MEINBERG SYNC ACADEMY

NTP Key Features and Concepts

Martin Burnicki

www.meinberg.academy

2020-07-16

Agenda

- Time Scales, Time Signals, and Expected Accuracy
- NTP Overview and Key Concepts
- Network Time Packet Sequence
- System Time Synchronization by *ntpd*
- NTP Server Design
- *ntpq -p* Command
- Redundancy Methods in NTP Networks
- LANTIME MRS Feature
- LANTIME Cluster Feature

Time Scales - UTC and TAI

- UTC (Universal Time, Coordinated) is primary time standard of the world
- Officially formalized in 1970, used since 1972
- Successor of GMT (Greenwich Mean Time) which was directly derived from the Earth rotation -> variable length of a second
- UTC is based on the SI second -> atomic time, constant length of a second
- UTC uses leap seconds to stay in synch with the Earth rotation
- Atomic clocks from several institutes around the world contribute to UTC

- TAI (Temps Atomique International) is the high-precision atomic standard time used as the basis for UTC: monotonic, no leap seconds
- Whenever a leap second is inserted, UTC falls behind TAI by 1 s more
- TAI is the default time scale used by PTP/IEEE1588

- UTC = TAI - 37 s (Initial 10 s + 27 leap secs since 1972)

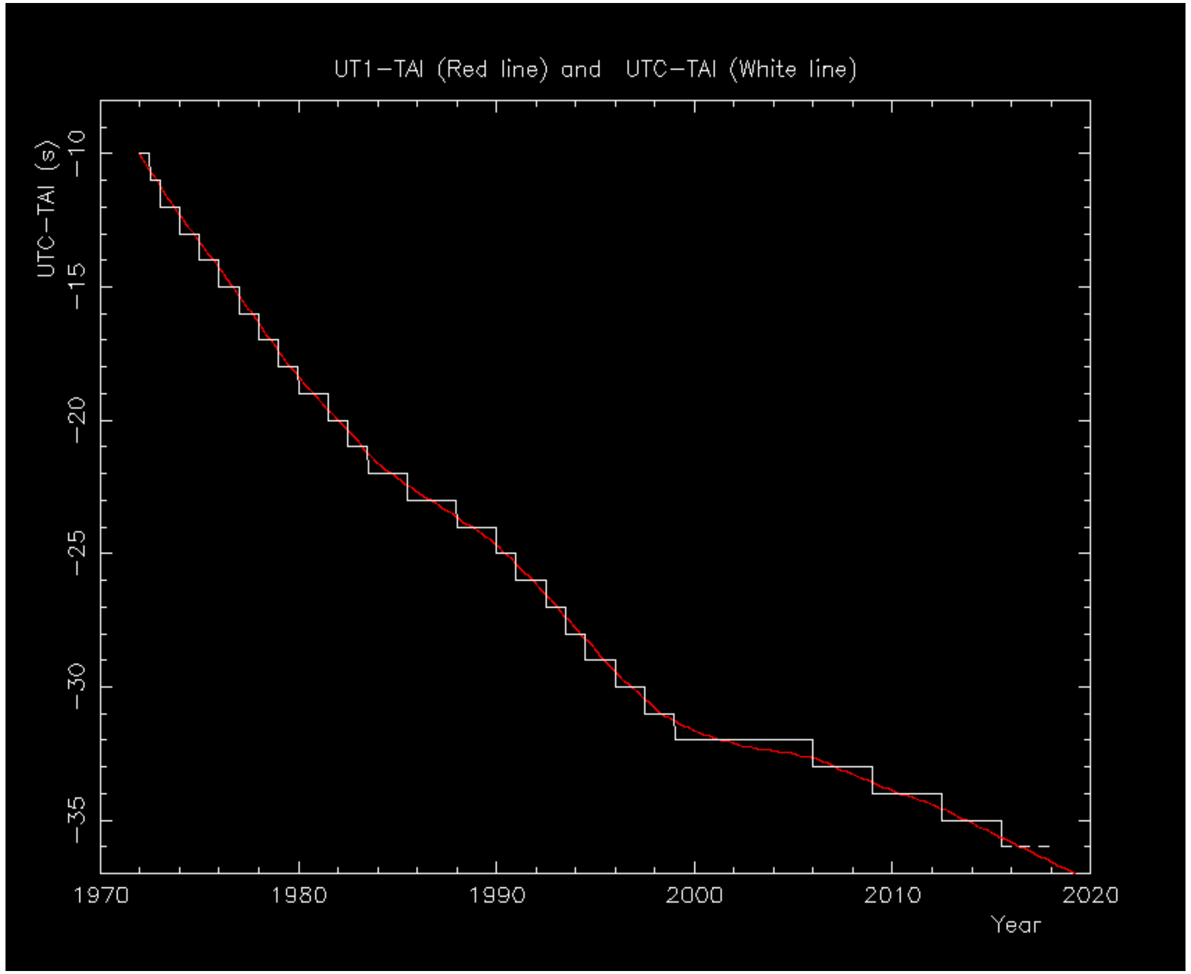
Time Scales - GPS vs. UTC and TAI

- GPS system time is linear like TAI, no leap seconds
- Initial GPS system time was set to match UTC in 1980, which was 19 s behind TAI at that time -> GPS system time is constantly behind TAI by 19 s
- UTC is behind GPS by a variable number of seconds, e.g.:

$$\text{UTC} = \underbrace{\text{TAI} - 19\text{ s}}_{\text{GPS Time}} - \underset{\searrow \text{ Num. of leap secs since 1980}}{18\text{ s}}$$

- At UTC midnight, TAI and GPS time are already ***after*** midnight.
- GPS satellites provide current TAI offset and leap second information, so receivers can derive UTC from GPS system time

