

CT420 REAL-TIME SYSTEMS

THE NTP PROTOCOL

Dr. Michael Schukat



Recall: Computer Clock Options

2

□ **Option A:**

- Stick to crystals
- Precision manufacturing → costly
- Temperature Compensated Crystal Oscillator (TCXO)
- Oven Controlled Crystal Oscillator (OCXO)
- Works indoors

□ **Option B:**

- Buy an Atomic Clock (\$50,000 - \$100,000)
- .. or GNSS Receiver (based on atomic clock), but doesn't work indoors
- .. or time signal radio receiver, e.g. DCF77, if you are based in central Europe

□ **Option C:**

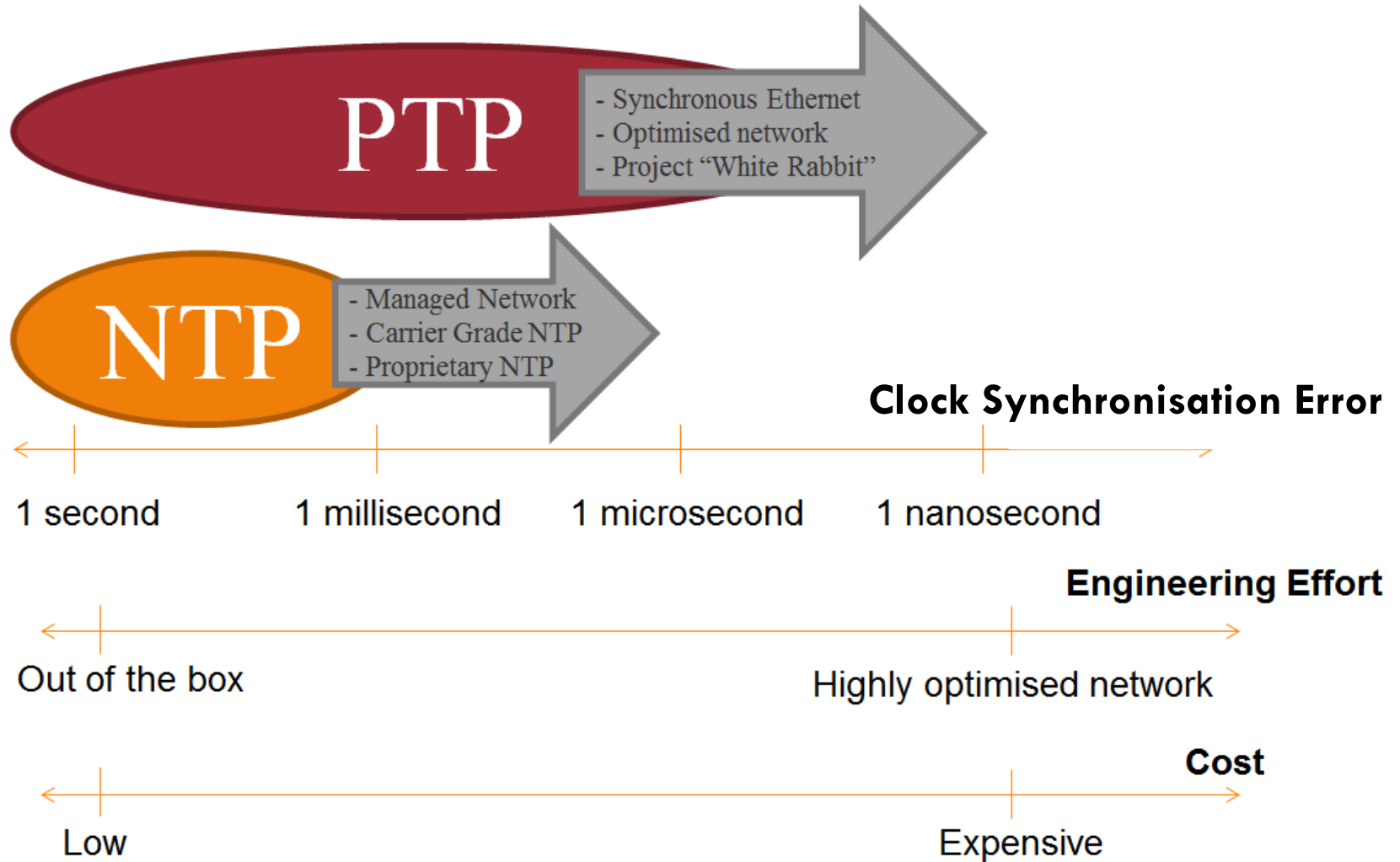
- Software based approach to discipline cheap crystal clocks
- Lesser quality but useful for certain applications
- Works indoors too!

Distributed Master Clocks

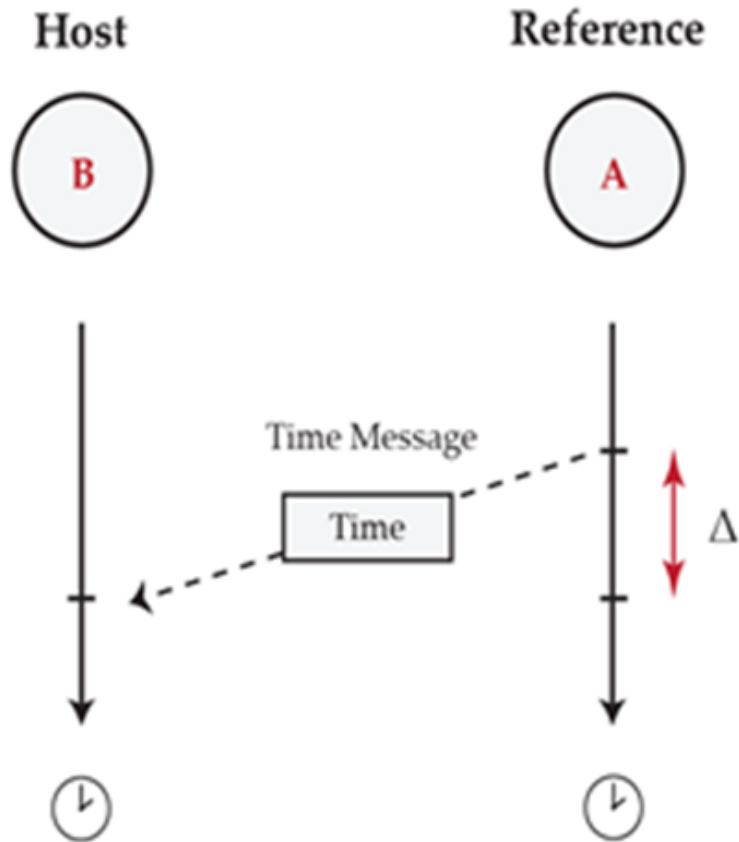
- Such clock(s) provide a time reference to hosts that are interconnected via a network
- Underlying time-synchronisation protocols combine aspects of
 - ▣ Cristian's algorithm, i.e. RTD calculation
 - ▣ Berkely's algorithm, i.e. combining multiple reference time sources
- Good time synchronisation requires
 - ▣ good time references – that's easy (GPS, atomic clocks, etc.)
 - ▣ predictable / symmetric / deterministic network latencies – that's doable in LAN setups, but not guaranteed in Internet data communication
- 2 main protocols
 - ▣ Network Time Protocol (NTP) (www.ntp.org)
 - RFC 5905 (www.ietf.org)
 - Originally Unix-based NTP daemon, now ported to most OS
 - Version 4 standardised in 2010
 - One of the first Internet protocols that evolved
 - ▣ Precision Time Protocol (PTP)
 - IEEE 1588-2008, with most recent update as IEEE 1588-2019 (also called PTP 2.1)
 - Designed for managed networks, e.g. LAN

