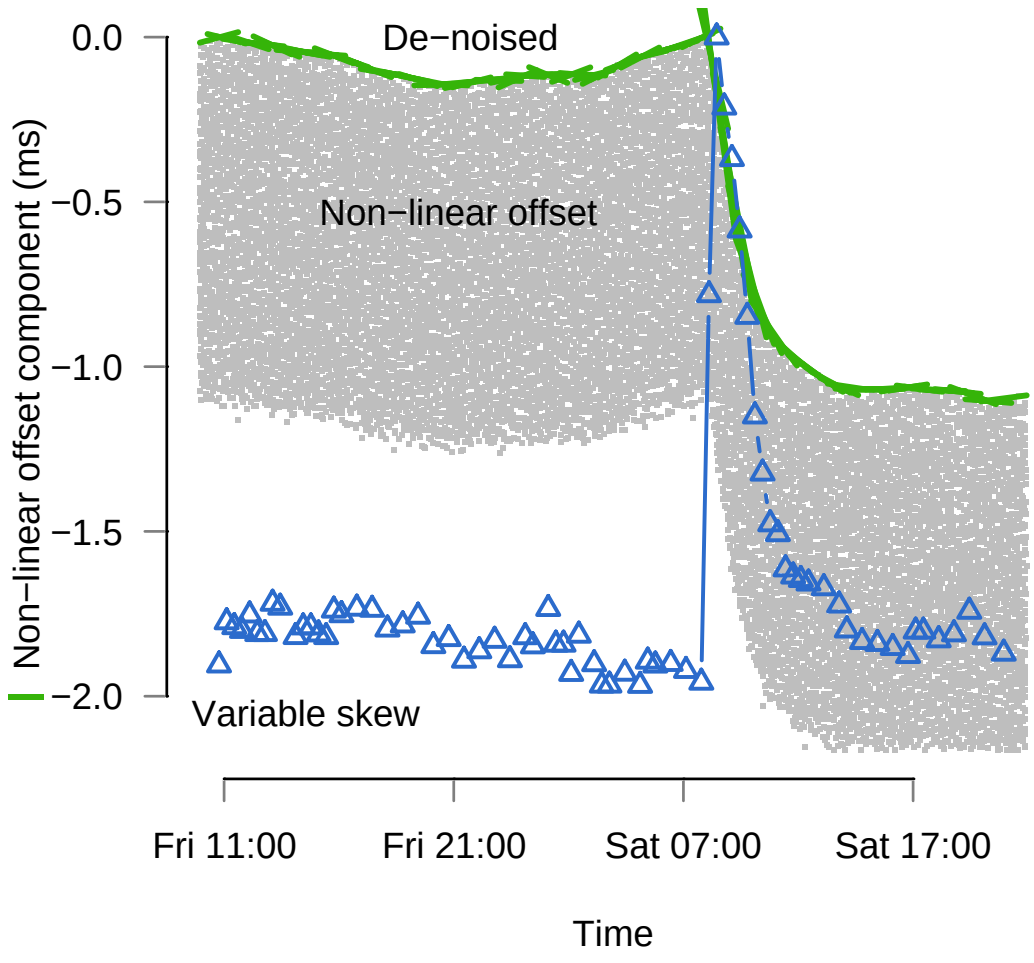
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