Leap Seconds to Adjust UTC



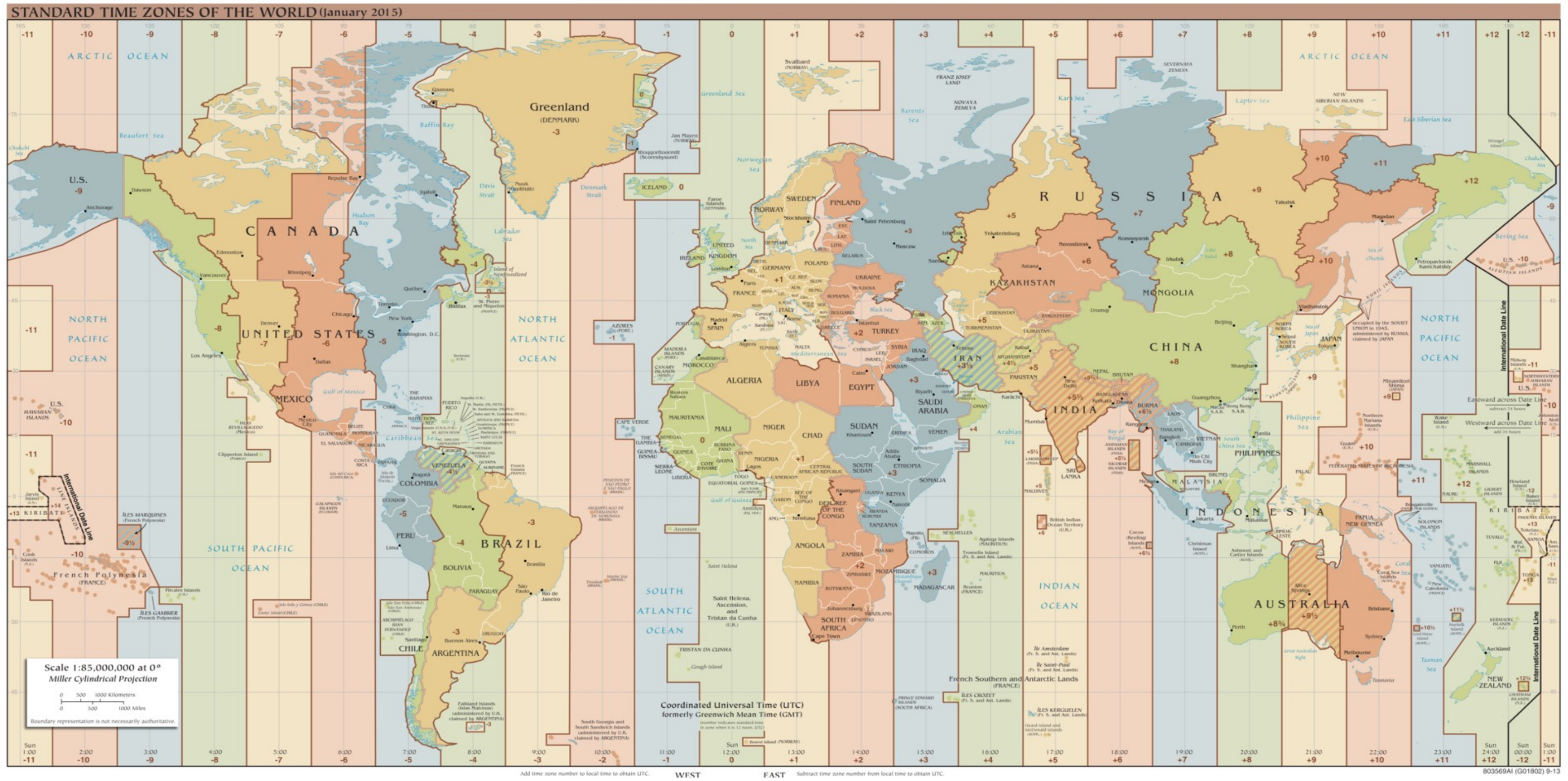
Source:
International Earth Rotation Service
http://hpiers.obspm.fr/eop-pc/earthor/utc/leapsecond.html

Local Time and Daylight Saving Time (DST)

- Local Standard Time is derived from UTC by applying a constant offset
- Optionally switching to/from daylight saving time (DST, Summer Time) by applying an additional offset
- Time zone and rules for DST are determined by legislation of individual countries
- In some countries the rules are changed frequently by the government, other countries have fixed rules for many years
- Protocols like NTP and PTP usually use UTC or TAI, so don't care about local time
- Time zone info distributed by OS vendors, e.g. using IANA TZDB or registry entries

Time Zones

- Time Zones defined geographically at slots of approx. 15 degrees of longitude -> 360° / 24 h
- Offsets from UTC are normally an integer number of hours, up to +/- 12 hours
- Some exceptions, e.g.: +8:45 h, +9:30 h, +12:45 h, +13:00 h, +14:00 h
- Definition of local time and DST switching is subject to legislation of countries


STANDARD TIME ZONES OF THE WORLD (January 2015)
Coordinated Universal Time (UTC) formerly Greenwich Mean Time (GMT)
Scale 1:85,000,000 at 0°
Miller Cylindrical Projection
WEST
EAST
International Date Line

Source: Wikipedia

Accuracy of Different Time Sources

- GPS/GNSS Receiver : ~ 50 ns
- IRIG DCLS +/- 1µs, IRIG AM +/- 10 µs
- PZF: 20 - 50 µs between devices located close to each other
- NTP: depending on network properties:

Internet / WAN:	< +/- 20 ms typical, usually better
LAN:	< +/- 2 ms typical, usually better
	< +/- 1 ms possible in controlled LAN

- PTP (IEEE1588): depending on network and device capabilities:

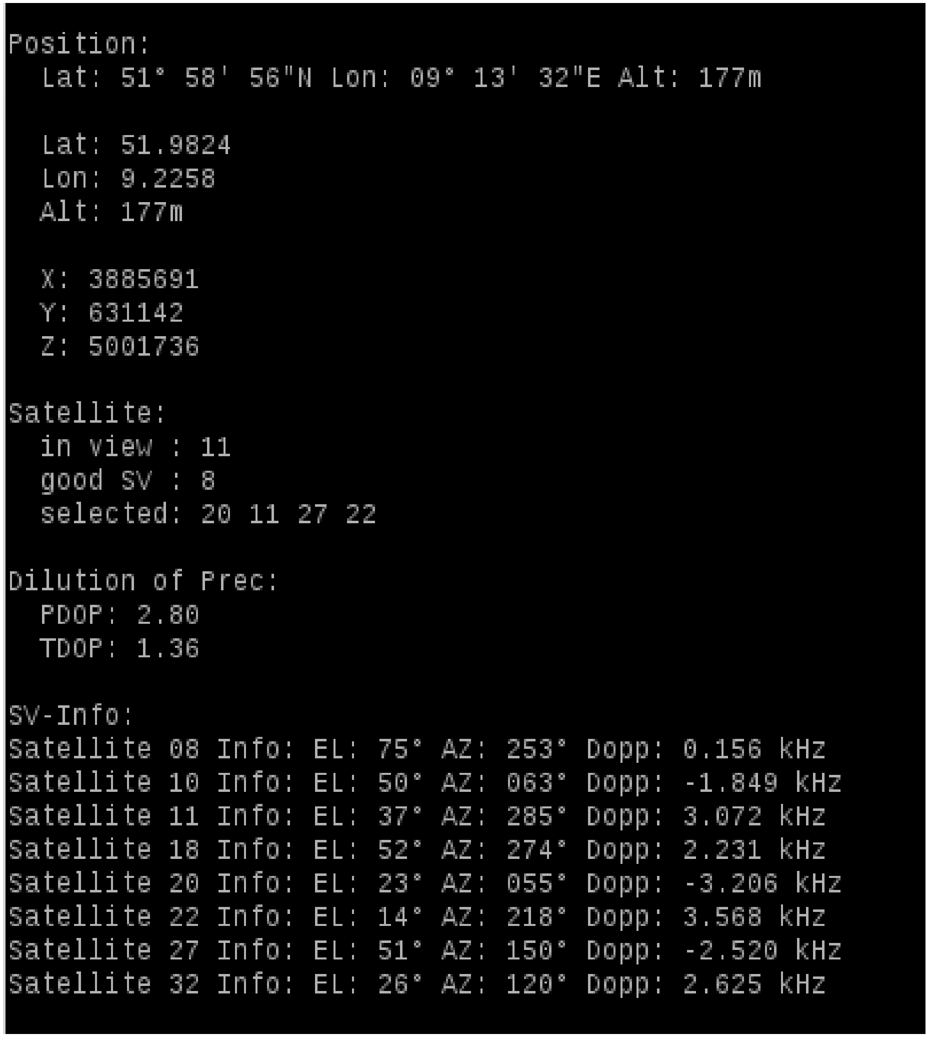
LAN:	< +/- 250 ns typical
	< +/- 50 ns if appropriate infrastructure

- Accuracy depends strongly on infrastructure, implementation, configuration, etc., especially even at the **client** side.
- Degradation e.g. due to overloaded network connections

GPS Basics

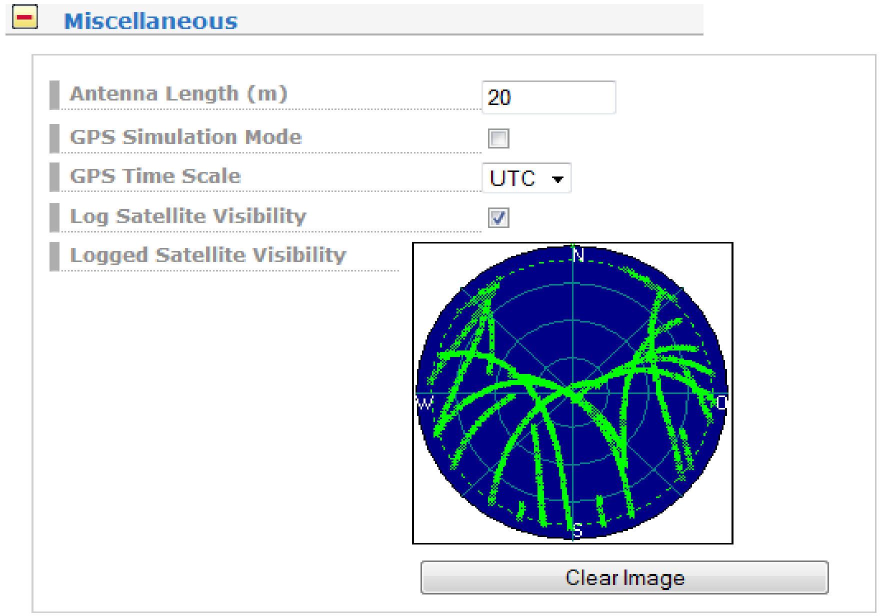
- 32 possible GPS satellites with an atomic clock on board
- At least four satellites have to be received simultaneously to be able to determine both the position (x,y,z) and current time (t)
- Accurate receiver position is required to be able to compensate signal delay
- GPS system time is linear: no leap seconds, no time zones
- GPS/UTC offset and leap second information broadcast by the satellites
- UTC = GPS - number of leap seconds since 1980 (e.g. 18 s in 2019)
- Other GNSS systems (GLONASS, Galileo, Beidou, etc.) work in a similar way
- Usage of several GNSS systems in parallel increases reliability