NTP and PTP Characteristics

4



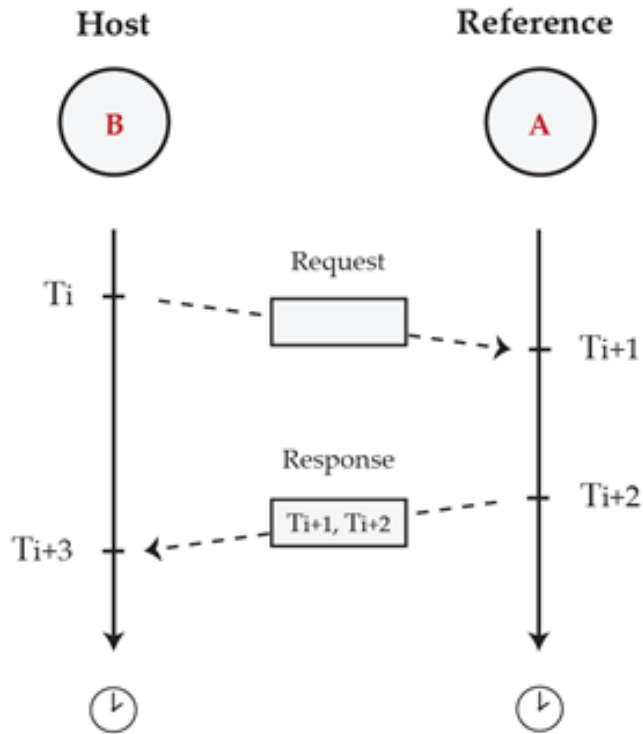
Uni-Directional Synchronisation



- A reference [clock] sends a time stamp to a host via a network
- The host uses the time stamp to set its local clock
- Useful when message latencies are minor relative to synch levels required

Round-Trip Synchronisation (RTS)

6



- A Host request message, followed by a Reference response message, with known (local) submission and arrival times, allow for the calculation of
 - round-trip delay (1)
 - host clock error, i.e. phase offset (2)
- **Variations of RTS form basis of NTP and PTP**

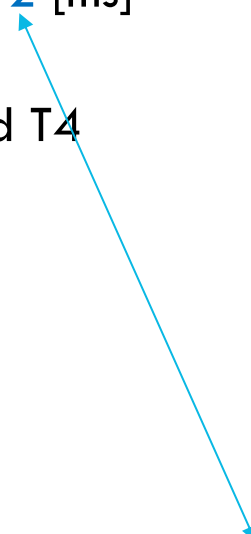
$$(1) \quad \delta = (T_{i+3} - T_i) - (T_{i+2} - T_{i+1})$$

$$(2) \quad \theta = (T_{i+1} - T_i) + (T_{i+2} - T_{i+3})/2$$

δ is eliminated, assuming symmetric uplink-downlink delay

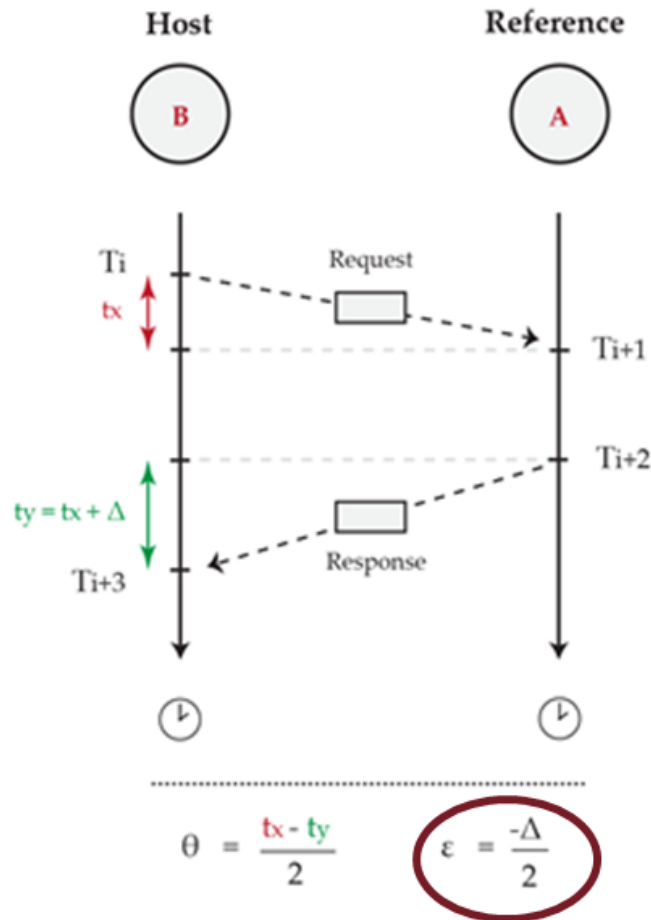
Example RTD and Offset Calculation

7

- Assume
 - the Host clock is +2 [ms] ahead in relation to the Reference clock
 - a one-way delay of 1 [ms], and a RTD of 2 [ms]
 - a Reference processing time of 3 [ms]
 - We name the 4 timestamps T1, T2, T3 and T4
 - Therefore, with T1 = 00:00:00:005
 - T1 = 5 (skip leading 0s)
 - T2 = 5 + 1 - 2 = 4
 - T3 = 4 + 3 = 7
 - T4 = 5 + 1 + 3 + 1 = 10
 - RTD = (T4 - T1) - (T3 - T2) = (10 - 5) - (7 - 4) = 2 [ms]
 - OFF = ((T2 - T1) + (T3 - T4)) / 2 = ((4 - 5) + (7 - 10)) / 2 = ((-1) + (-3)) / 2 = -2 [ms] (that's the Host clock correction required)
- 

Host Clock Synchronisation Errors due to Asymmetric Delays

8



- Asymmetric delays caused by uplink / downlink differences, for example in
 - Different ingress/egress router queues
 - traffic shaping
 - Different routing paths
- Need for
 - tightly engineered, managed and configured networks
→ see PTP
 - “build-in” compensation mechanisms
→ see NTP

Example Incorrect Offset Calculation

9

- Assume
 - ▣ the Host clock is +2 [ms] ahead in relation to the Reference clock
 - ▣ asymmetric delays of 1 [ms] H->R, and 4 [ms] R->H, resulting in a RTD of 5 [ms]
 - ▣ a Reference processing time of 3 [ms]
- Therefore, with T1 = 00:00:00:005
- T1 = 5 (skip leading 0s)
- T2 = 5 + 1 - 2 = 4
- T3 = 4 + 3 = 7
- T4 = 5 + 1 + 3 + 4 = 13
- RTD = (T4 - T1) - (T3 - T2) = (13 - 5) - (7 - 4) = 8 - 3 = 5 [ms] (correct)
- OFF = ((T2 - T1) + (T3 - T4)) / 2 = ((4 - 5) + (7 - 13)) / 2
= ((-1) + (-6)) / 2 = -3.5 [ms] (that's obviously not correct)