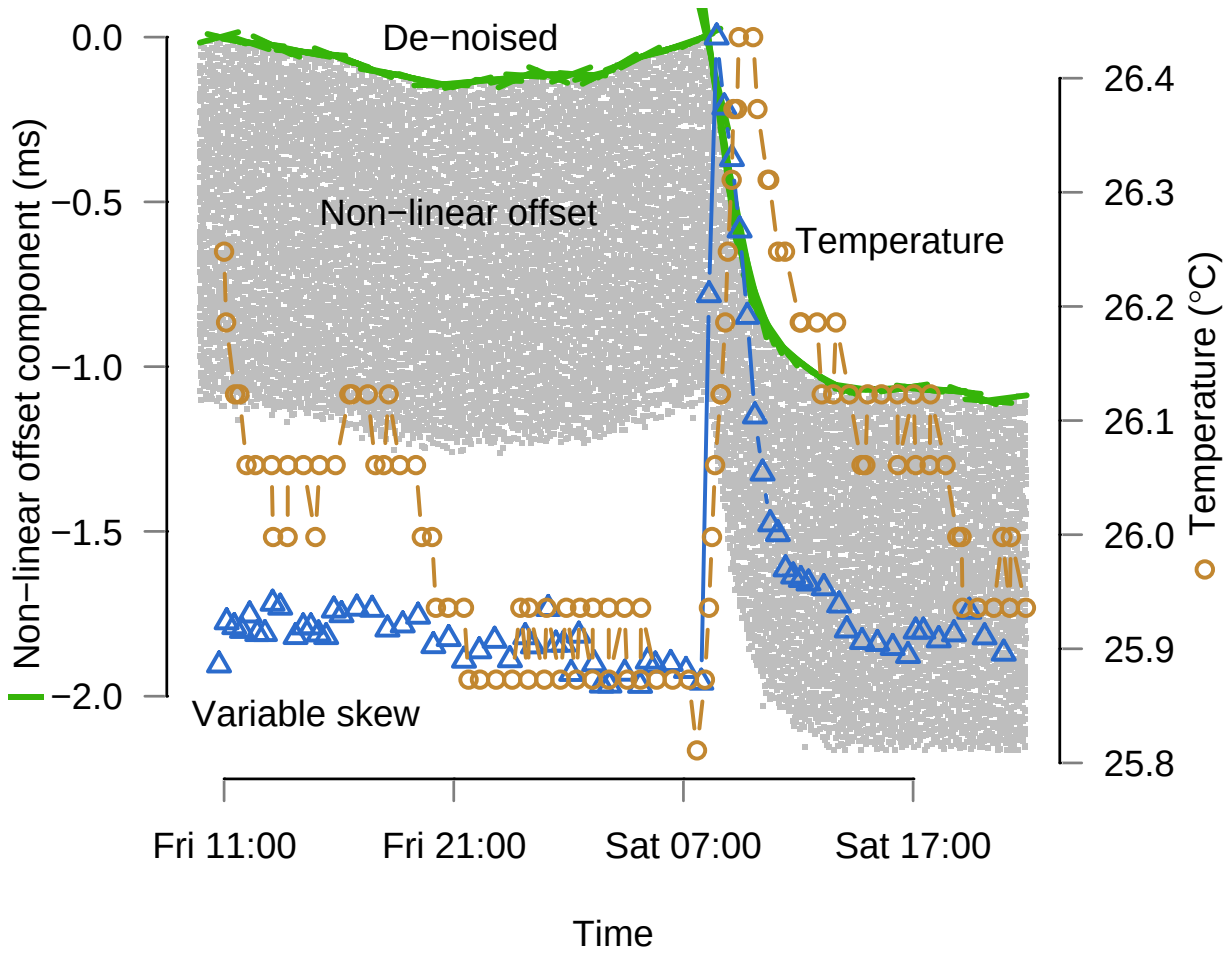
Remove noise

↓

Differentiate

↓

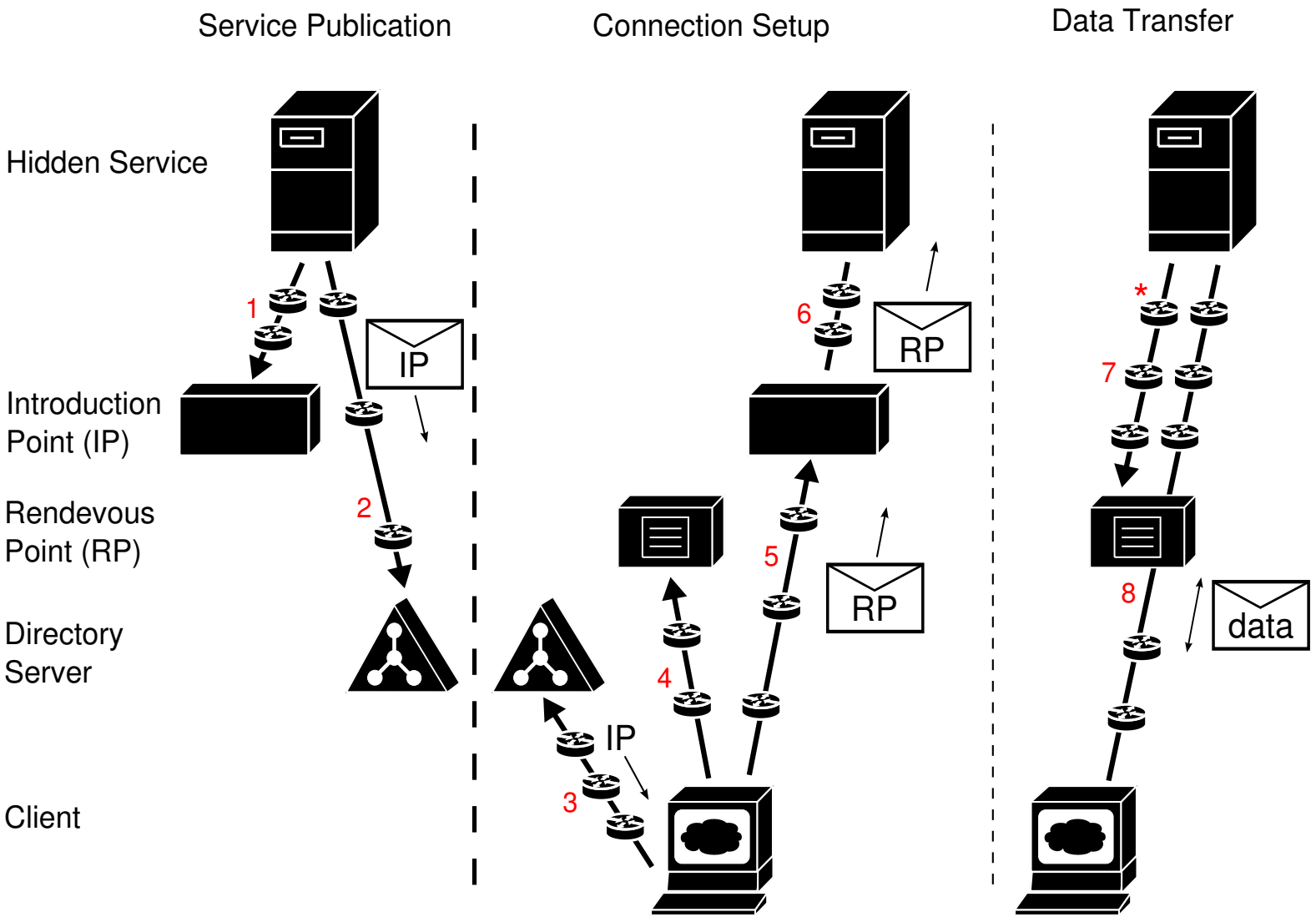
Compare to temperature



Tor is a low-latency, distributed anonymity system

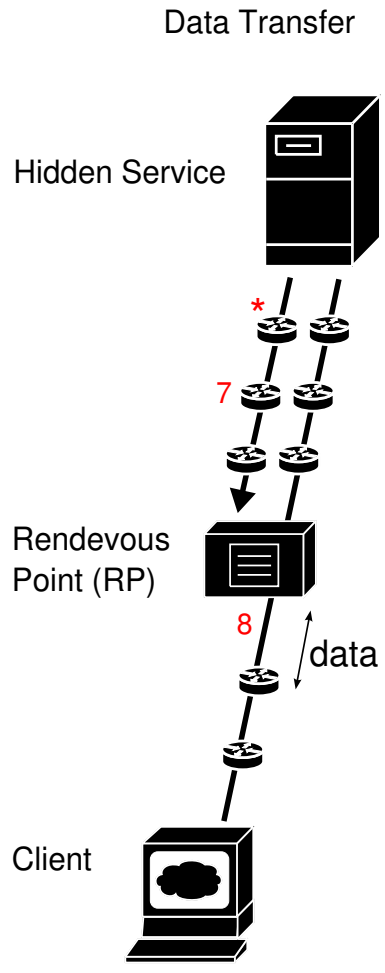
- Real-time TCP anonymisation system (e.g. web browsing)
- Supports anonymous operation of servers (hidden services)
- These protect the user operating the server and the service itself
- Constructs paths through randomly chosen nodes (around 1 000 now)
- Multiple layers of encryption hide correlations between input and output data
- No intentional delay introduced

Hidden services are built on top of the anonymity primitive the Tor network provides



The IP address of hidden services can be found through traffic analysis (Øverlier, Syverson, 2006)

- One Tor node (∗), selected at random by the hidden service, knows the hidden service's real IP address
- If a malicious client also controls a Tor node (easy), then eventually his node will be selected on that path
- Data is encrypted, so the malicious Tor node cannot trivially detect when it is being used to access the hidden service
- However enough timing patterns remain to identify this event, and so allowing the malicious client to locate the hidden server
- This attack is now resisted by the hidden service selecting fixed *guard* nodes for ∗

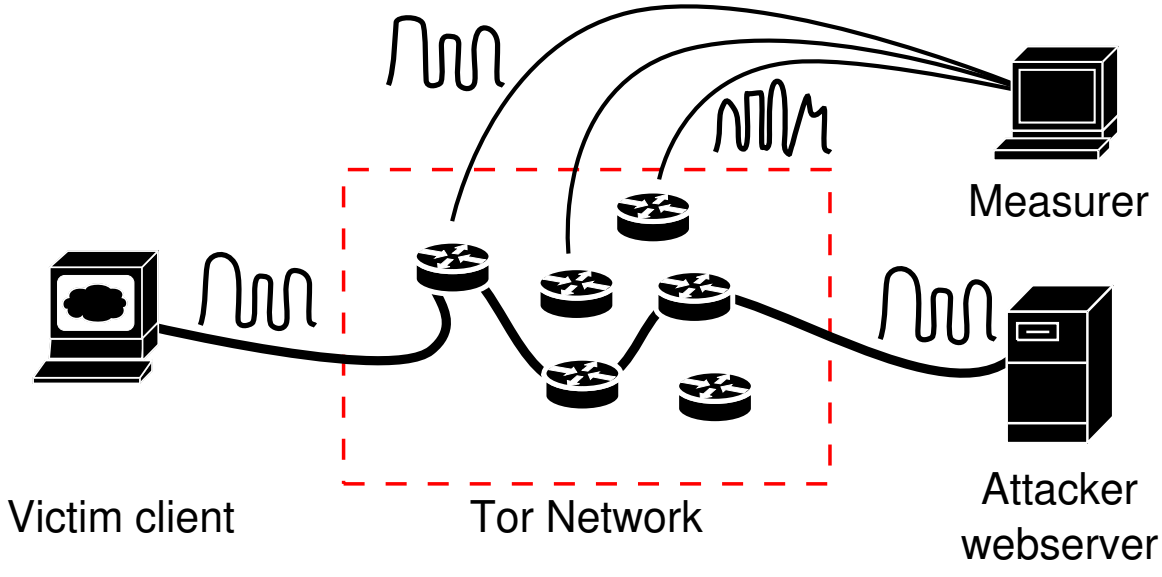


Even if an attacker cannot observe the network, traffic analysis is still possible (Murdoch, Danezis, 2005)