GPS Receiver Status



```
Position:
  Lat: 51° 58' 56"N Lon: 09° 13' 32"E Alt: 177m

  Lat: 51.9824
  Lon: 9.2258
  Alt: 177m

  X: 3885691
  Y: 631142
  Z: 5001736

Satellite:
  in view : 11
  good SV : 8
  selected: 20 11 27 22

Dilution of Prec:
  PDOP: 2.80
  TDOP: 1.36

SV-Info:
Satellite 08 Info: EL: 75° AZ: 253° Dopp: 0.156 kHz
Satellite 10 Info: EL: 50° AZ: 063° Dopp: -1.849 kHz
Satellite 11 Info: EL: 37° AZ: 285° Dopp: 3.072 kHz
Satellite 18 Info: EL: 52° AZ: 274° Dopp: 2.231 kHz
Satellite 20 Info: EL: 23° AZ: 055° Dopp: -3.206 kHz
Satellite 22 Info: EL: 14° AZ: 218° Dopp: 3.568 kHz
Satellite 27 Info: EL: 51° AZ: 150° Dopp: -2.520 kHz
Satellite 32 Info: EL: 26° AZ: 120° Dopp: 2.625 kHz
```

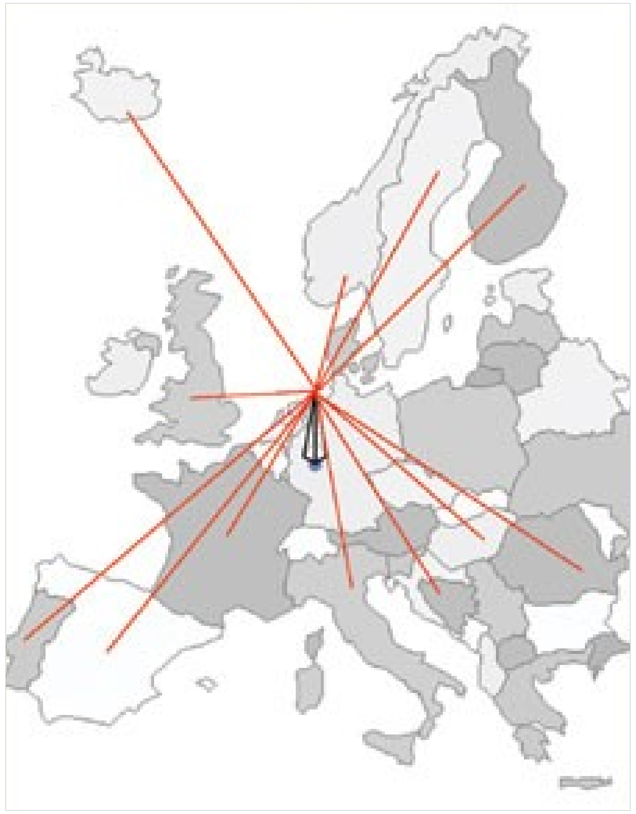
Satellite Visibility in LANTIME Web GUI



Miscellaneous

Antenna Length (m)	20
GPS Simulation Mode	☐
GPS Time Scale	UTC
Log Satellite Visibility	✓
Logged Satellite Visibility	

DCF77 Long Wave Signal



- Long Wave DCF77 Transmitter in Mainflingen near Frankfurt, Germany
- Disseminates the legal time of Germany, controlled by atomic clocks at PTB, Braunschweig
- CET / CEST: Centre European (Summer) Time
- Carrier frequency 77.5 kHz
- Reception up to more than 1200 km distance
- PZF is an additional pseudo-random noise phase modulation, accuracy as good as about 50 µs
- Signal propagation delay not constant

NTPv4 Network Time Protocol - Overview

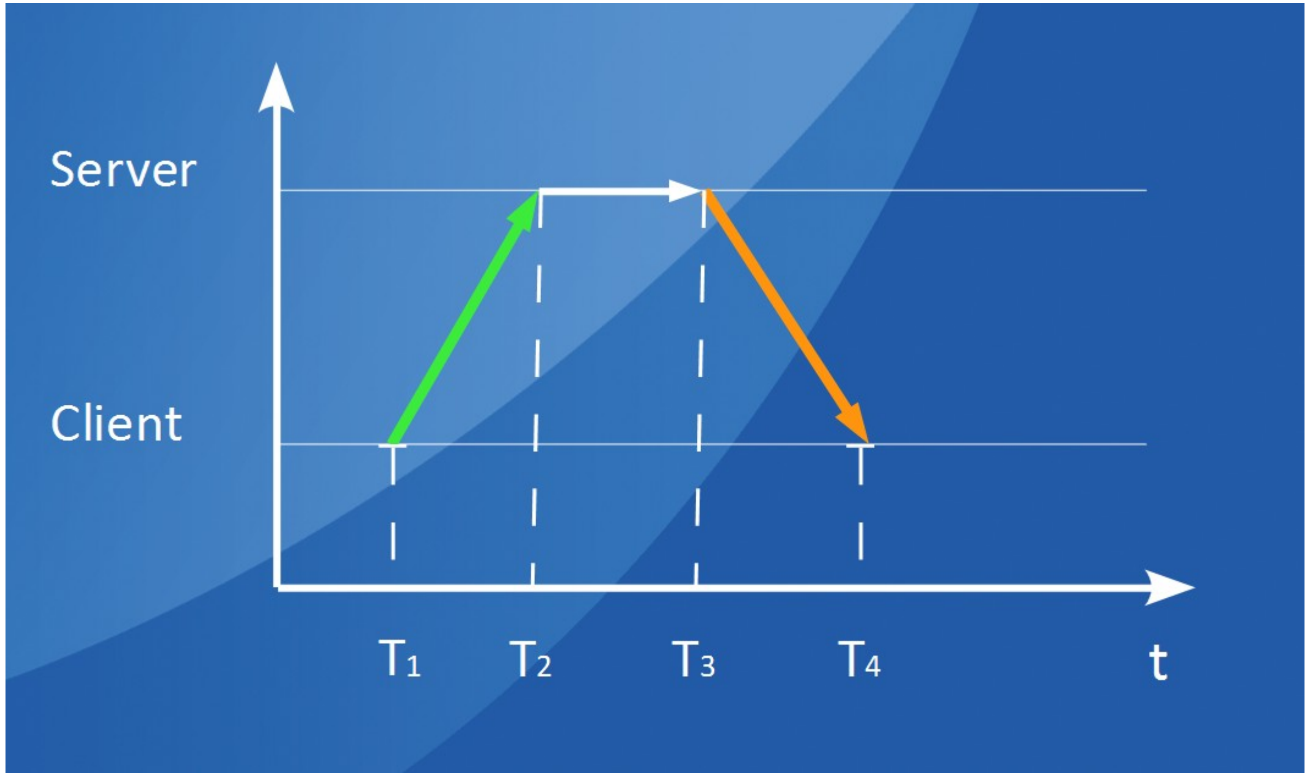
- Developed in the 1980s by Prof. David L. Mills
- One of the oldest and most widely used network protocols
- By now the 4th version published
- Normally uses UTC time, no information on time zones, no daylight saving time, etc.
- UDP based transport protocol, IPv4 and IPv6, standard port 123
- **Distinguish** between the **protocol** and **implementations**, i.e. client or server software
- *ntpd* is the reference implementation from ntp.org, open source
- *ntpd* is free of charge, standardized, and regularly updated
- *ntpd* can achieve high precision time even on standard hardware
- *ntpd* is available for different OS (Unix, Windows, ...)
- *ntpd* has integrated redundancy and security
- *ntpd* adjusts system time smoothly, without time leaps
- SNTP refers to simplified implementations of NTP, uses same packet format