NTP

10

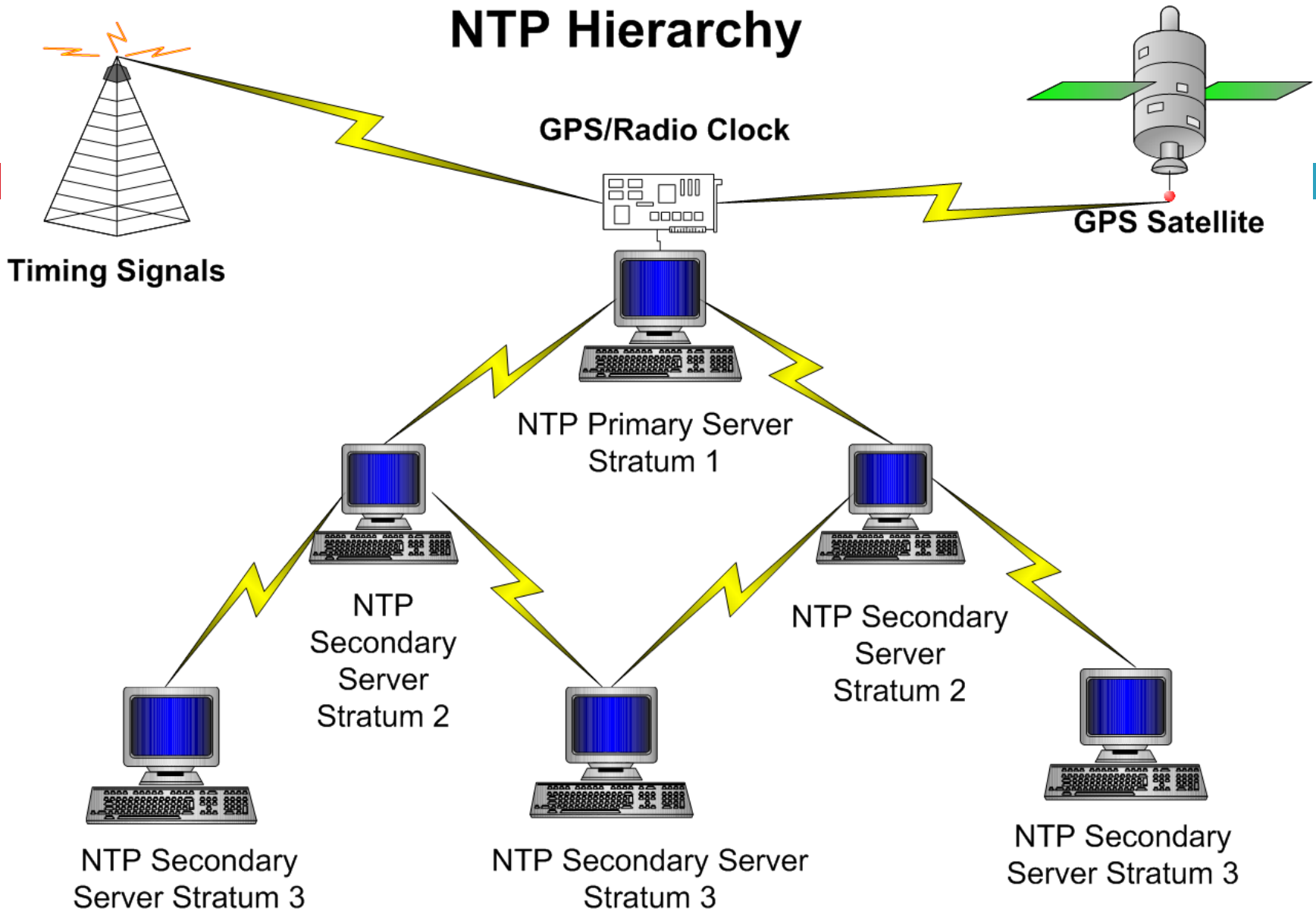
- The NTP architecture, protocol and algorithms have evolved over the last 40 years to the latest NTP Version 4
- **Internet standard protocol** for time synchronisation and coordinated UTC time distribution
- **Fault tolerant protocol** – automatically selects the best of several available time sources to synchronise with
- **Highly scalable** – nodes form a hierarchical structure with reference clock(s) at the top
 - ▣ Stratum 0: Time Reference Source
 - GPS / TAI atomic clocks / DCF 77
 - ▣ Stratum 1: Primary Time Server
- Applies some general principles (as discussed before)
 - ▣ Avoid setting clocks backward
 - ▣ Avoid large step changes
 - Amortise the required change (+/-) over a series of short intervals (e.g. over multiple ticks)

NTP

11

- NTP is the longest running and continuously operating Internet protocol (since around 1979)
- Government agencies in many other countries and on all continents (including Antarctica) operate public NTP primary servers
- National and regional service providers operate public NTP secondary servers synchronised to the primary servers
- Many government agencies, private and public institutions, including universities, broadcasters, financial institutions and corporations operate their own NTP networks

NTP Hierarchy

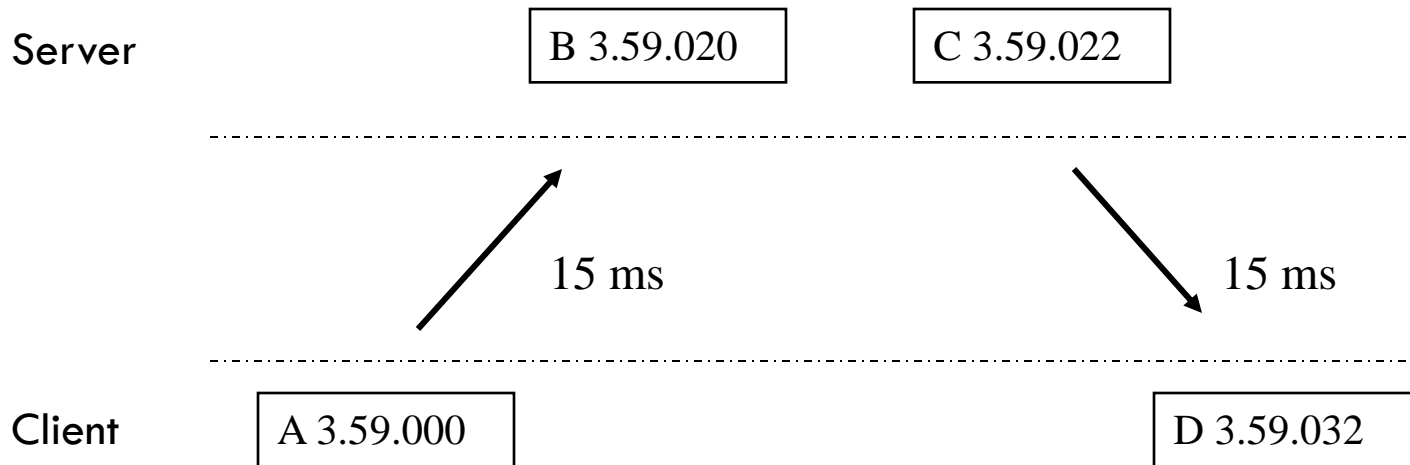


Client / Server Mode

- UDP is used for data transfer (no TCP), i.e., NTP over UDP
 - ▣ UDP port 123
 - ▣ Optionally use of broadcasting or multicasting (not covered here)
- Several packet exchanges between NTP client and Stratum server take place to determine client offset (see also next slide)
 - ▣ Client
 - *Sends packet with originate timestamp **A***
 - ▣ Server receives such a packet and returns response containing A as well as:
 - *receive timestamp **B***
 - *transmit timestamp **C***
 - ▣ Client receives this packet and
 - *processes A, B, C as well as (this) packet arrival time **D***
 - *Determines offset and Round-Trip Delay (RTD)*