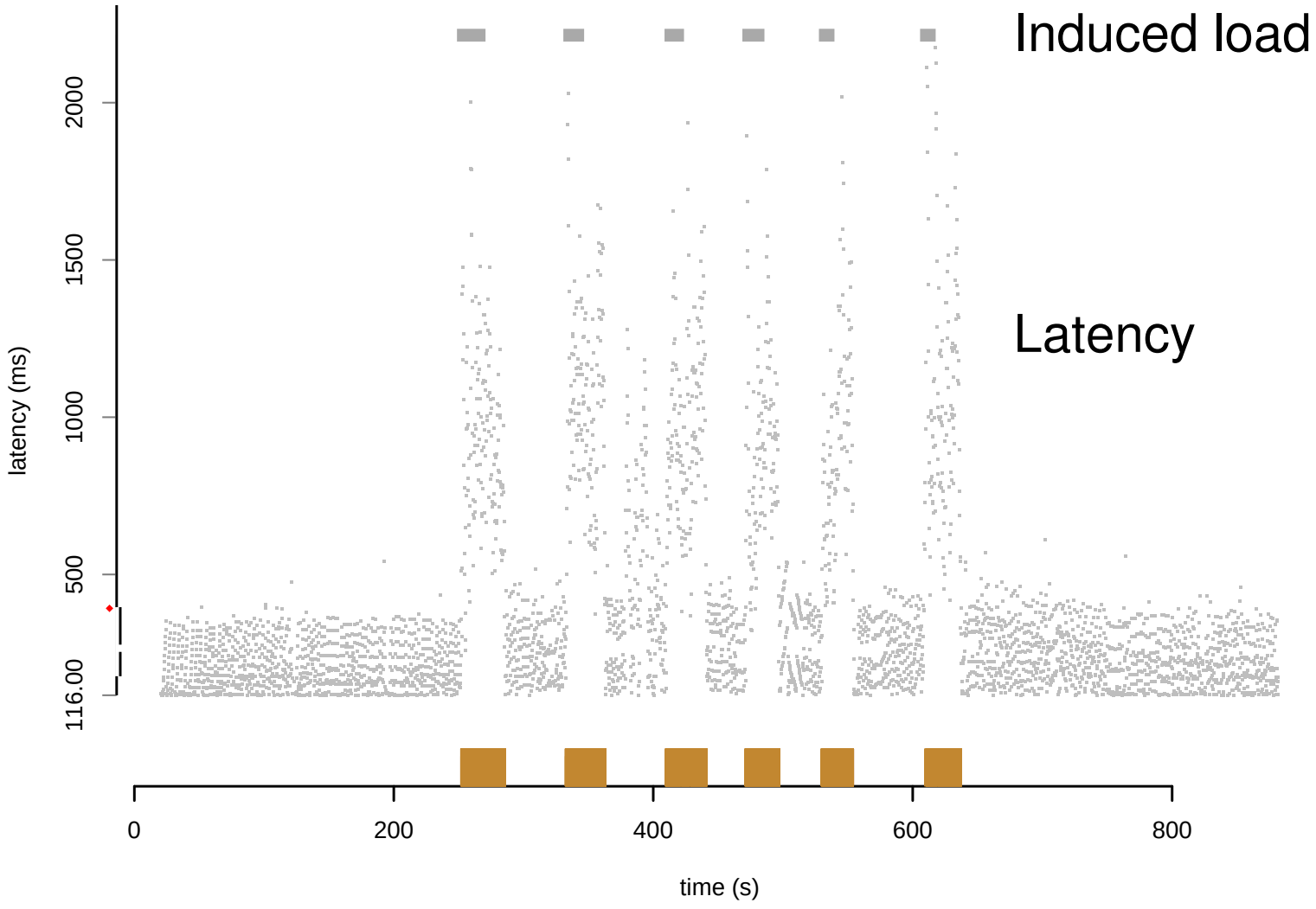
Attacker inserts traffic pattern into anonymous stream
Measurer probes all Tor nodes for their latency

↓

Nodes along path that the anonymous stream takes will exhibit the same pattern

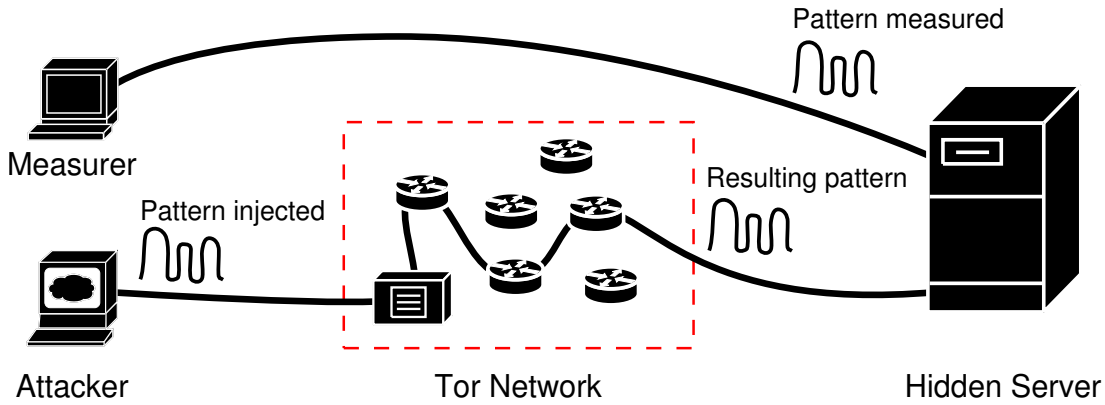


The latency of one connection going through a Tor node is strongly affected by its network load