NTP - How does it work?

- NTP Clients request time information from an NTP Time Server
- NTP servers send their current time and status to clients in reply
- Clients accept the time information from a server only if the server is synchronized
- Unlike in telecommunications, NTP stratum numbers just indicate a hierarchy level in the time synchronization chain, not a specific accuracy class.
- Reference time sources like GPS/GNSS receivers represents the highest stratum level, stratum 0
- Stratum-1 servers have direct access to such a reference clock
- NTP client of a stratum-*n* server can become stratum-(*n*+1) server
- *ntpd* is a powerful client and server:
  - Reliability: statistic filtering to determine and compensate network delays
  - Redundancy: can query several sources, detects “falsetickers” and prefers alternate time sources, if available

NTP Packet Exchange

NTP assumes symmetric network packet delay: a trip from Client to Server takes the same time as back from server to client.



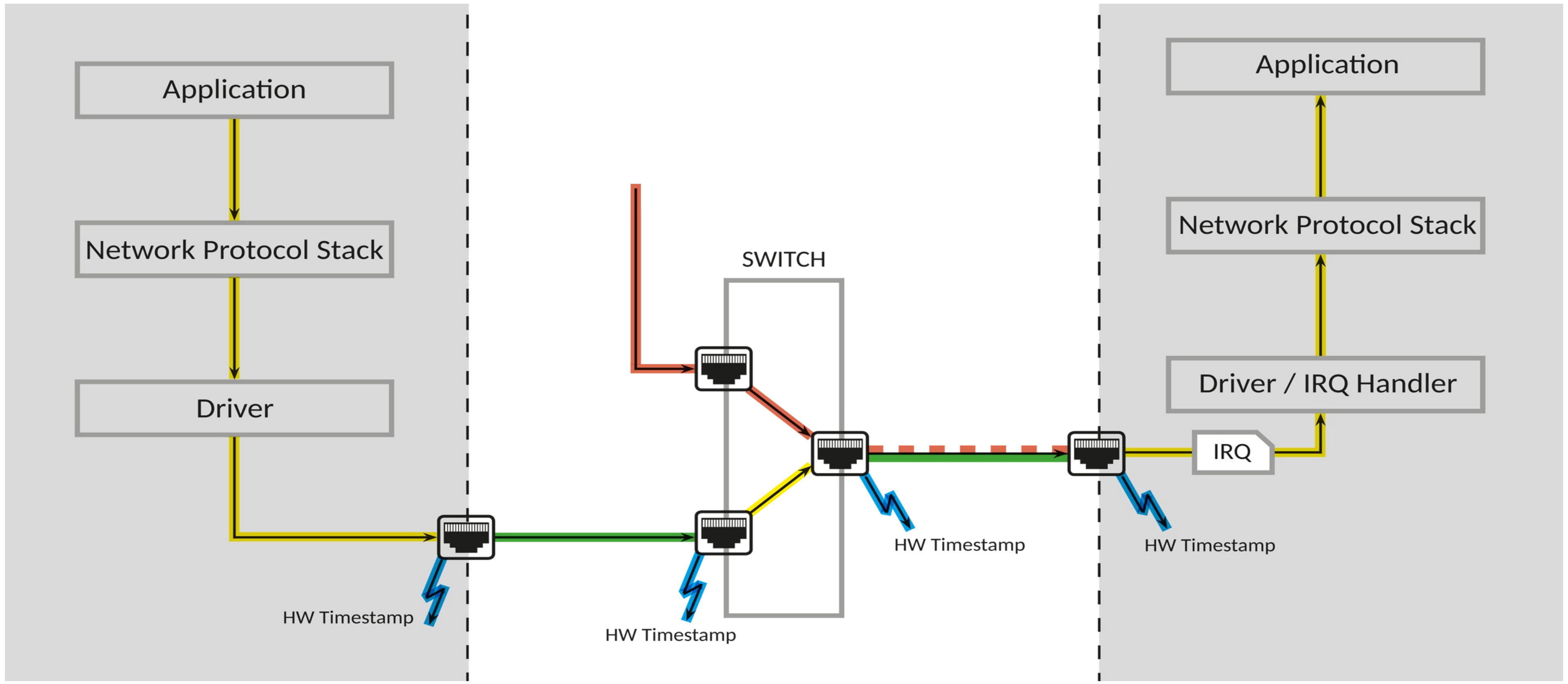
T_1-> Transmit TS, client time
T_2-> Receive TS, server time
T_3-> Transmit TS, server time
T_4-> Receive TS, client time