NTP Operation

14



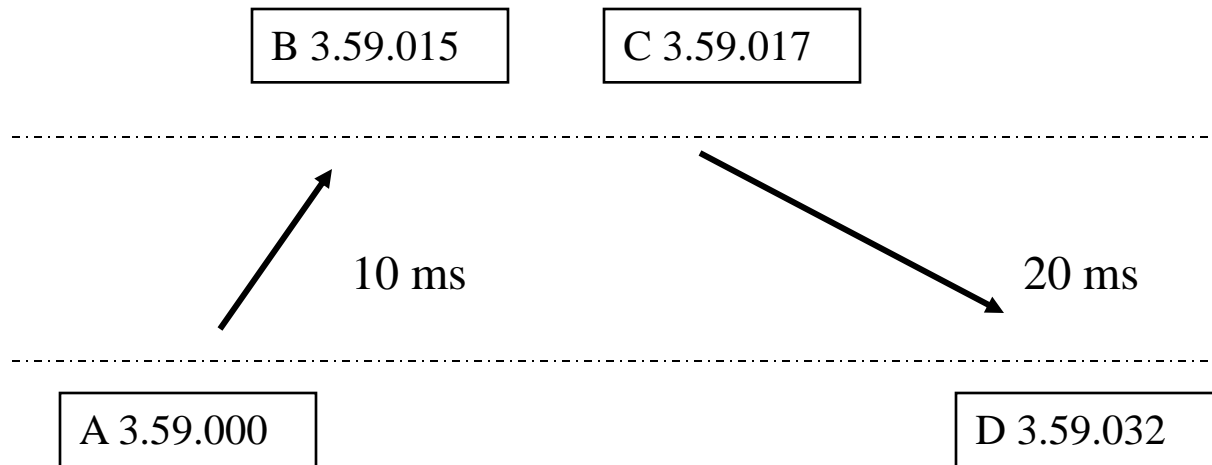
Symmetric Network : 15 ms delay each way

The client's clock lags 5 ms behind the server's clock

$$\text{RTD} = (D - A) - (C - B) = 32 - 2 = \mathbf{30 \text{ msec}}$$

$$\text{Offset} = ((B - A) + (C - D)) / 2 = (20 + (-10)) / 2 = 10 / 2 = \mathbf{5 \text{ msec}}$$

Problem again: Network Delay Asymmetry



The client's clock still lags 5 ms behind the server's clock ...

But there is an asymmetric network latency, i.e., 10 ms vs 20 ms

$$\text{RTD} = (D - A) - (C - B) = 32 - 2 = 30 \text{ msec}$$

$$\text{Offset} = ((B - A) + (C - D)) / 2 = (15 + (-15)) / 2 = 0 / 2 = \mathbf{0 \text{ msec}}$$

Typical NTP Performance

16

- Small LAN
 - ▣ ~10 microseconds best case ever on 2-node LAN
 - ▣ ~220 microseconds on real-world small LAN
- Typical large-building LAN
 - ▣ ~2 ms
- Internet with few hops
 - ▣ 10 – 20 ms
- Long distance and/or slow or busy link
 - ▣ 100 ms – 1 s
- Accuracy further degraded on networks with asymmetric traffic delays
- Some of these configurations are further explored in assignment 1

NTP Time Format

17

- Reference scale is UTC
- Time parameters are 64 bits long:
 - ▣ Seconds since January 1, 1900 (32 bits, unsigned)
 - ▣ Fraction of a second (32 bits, unsigned)
- Dynamic range: 136+ years
 - ▣ rollover in 2036
- Resolution: 2^{-32} seconds ~ 232 picoseconds

NTP Protocol Header

19

- **LI** Leap Indicator: 2-bit
 - 0 = no warning
 - 1 = last minute of the day has 61 seconds
 - 2 = last minute of the day has 59 seconds
- **VN** Version number: 2-bit
 - currently 4
- **Mode**: 3-bit integer, including
 - 3 = client
 - 4 = server
- **Stratum**: 8-bit integer for Stratum server hierarchy level, including
 - 1 = primary server (i.e. stratum 1)
 - 2-15 = secondary server