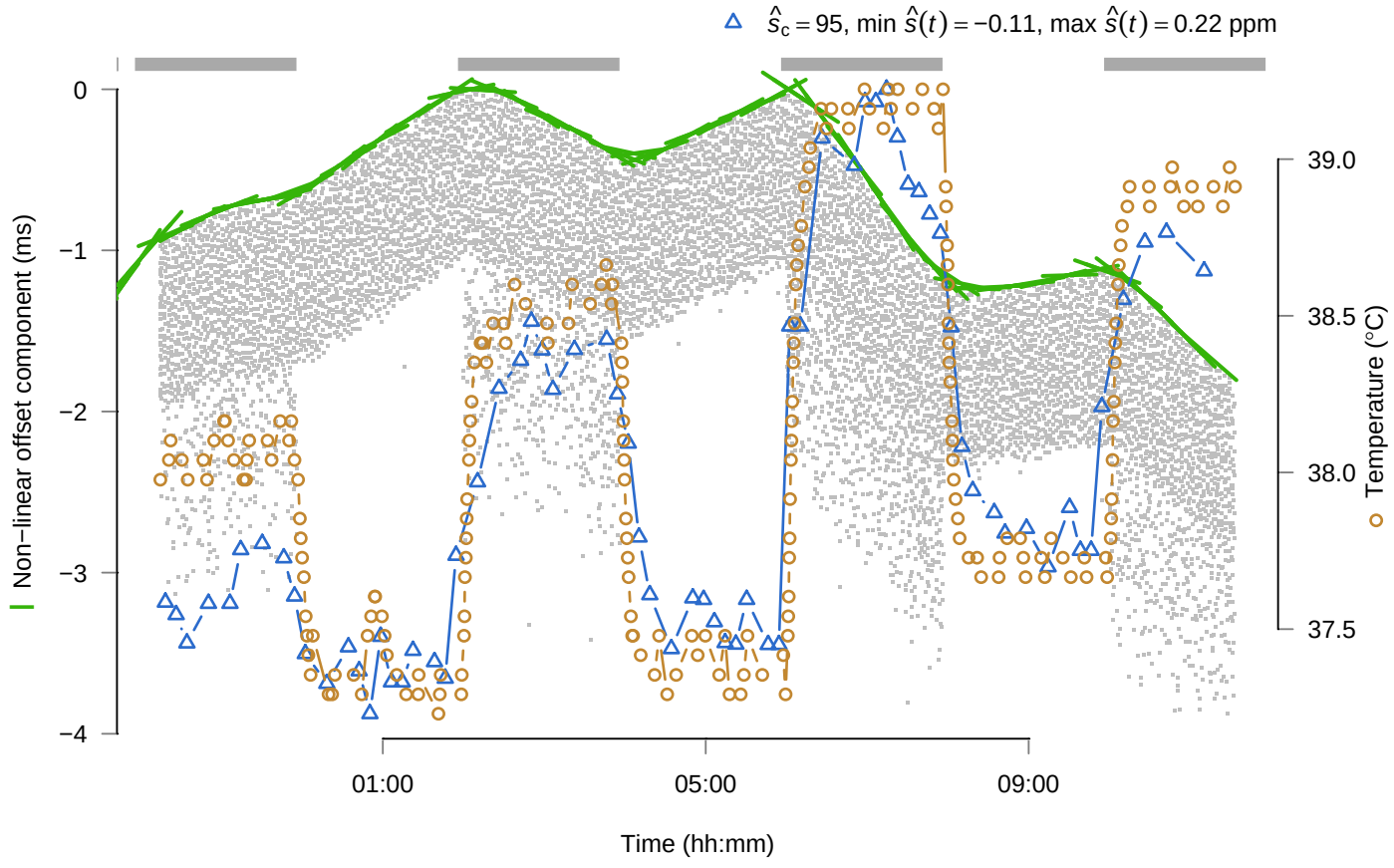
The attack can be resisted with QoS features but there remains a temperature covert channel

- Prevent one stream going through a node from interfering with any others
- Hard QoS guarantee on every stream, and no more connections accepted than there is capacity
- When one stream is not used, no other streams may use the resources released, so CPU will be idle
- Then the CPU will cool down so the clock skew will change accordingly, allowing connections to be tracked
- Validated with Tor hidden services on a private Tor network



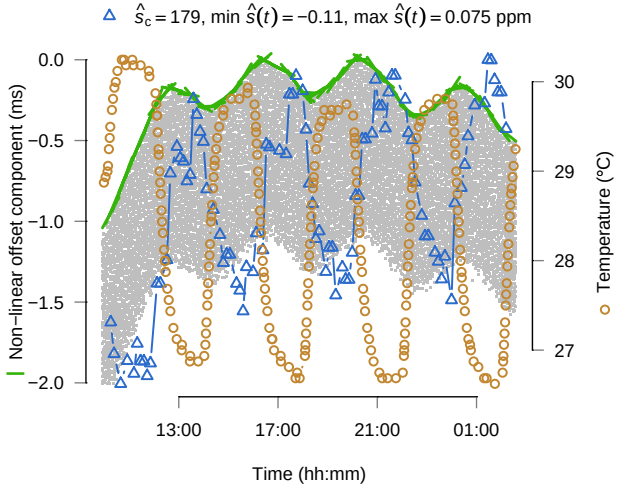
The load of a hidden service can be estimated by measuring temperature induced clock skew

- Attacker induces load by making requests to the hidden server
- Here, a periodic 2 hour on, 2 hour off pattern was used
- Measurer records clock offset and derives temperature



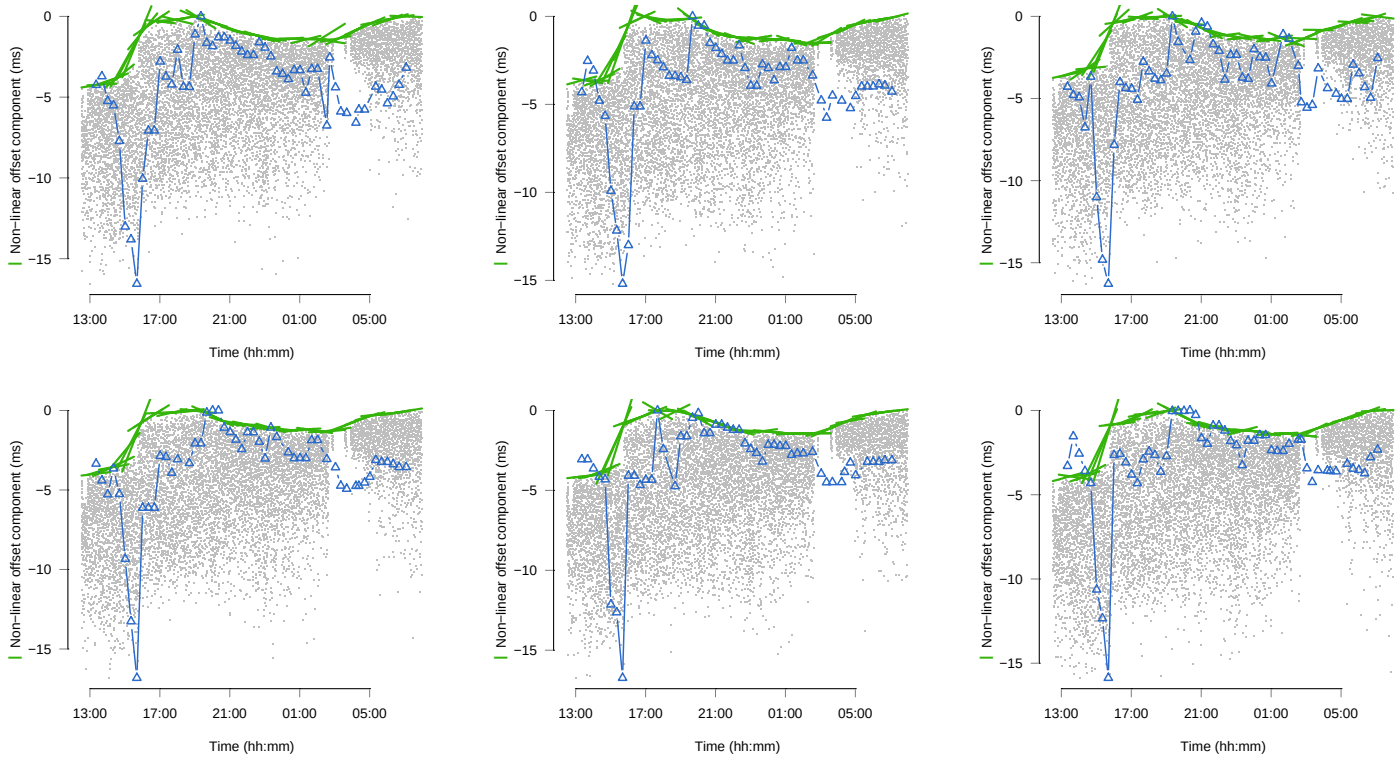
This temperature covert channel can be applied in a variety of different situations