Roundtrip Delay: $(T_4-T_1)-(T_3-T_2)$

1-way Delay = Roundtrip / 2

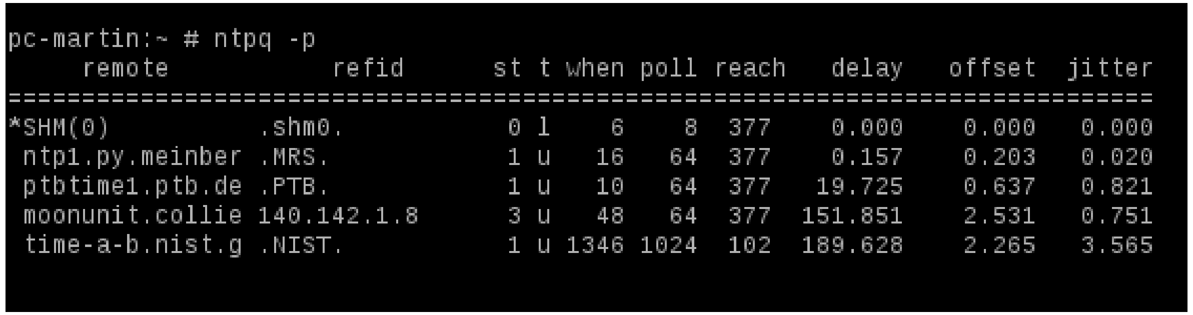
Time Offset = $\frac{(T_2-T_1)+(T_3-T_4)}{2}$

Network Packet Delays



Delay, Jitter, Offset, Accuracy

- Command ***ntpq -p*** is used to monitor the synchronization status of *ntpd*



```
pc-martin:~ # ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*SHM(0)          .shm0.           0 l    6    8  377    0.000    0.000   0.000
 ntp1.py.meinber .MRS.            1 u   16   64  377    0.157    0.203   0.020
 ptbtime1.ptb.de .PTB.            1 u   10   64  377   19.725    0.637   0.821
 moonunit.collie 140.142.1.8      3 u   48   64  377  151.851    2.531   0.751
 time-a-b.nist.g .NIST.           1 u 1346 1024  102  189.628    2.265   3.565
```

st	stratum
t	type of clock source (local, unicast)
when	time since the last request
poll	polling interval
reach	status of eight last pollings
delay	estimated packet delay
offset	est. time difference between client and server
jitter	→ deviation of offset / delay

- *SHM0* is GPS PCI card used as refclock
- *ntp1* is a LANTIME device on the local network
- *ptbtime1* is one of the PTB servers on the internet: **offset < 1 ms despite ~20 ms packet delay**
- *moonunit* is a random public server in the U.S.: **offset ~2.5 ms despite ~150 ms packet delay**
- *time-a-b* is a public NIST server in the U.S.: **offset ~2.5 ms despite 190 ms packet delay** and limited reachability

Unicast, Broadcast, Multicast, Manycast

- NTP uses **unicast UDP request/reply** packets by default

- **Broadcast** from server is supported, but:
  - Broadcast packets ***not forwarded*** across subnet boundaries

- **Multicast** servers:
  - Well-known multicast port assigned for NTP by IANA
  - Unlike broadcast packets, multicast packets ***can be forwarded*** to other subnets -> TTL

- **Manycast**: Server discovery scheme, based on multicast
  - Clients send multicast packets with a low TTL
  - If no reply, they send again with increased TTL
  - TTL increased, until "enough" servers have been discovered

- **Problem** of broadcast and multicast:
  - Reduced accuracy, clients are unable to determine and compensate the network delay
  - Any node can send broadcasts, so **authentication should be used**

- Which modes are supported depends on implementation; *ntpd* supports all

NTP Authentication

- ***NTP authentication*** does ***not*** encrypt packets, nor is it used to decide whether a client is allowed to query the time. ***Restrictions*** are used for the latter.

- NTP authentication means that **signatures are appended to packets** so that the ***client*** can be sure the NTP packet it receives really originates from the configured server, and has not been spoofed or modified by a man in the middle.

- Original approach was ***symmetric keys*** since NTP v3, still supported by NTP v4.

- NTP v4 introduced an ***autokey feature*** which uses private/public key pairs

- Today *autokey* is considered ***too weak***, so a new **Network Time Security** (NTS) protocol has been developed by a working group which includes crypto experts.

- NTS is available, implementation in standard NTP software is pending

- Authentication is also required for remote configuration

- None of the above is compatible with packet signatures supported by ***w32time***

ntpd Access Restrictions

- ***ntpd*** can be configured to restrict specific access to particular IP addresses or subnets
- Different restrictions are possible for querying current time, querying the current status (e.g. *ntpq -p*), etc.
- Restrictions are defined in the configuration file, default restrictions are defined by OS or package maintainers
- Appropriate restrictions could also be used to prohibit using *ntpd* for denial-of-service attacks.
- Detailed description of restrict options available at http://doc.ntp.org/current-stable/accopt.html#restrict

ntpd in the Role of a Client

- Can query the time from different time sources, e.g. NTP servers and/or refclocks. Evaluates, classifies and weights the replies from all configured sources, and selects the “best” time source as **system peer**, marked by a **'*'** or **'o'** in the output of *ntpq -p.* The stratum of the selected system peer determines the stratum of the client.

- Remaining good time sources are called ***candidates***, marked with a **'+'** character. Candidates can become a new *system peer* if the current *system peer* becomes unavailable. ***Falsetickers*** are detected, discarded, and marked with a **'-'** character.