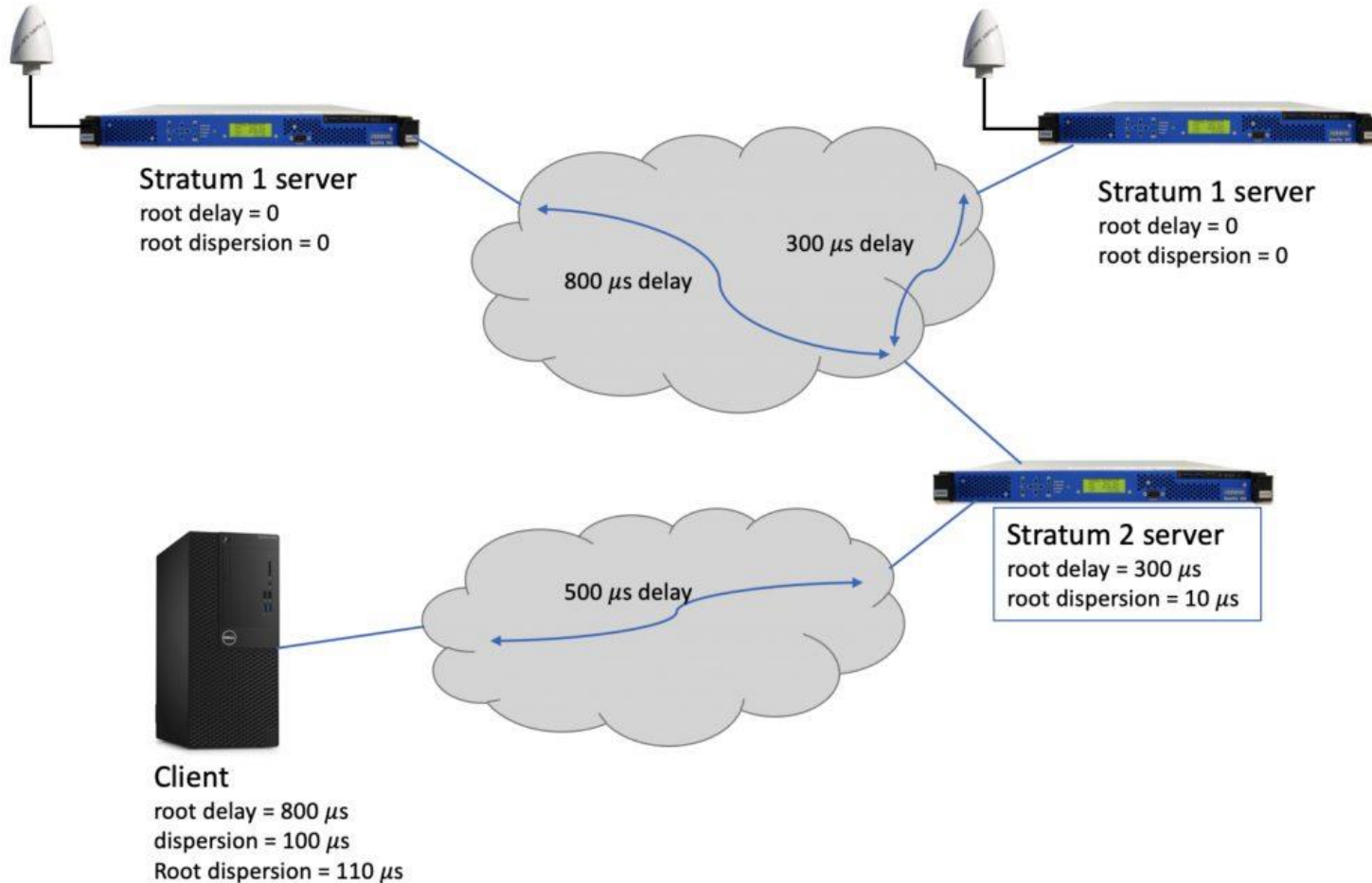
NTP Protocol Header

20

- **Poll:** 8-bit signed integer representing the maximum interval between successive messages, in \log_2 seconds
 - ▣ This field indicates the interval at which the client will poll the NTP server for time updates
 - ▣ The client dynamically adjusts this interval based on its clock's stability and the network conditions to balance accuracy and network load
- **Precision:** 8-bit signed integer representing the resolution of the system clock (the tick increment), in \log_2 seconds
 - ▣ E.g., a value of -18 corresponds to a resolution of about one microsecond, i.e. 2^{-18} seconds
- **Root Delay:** Round-trip packet delay from a client to a stratum 1 server
 - ▣ It gives a crude estimate of the worst-case time transfer error between a client and a stratum 1 server due to network asymmetry, i.e. if all of the round-trip delay was in one direction and none in the other direction

Example Root Dispersion and Root Delay

21



Dispersion, Root Dispersion

22

- For a single client clock the **dispersion** is a measure of how much the client's clock might drift during a synchronisation cycle:
 - ▣ Dispersion = $DR * (D - A) + TS$, with
 - $(D - A)$ = Duration of a synchronisation cycle
 - ▣ A = First timestamp, see previous example
 - ▣ D = 4th (last) timestamp, see previous example
 - DR = Local clock skew, i.e., déviation of actual clock tick frequency to nominal tick frequency
 - TS = Timestamping errors due to the finite resolution of the clock, and delays in reading the clock when fetching a timestamp
- The **root dispersion** of a client' clock is the combined dispersions of all stratum servers along the path to a Stratum 1 server
- The **root distance**
 - ▣ is the sum of root dispersion and half the root delay
 - ▣ provides a comprehensive measure of the maximum error in time synchronization is the total worse case timing error accumulated between the stratum 1 server and the client.

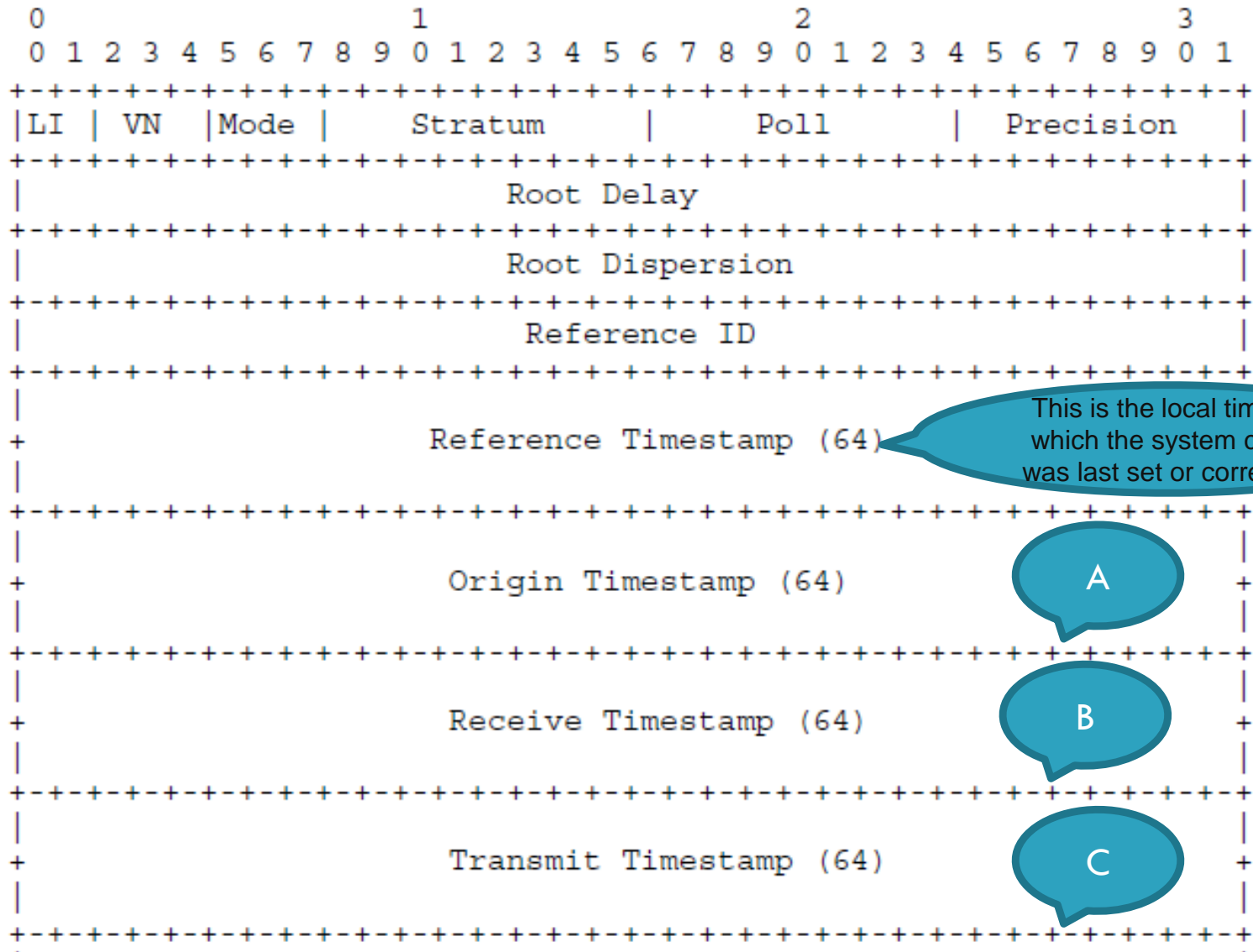
Common Synchronisation Source Time Reference Identifier Codes

23

- **Reference ID** (refid): 32-bit code (4 ASCII bytes) identifying the particular server or reference clock, see examples below