- Inter-process communication through modulating load and hence temperature
  - Fixed scheduling will not defend against this
  - Relies on second time source, affected differently by temperature; could be remote (NTP) or local (sound card)
- Temperature effects can cross “air-gap” security barriers
  - Confirmed in rack-mount computers; plausible for “blade” arrangements too
- General purpose communication



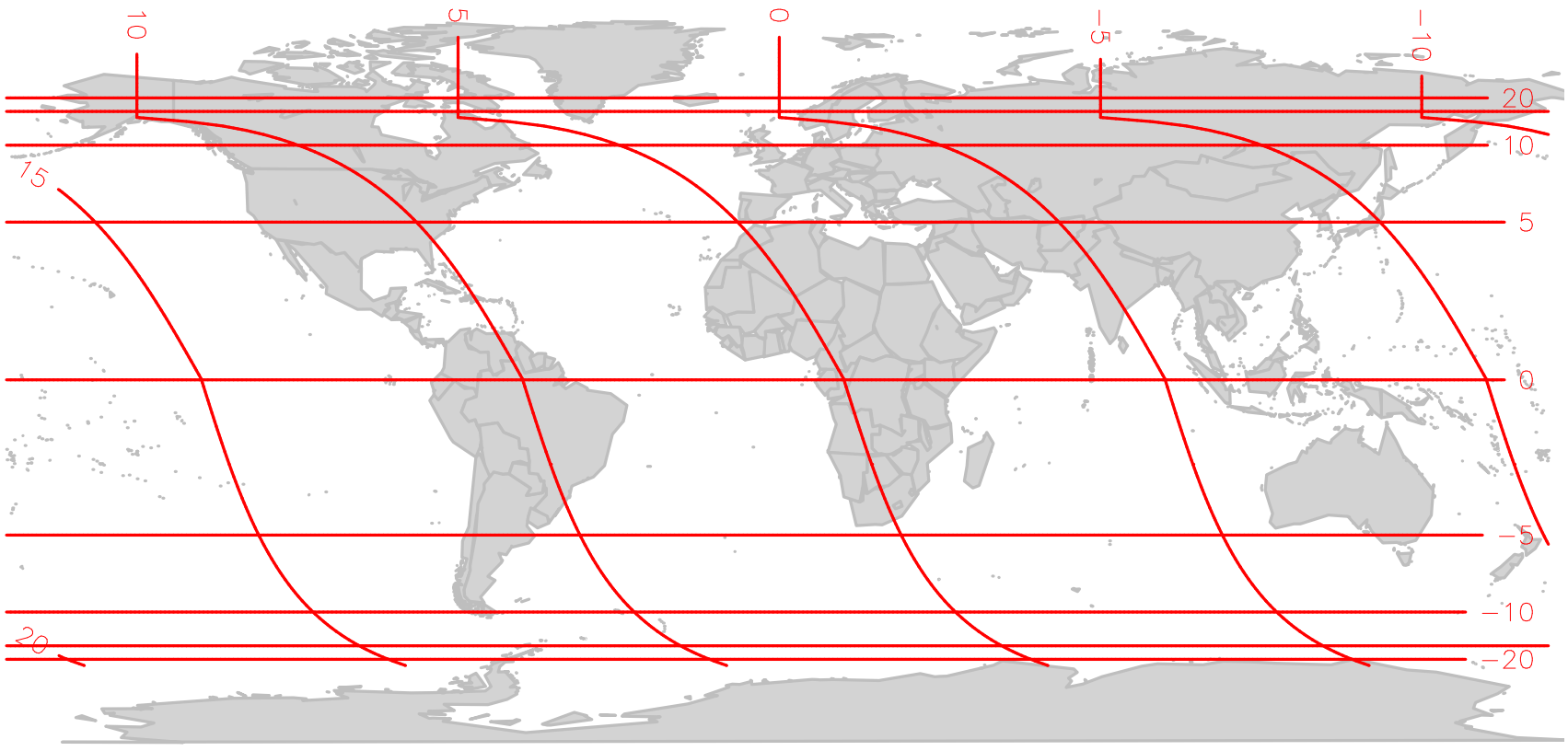
Clock skew identifies both machine identity (absolute skew) and environment (relative)

- Computers can be identified by clock skew
- Temperature information can indicate environment
- Applied to investigate suspected “Sybil” attack on Tor, to discover than the 30 suspicious Tor nodes were actually 2 physical machines