- Polling interval by default selected automatically, 64s to 1024 s. Only in special cases the polling interval should be explicitly defined.

- System time is ***slowly slewed*** if the computed time offset is below the ***step threshold*** (128 ms by default), and ***stepped*** if the offset to be corrected is above this threshold.

- If time offset exceeds the ***panic gate*** (1000 s) during operation, *ntpd* terminates itself.

- If a **leap second** has been scheduled, *ntpd* passes the announcement to the operating system (except Windows), and **the OS kernel handles the leap second**.

ntpd in the Role of a Server

- *ntpd* maintains an internal status that is sent to clients.

- After startup the leap bits are '11' and the stratum is 16 -> not synchronized
  ->Clients don't accept a server in this state

- After server (in role of a client) has synchronized, the leap bits change to '00', and the stratum becomes the stratum of the *system peer* + 1.
  -> Clients consider the server as reachable in this state

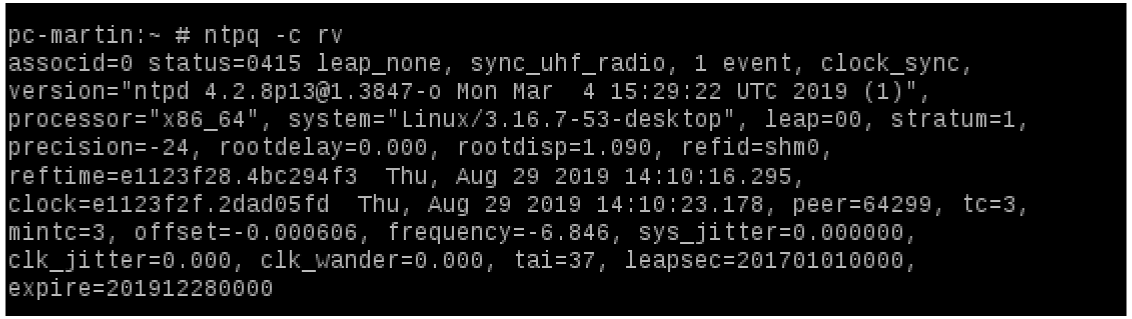
- If the server's time sources become unavailable, *ntpd* does ***not*** set the leap bits back to '11', and does ***not*** change its stratum back to 16 by default. Instead, the server's ***root dispersion*** starts increasing slowly over time.

- **Clients** can detect the increasing root dispersion, know this server is drifting, and can decide:
  - Switch to a different server, if a better one is available
  - Continue accepting this server, if no better one is available

- For dumb clients it is possible to let the stratum change when the server's time sources become unavailable, but this is ***not recommended***.

The Internal Status of *ntpd*

The command ***ntpq -c rv*** can be used to check the internal status of *ntpd*:



```
pc-martin:~ # ntpq -c rv
associd=0 status=0415 leap_none, sync_uhf_radio, 1 event, clock_sync,
version="ntpd 4.2.8p13@1.3847-o Mon Mar  4 15:29:22 UTC 2019 (1)",
processor="x86_64", system="Linux/3.16.7-53-desktop", leap=00, stratum=1,
precision=-24, rootdelay=0.000, rootdisp=1.090, refid=shm0,
reftime=e1123f28.4bc294f3  Thu, Aug 29 2019 14:10:16.295,
clock=e1123f2f.2dad05fd  Thu, Aug 29 2019 14:10:23.178, peer=64299, tc=3,
mintc=3, offset=-0.000606, frequency=-6.846, sys_jitter=0.000000,
clk_jitter=0.000, clk_wander=0.000, tai=37, leapsec=201701010000,
expire=201912280000
```

- The output shows version numbers and status information including:
  - The leap bits ('00' in the example above)
  - The stratum number (1 in the example above)
  - The root dispersion
  - Leap second file information
  - Leap smearing information, if configured

Tasks of the NTP Server Software

The task of an NTP server is “easy“:

- Record the arrival time (T_2) of the request packet received from a client
- Check if there is a restriction that forbids sending a reply packet, check an optional signature, etc.
- Add the transmit time (T_3) and status information to the packet and send it back to the client as reply

→ There is no way for the NTP server to figure out how long the request packet from the client has been traveling before it arrived, or how long it takes until the reply packet arrives at the client. Only the client can possibly determine and account for these delays!

Tasks of the NTP Client Software

Task of an NTP Client is more complex:

- Send a time request with an originate time (T_1) to an NTP Server
- Wait for a reply
- Get a system time stamp when the reply packet arrives (T_4)
- **Evaluate** the packet and its information

Evaluation at the client side:

- Calculate the network delay and time offset to the server
- Consider values from several time requests/replies
- Identify and eliminate network spikes
- Calculate system time corrections
- Apply system time correction according to options available under the specific OS

NTP and Leap Seconds

- Leap second announcements can be propagated via the NTP protocol, so clients can receive an announcement from their server(s).

- Servers need a reliable source of leap second information, which can be:
  - A **reference clock**, if the time signal provides the information (e.g. GPS), and if the protocol of the refclock supports this (the parse driver and SHM driver used by Meinberg both do).
  - A **leap second file** as provided by IERS or NIST. Leap second files also provide a TAI offset, but have an expiration date, so the file should always be updated when a new version becomes available.

- If the local system clock of the OS supports this, *ntpd* just passes leap second announcements to the OS kernel, and **the OS kernel handles the leap second** when appropriate.

- Only very current versions of Windows really support leap seconds, if explicitly configured, but there is no public API call which can be used to pass the information to the kernel. For older Windows versions *ntpd* provides a workaround which slews the system time quickly to account for an inserted leap second.

- *ntpd* also supports leap second smearing, which is only useful for very special cases.