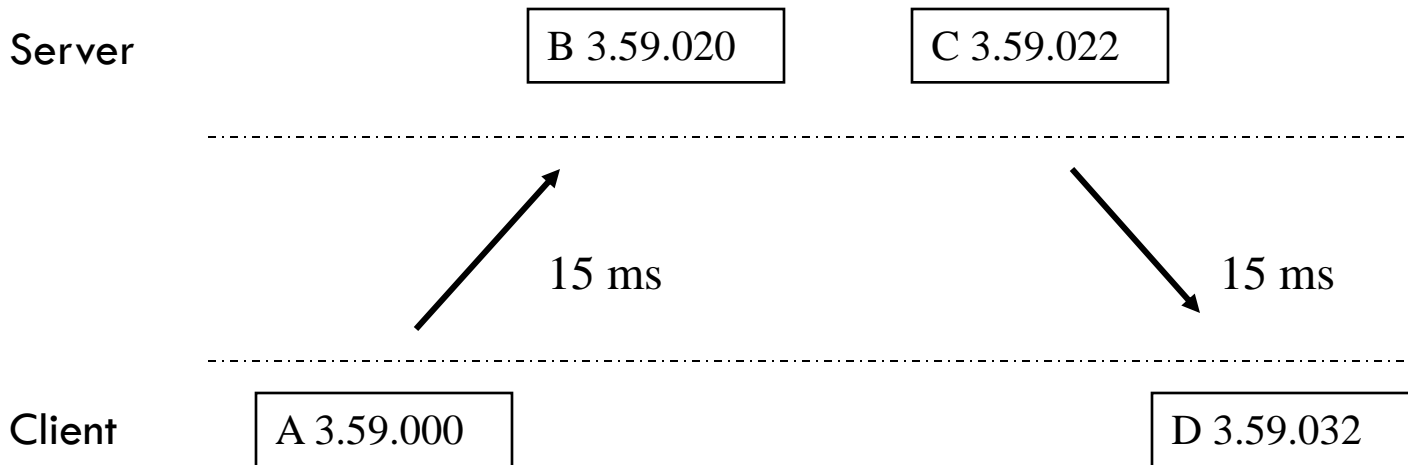
refid	Clock Source
GPS	Global Positioning System
GAL	Galileo Positioning System
PPS	Generic pulse-per-second
DCF	LF Radio DCF77 Mainflingen, DE 77.5 kHz
WWV	HF Radio WWV Fort Collins, Colorado
GOOG	Unofficial Google Refid used by Google NTP servers as time4.google.com

NTP Protocol Header



Recall: NTP Operation

25



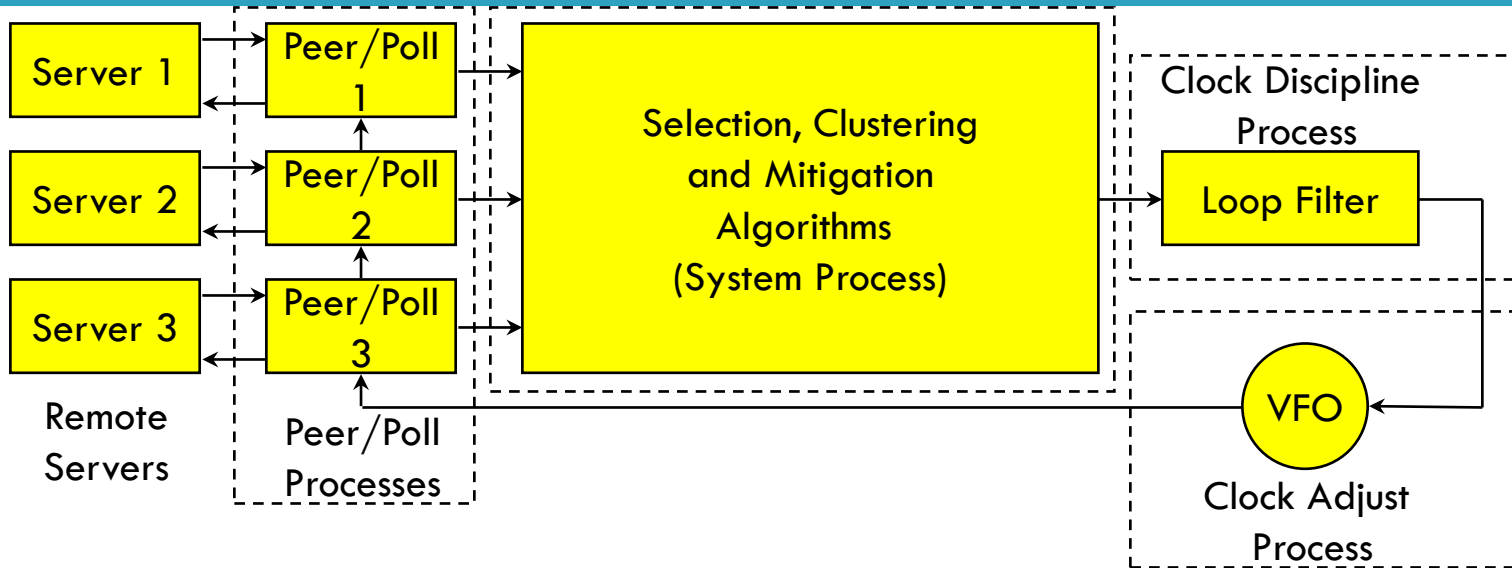
Symmetric Network : 15 ms each way (actual delay)

$$\text{RTD} = (D - A) - (C - B) = 32 - 2 = \mathbf{30 \text{ msec}}$$

$$\text{Offset} = ((B - A) + (C - D)) / 2 = (20 + (-10)) / 2 = -10 / 2 = \mathbf{-5 \text{ msec}}$$

Architectural Overview

26



- An NTP client synchronises with multiple stratum servers
- It uses a range of algorithms to deal with variable and asymmetric non-deterministic network delays and to determine its most likely offset, thereby running a series of processes:
 - ▣ Peer process runs when a packet is received
 - ▣ Poll process sends packets at intervals determined by the clock discipline process and remote server
 - ▣ System process runs when a new update is received
 - ▣ Clock discipline process implements clock time adjustments
 - ▣ Clock adjust process implements periodic clock frequency (VFO) adjustments (FYI only)

NTP Operation Overview

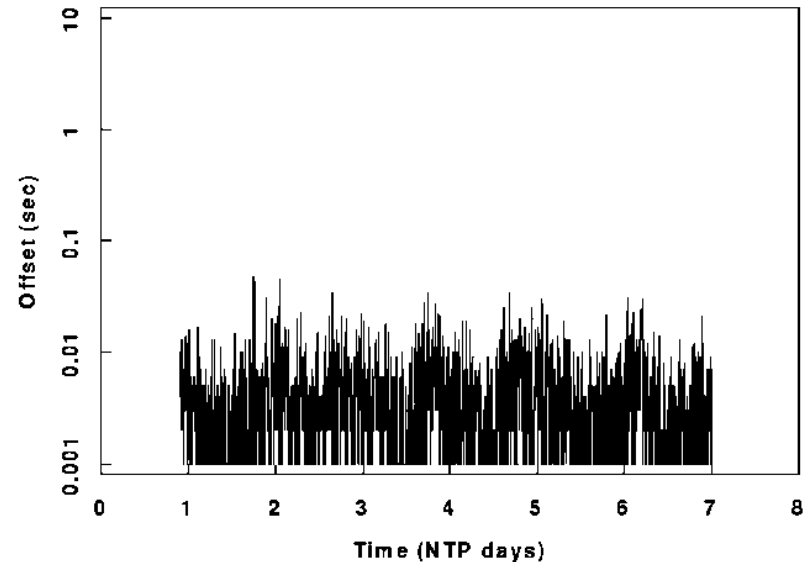
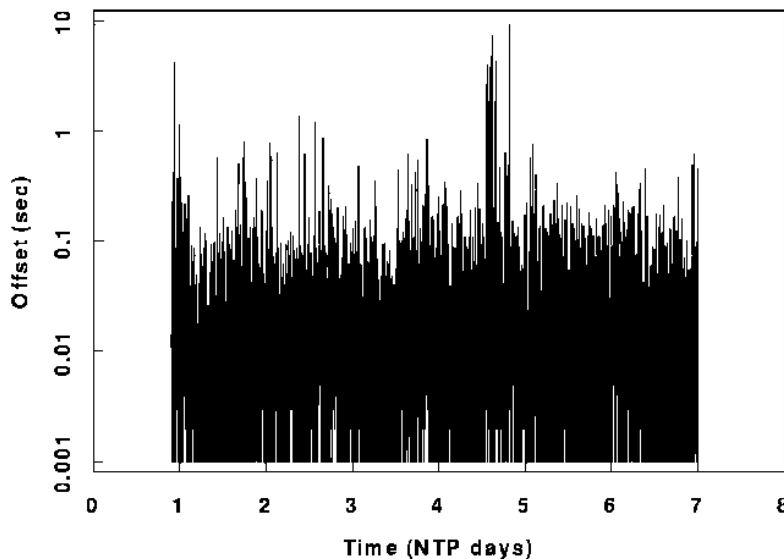
27