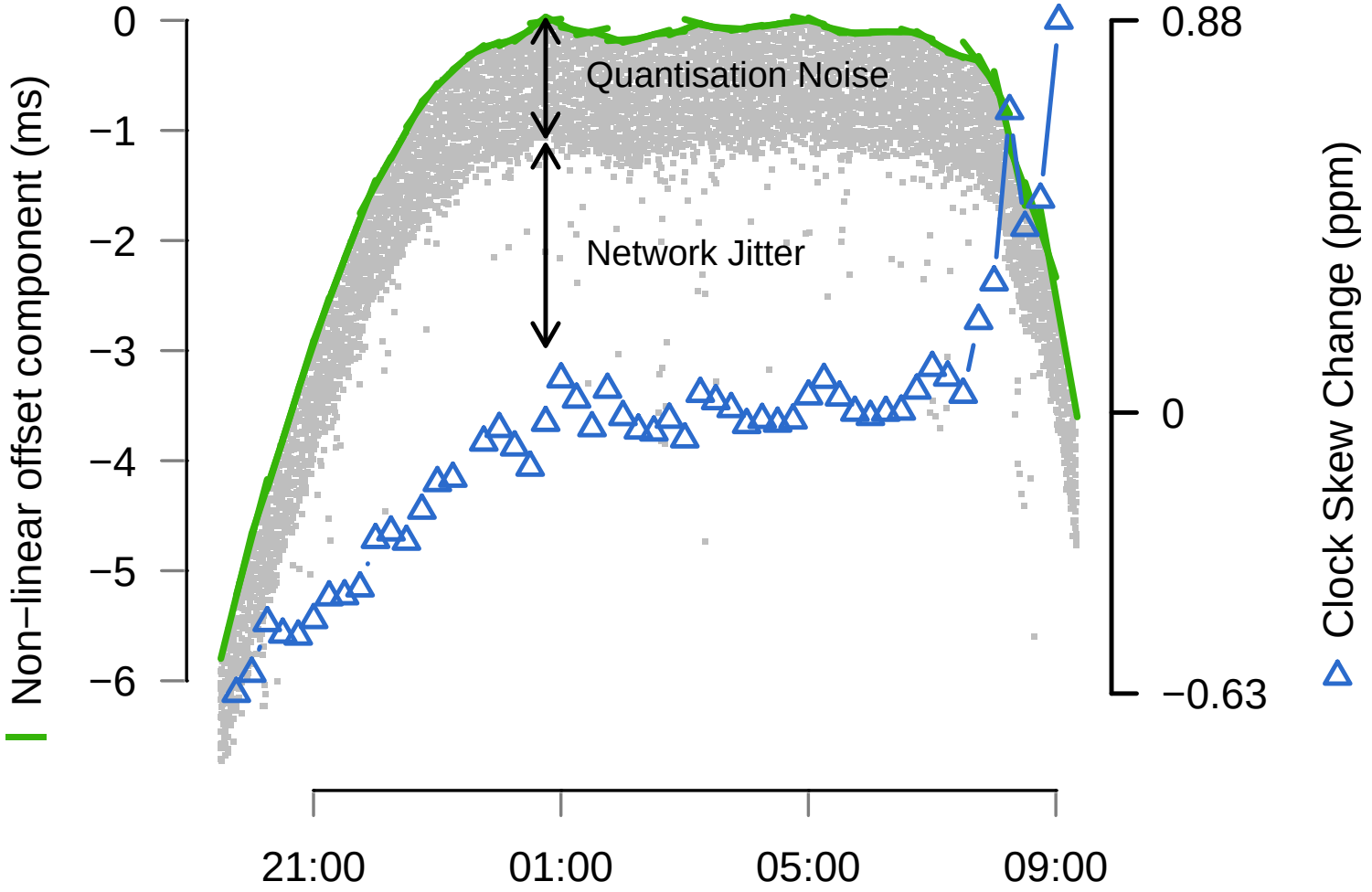


From the changes in temperature of a machine, we can even estimate its location

- If length of day and middle/start/end of day can be found, locations of targets can be found
- Imprecise, time-consuming and affected by local conditions (air conditioning) but perhaps could provide coarse-grained coordinates

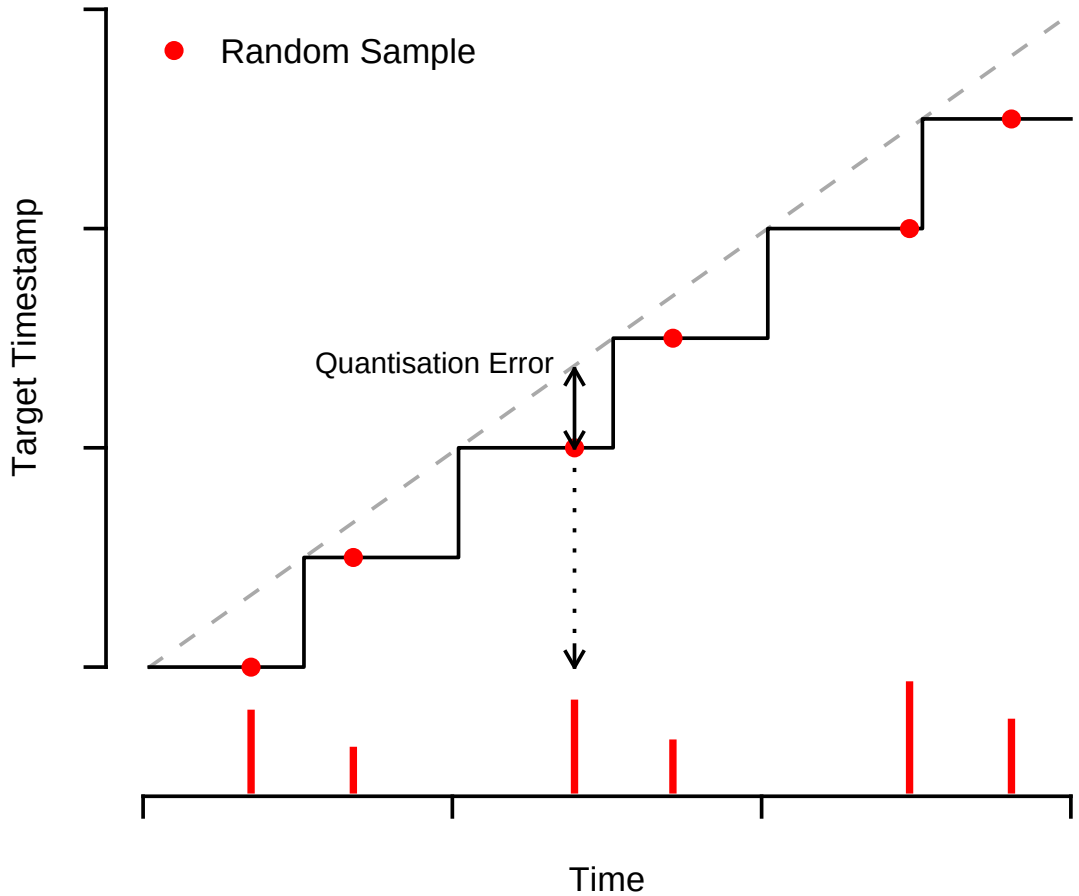


Measurement errors have two sources: quantization noise and network jitter



Many samples, over a long time, are needed to eliminate this noise

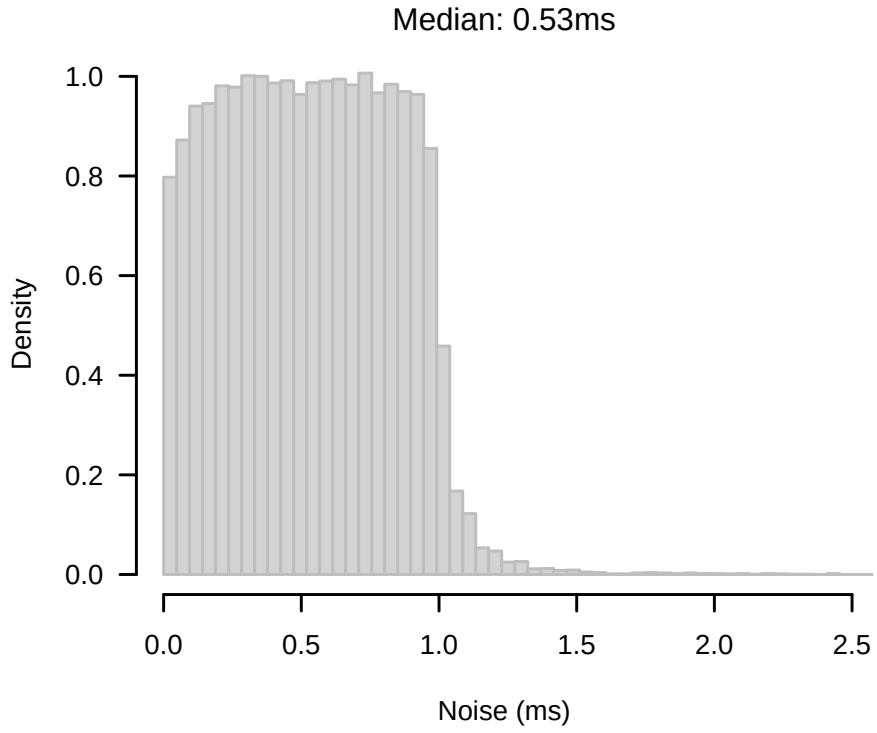
Quantization noise of a sample depends on how close it was to a clock-edge



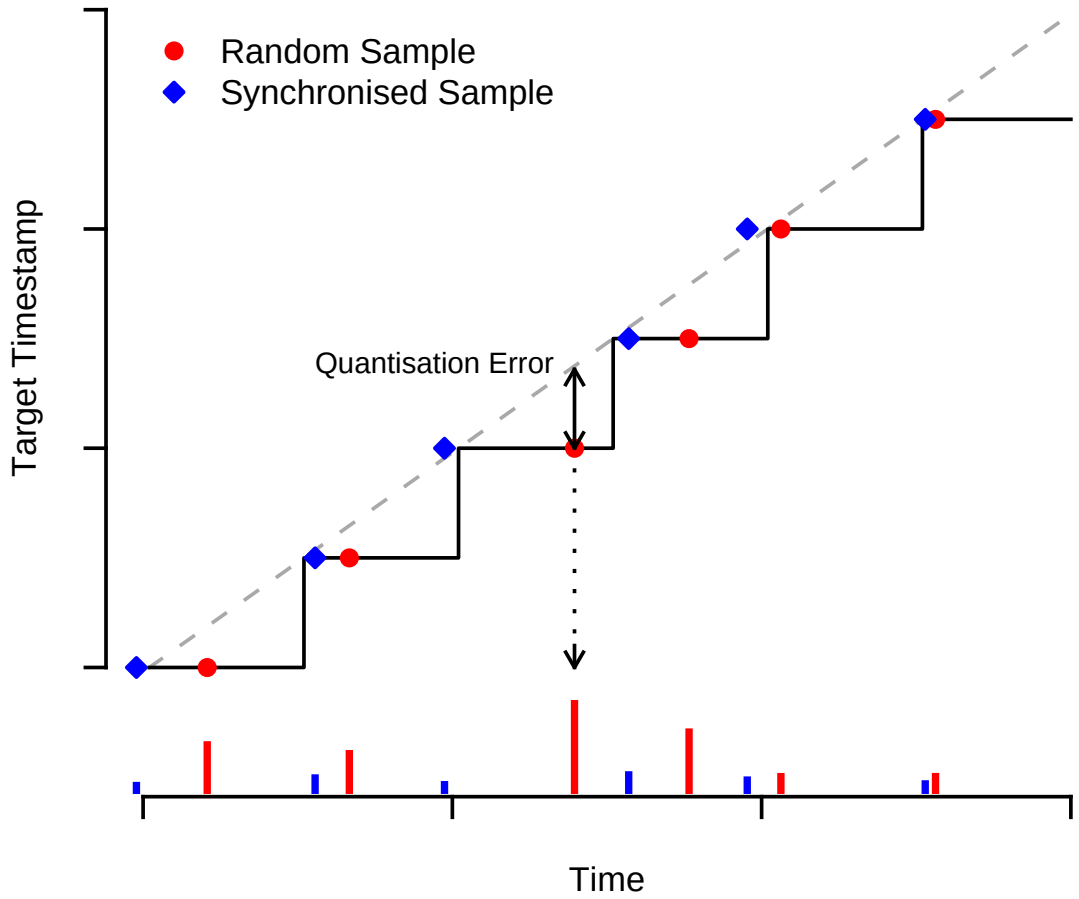
Only the samples made near clock edges contribute to the accuracy of the skew measurement

Quantization noise level depends on the period of the timestamp clock

- For the 1 kHz clock shown here, the maximum quantization error is 1 ms
- 250 Hz clocks in some versions of Linux have a period of 4 ms
- 1 Hz HTTP timestamps, which may be the only one available, have a 1 s period



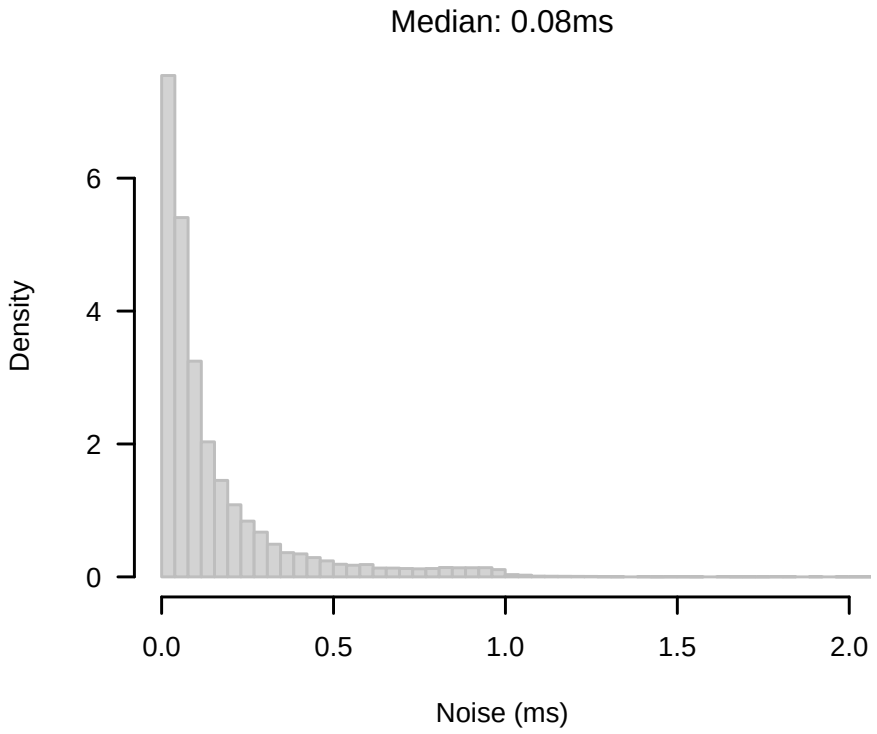
Quantization noise can be effectively eliminated by sampling just before or after clock ticks