- For each stratum server there is a *Poll* process that sends NTP packets at intervals ranging from 8 s to 36 hr
- The corresponding *Peer* processes receive NTP packets and after performing some packet sanity tests, T1 - T4 are determined / extracted
- The NTP daemon calculates offset and delay as seen before
- The time series of offset and delay values calculated by multiple peer processes are processed by a sequence of algorithms
 - ▣ Eliminate servers with long RTD, or servers that show “strange” offsets, which for example are the result of network asymmetries

Clock Filter Algorithm

28

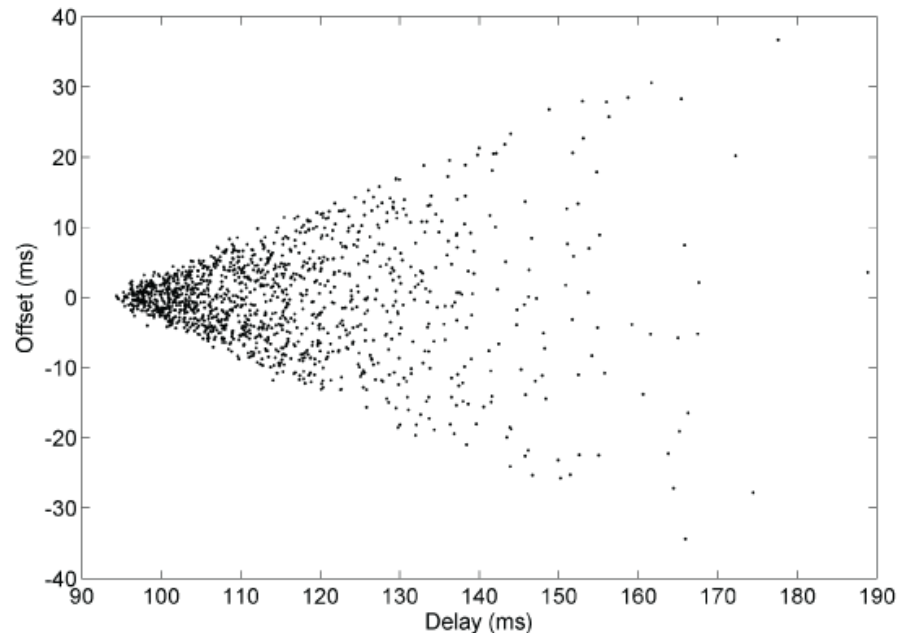
- For each stratum server it uses a sliding window of eight samples and picks out the sample with the least expected error
 - ▣ I.e. choose sample with minimum roundtrip delay (RTD)
 - ▣ Effective at removing spikes resulting from intermittent network congestions



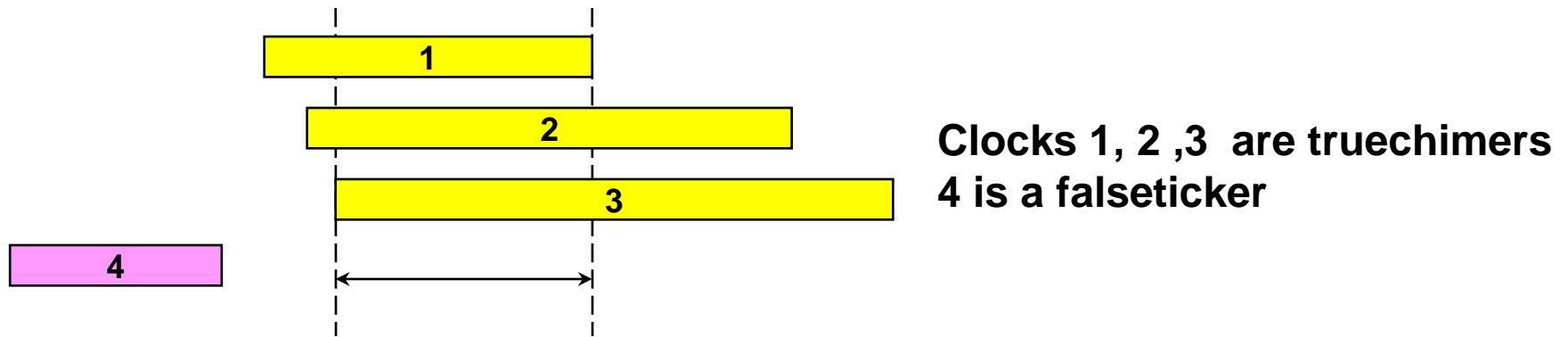
Motivation Clock Filter Algorithm

29

- The wedge scattergram plots sample points of offset versus delay (RTD) collected over a 24-hr period by a client clock communicating with a single stratum server
 - ▣ For this experiment the client clock is externally synced to the stratum server, so the offset should be zero
- However, as the (network) roundtrip delay increases, the offset variability increases, resulting in increasingly larger offset errors
- Therefore, the best samples are those at the lowest delay
- This is taken into account by the clock filter algorithm