Now the noise level is independent of clock frequency

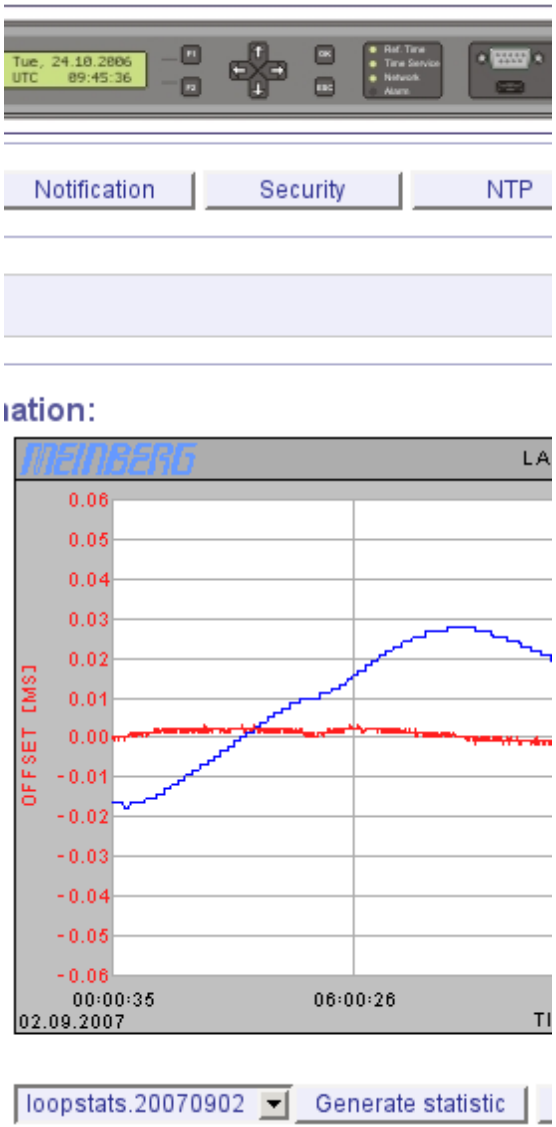
The improved mechanism works like a binary search

- The algorithm first locks onto a clock tick, and predicts the position
- Then it alternately samples a little before and after this point
- If it guessed right, the bounds are tightened, if wrong, it opens them up
- The resulting noise is far lower than random sampling



NTP causes problems for some, but not all variants of timestamp attacks

- By synchronising a clock with NTP, the constant skew is eliminated, defeating fingerprinting attacks
- Variable skew is distorted, but not removed, as the synchronisation process is slow
- Under Linux, the ICMP timestamps and TCP sequence number clocks are NTP-synced, whereas TCP timestamps are not (for good reason)
- Under FreeBSD, ICMP timestamps are also NTP-synced, TCP timestamps are not and TCP sequence numbers are partially randomised so not useful

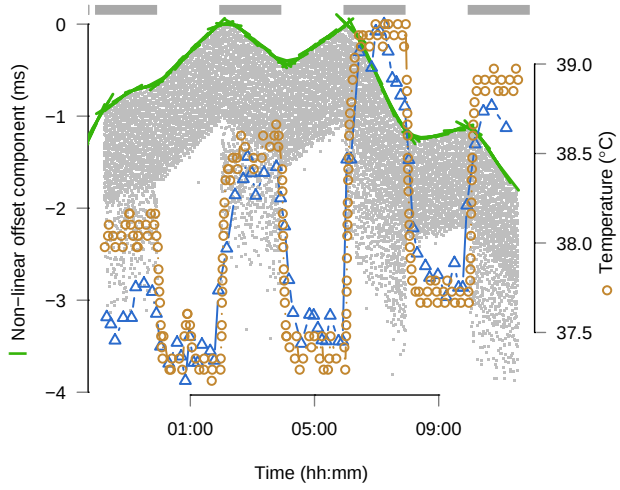


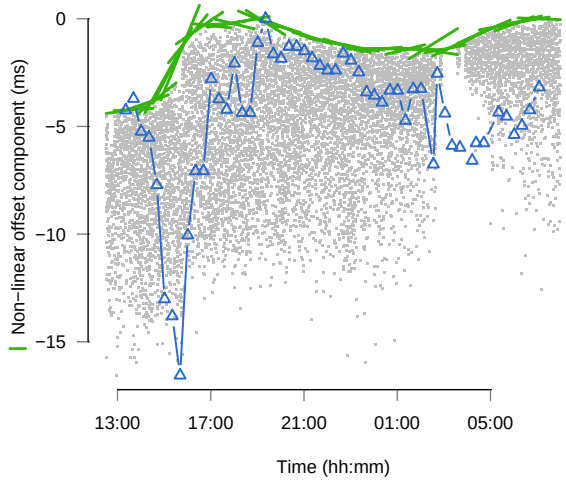
Defending against clock skew attacks is difficult and doesn't come for free

Defence	*Limitations*
Block timing information	Many low-level events are triggered on timer interrupts and this could be detected remotely
Run CPU at full load	Inefficient and must be done with care since different types of tasks can have varying temperature effects
Install a temperature compensated clock crystal	These might not have an adequate $< \pm 1$ ppm accuracy
Install an oven compensated crystal	Expensive, power hungry, but better accuracy

In summary, temperature covert channels are a viable attack even when other vectors are blocked

- Through clock skew, temperature and thus CPU load can be remotely measured, over tens of router hops
- By inducing load on a Tor hidden server and measuring the resulting clock skew, the hidden service pseudonym can be linked to its IP address
- Thermal covert channels are applicable in several other situations
- Even when a system is secure in one model of abstraction, stepping outside these limits can reveal additional attacks

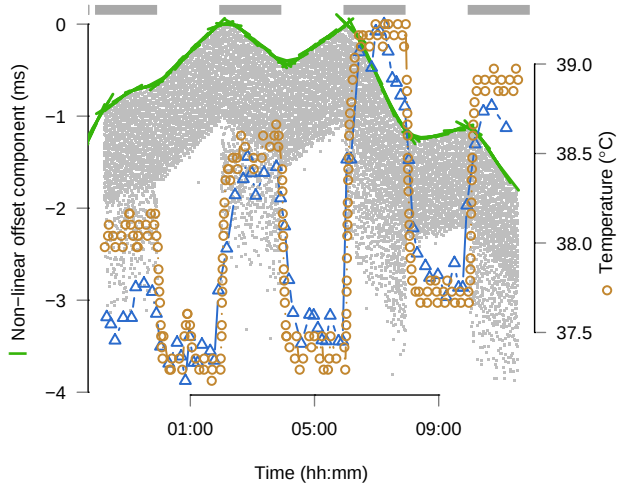


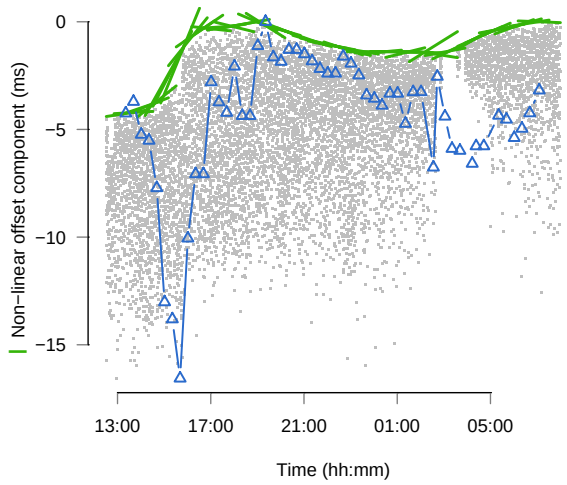


More at `www.cl.cam.ac.uk/users/sjm217`

In summary, temperature covert channels are a viable attack even when other vectors are blocked

- Through clock skew, temperature and thus CPU load can be remotely measured, over tens of router hops
- By inducing load on a Tor hidden server and measuring the resulting clock skew, the hidden service pseudonym can be linked to its IP address
- Thermal covert channels are applicable in several other situations
- Even when a system is secure in one model of abstraction, stepping outside these limits can reveal additional attacks





More at `www.cl.cam.ac.uk/users/sjm217`

Questions?