Mitigation Algorithms: Intersection Algorithm

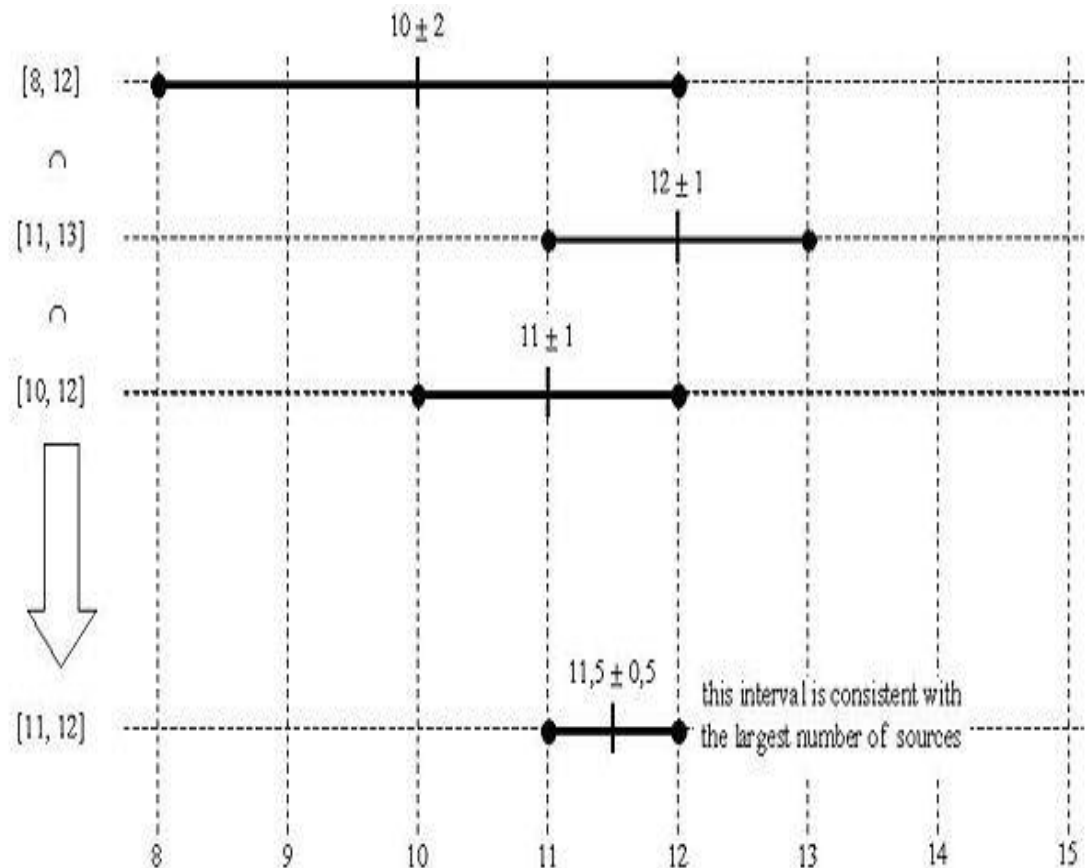


- Selects a subset of peers, i.e. stratum servers
- Based on intersection of confidence (offset) intervals
 - ▣ i.e. min / max offsets of a clock over recent x readings determine its interval
- Identifies truechimers & falsetickers
- Here: plot range of offsets calculated by each peer with 1, 2, 3 overlapping

Based on Marzullo's Algorithm (1984)

31

- Agreement protocol for estimating accurate time from a number of noisy time sources
- If we have offset intervals
- 10 ± 2 , 12 ± 1 , 11 ± 1 , then interval intersection is 11.5 ± 0.5
- If some intervals don't intersect, consider intersection of majority of intervals
- Algorithm eliminates false tickers
- See also https://en.wikipedia.org/wiki/Marzullo%27s_algorithm



FYI: Clustering (Clock Selection) and Combining Algorithm

- This **clock cluster algorithm** processes the truechimers returned by the clock intersection algorithm
- It produces a list of survivors by eliminating truechimers that have a comparably large root delay and root dispersion
- Finally, the **clock combining algorithm** averages the time offsets of survivors using their root dispersions as a weight
 - ▣ I.e., survivors with a small root dispersion have a higher weight

Clock Discipline

- The Combining Algorithm provides a final offset, so the client clock can be adjusted
- Recall (important when setting the clock):
 - ▣ No time reversal (clocks don't go back)
 - ▣ Avoid step changes (clocks do not change time abruptly)
 - ▣ Instead, make the clock ticking slower or faster
- The Unix Clock Model provides the Kernel variable **tickadj**
- It amortises required change gradually by making adjustment every tick, for example every 10 msec
- Example (see last lecture)

```
__interrupt void clock_handler() {
    Master_clock.tv_nsec += CLOCK_TICK_INCREMENT + tickadj;
    while (Master_clock.tv_nsec > ONE_SECOND_IN_NANO_SEC) {
        Master_clock.tv_nsec -= ONE_SECOND_IN_NANO_SEC;
        Master_clock.tv_sec++;
    }
}
```

Outline Assignment 1: NTP Benchmarking

38

- Setup NTP client
- Identify and configure Stratum servers from various geographic areas
- Use `ntpq` to collect NTP offset, delay and jitter concurrently for all chosen time servers over 8-12 hours
- Analyse, compare and contrast data

The ntpq Tool

```
C:\WINNT\System32\cmd.exe - ntpq msc-101

C:\>ntpq msc-101
ntpq> pe
      remote           refid      st t when poll reach  delay  offset  jitter
=====
*GPS_PALISADE(1) .GPS.          0 l   8   32  377   0.000   0.000   0.008
-lanczos.maths.t rackety.udel.ed  2 u  47   64  356  10.327  -14.213  1.365
+ntp2.ja.net      .GPS.          1 u  62   64  377  30.508   -1.123   0.638
-ntp-sop.inria.f  .GPS.          1 u  37   64  377  40.033   -1.932   4.135
-hora.cs.tu-berl .GPS.          1 u  42   64  377  46.887   -2.424   1.072
-ntp1-rz.rrze.un .DCFp.         1 u  50   64  377  60.548   -1.225   1.255
+swisstime.ee.et .DCFa.         1 u  61   64  377  40.088   -0.720   0.970
-ntp2.ptb.de     .PTB.          1 u  32   64  377  48.403   -1.474   0.522
ntpq> _
```

Time difference

```
C:\WINNT\System32\cmd.exe - ntpq msc-101

C:\>ntpq msc-101
ntpq> pe
      remote           refid      st t when poll reach  delay  offset  jitter
-----
*GPS_PALISADE(1) .GPS.          0 l   8  32  377   0.000   0.000   0.008
-lanczos.maths.t rackety.udel.ed  2 u  47  64  356  10.327 -14.213   1.365
+ntp2.ja.net      .GPS.          1 u  62  64  377  30.508  -1.123   0.638
-ntp-sop.inria.f .GPS.          1 u  37  64  377  40.033  -1.932   4.135
-hora.cs.tu-berl .GPS.          1 u  42  64  377  46.887  -2.424   1.072
-ntp1-rz.rrze.un .DCFp.         1 u  50  64  377  60.548  -1.225   1.255
+swisstime.ee.et .DCFa.         1 u  61  64  377  40.088  -0.720   0.970
-ntp2.ptb.de     .PTB.          1 u  32  64  377  48.403  -1.474   0.522
ntpq> _
```

Server Details

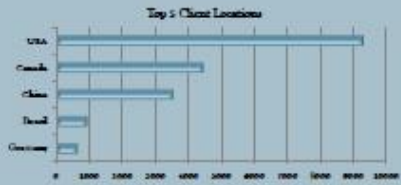
- **(st)**ratum level, with 0 == Stratum 1 server
- **(t)**ypes: l = local (e.g. GPS), u = unicast (i.e. networked)
- **when**: number of seconds since last response
- **poll**: interval in seconds between queries
- **reach**: Reachability in octal, e.g.
 - 11111111 = 377_8 = max
 - 11101110 = 356_8 → last + 5th probe lost
- **Symbol prefix of server name**
 - * : Synch Source – survivor with smallest dispersion
 - + : other candidates included in final combination algorithm
 - - : Discarded by clustering algorithm
 - x : Falseticker identified by intersection algorithm
- **delay**: RTD
- **jitter**: A measure of the variability in the time offset between the client and the server; it indicates the stability of the clock offset over time

ntp-galway.hea.net

43



- GPS receiver
- DCF 77 Radio
- Operational since 2002
- > 2,000,000 different clients
- 50,000 requests per hour



NTP Server

Discipline of Information Technology

NUI Galway

Jonathan Skerrett
jskerrett@nui-galway.ie

Dr. Hugh Mahon
hugh.mahon@nui-galway.ie



This map of the world presents a visualisation of the client base of the Network Time Protocol (NTP) server (ntp-galway.hca.nct) located in the Discipline of Information Technology NUI Galway.

The Network Time Protocol is a protocol designed to synchronise the clocks of computer systems connected via variable latency, dynamic networks.

As of 2009 the number of individual clients requesting time from the NTP server stands at just under 25,000 while the number of requests handled by the server per hour is roughly 62,000.