Leap Second Problems

- After UTC was invented in 1970 a leap second was inserted nearly every 1 or 1.5 year, and developers were familiar with leap seconds.
- From 1999 up to 2006 there was a 7-year period without any leap second, so some developers didn't have leap seconds on their mind, and when another leap second occurred in 2006 there were a couple of problems, e.g. some Linux kernels had a deadlock.
- A general problem is that most operating systems just step the time back by 1 s to insert a leap second, and applications (e.g. databases) get confused if the system time is stepped back.
- This is a problem of the operating systems and applications, not of the time synchronization software which just forwards the announcement to the kernel.
- **Leap second smearing** has become popular to avoid having the time stepped back, but smearing means that the **system time is not accurate during the smear interval**.

NTP in various Operating Systems

High resolution of the OS system time required for accurate calculation of packet delays and time offsets.

Linux, BSD and other Unix-like Systems:

- High time resolution (micro or even nanosecond)
- Good API to apply small time corrections
- Leap Seconds are supported within the OS

→ good time disciplining is feasible

Windows:

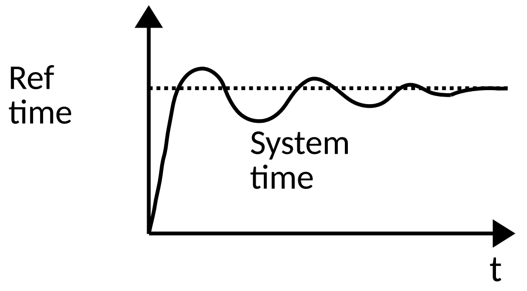
- Time resolution depends on Windows version, partly limited
- ca. 16 ms resolution in versions until Windows XP / Server 2003
- 1 ms resolution in Vista / Win 7 / Server 2008
- 100 ns resolution in Win 8 / Server 2012 and later
- Microsoft's w32time NTP implementation only accurate on very current Win 10 / Server 2016 / Server 2019 versions

System Time Adjustment

UTC only!

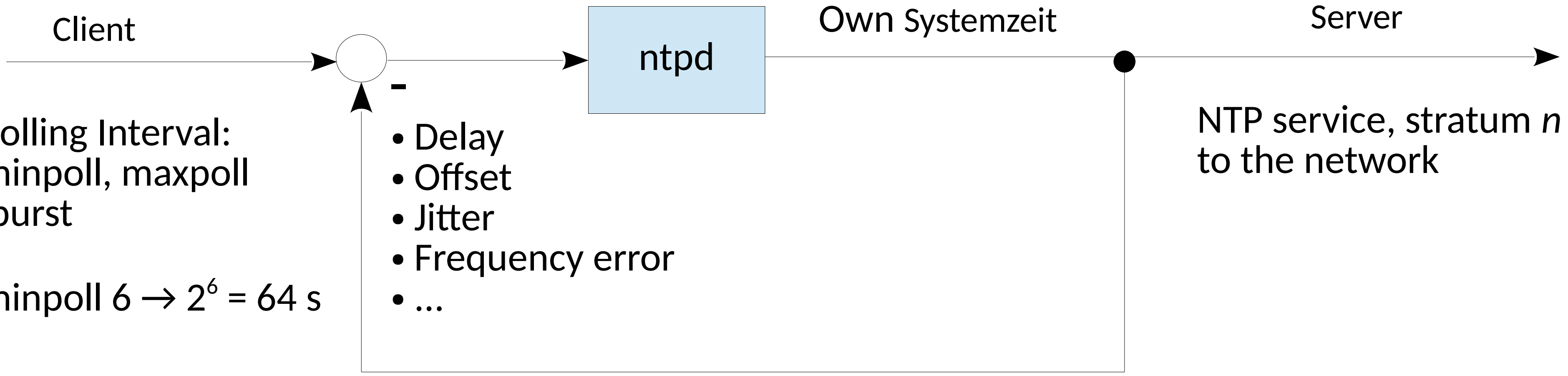
Ref. Time:

- e.g. GPS / GNSS
- DCF77
- TCR
- Stratum *n-1* server



System Time:

- LANTIME
- Windows time
- LINUX



- Delay
- Offset
- Jitter
- Frequency error
- …

Polling Interval:
minpoll, maxpoll
iburst

minpoll 6 → 2^6 = 64 s

NTP service, stratum *n* to the network

System Clock Drift Compensation

- Oscillator frequency varies more or less with ambient temperature
  → Stability, Quartz, TCXO, OCXO, Rubidium, …

- Frequency drift compensation by adjusting the oscillator frequency, if possible

- Two different approaches to compensate clock drift:
  1. Small time steps → frequency fix, number of ticks per seconds varies
  2. Frequency shift → number of ticks per second constant, frequency varies more or less

- "Best" approach depends on the application

- Duration of a correction depends on level of variation:
  - Small corrections result in long duration
  - Short duration requires larger corrections

Resource Requirements

The number of possible clients depends on the hardware performance (CPU power) of the NTP server and the available network bandwidth.

- 64000 clients send 1 request every 64 s --> 1000 req/s
- 2 NTP packets (request and reply) per poll --> 2000 packets/s
- Size of an NTP packet < 128 bytes --> 2000 * 128 bytes = 256000 bytes/s = 2.048 Mbit/s
- Network load ~2 % @ 100 Mbit link speed, ~0.2 % @ Gbit link speed, **for 64000 clients**

Meinberg LANTIME:

- Up to 6000 or 400000 req/s, depending on LANTIME model
- Reduction through additionally running services (e.g. monitoring)
- Reduction if authentication is used

Ports that need to be opened in the firewall:

- 123/UDP for NTP
- 443/TCP for HTTPS (optional)
- 22/TCP for SSH (optional)
- 53 UDP/TCP for DNS (optional)
- 161/TCP for SNMP (optional)
- 162/TCP for SNMP Traps (optional)

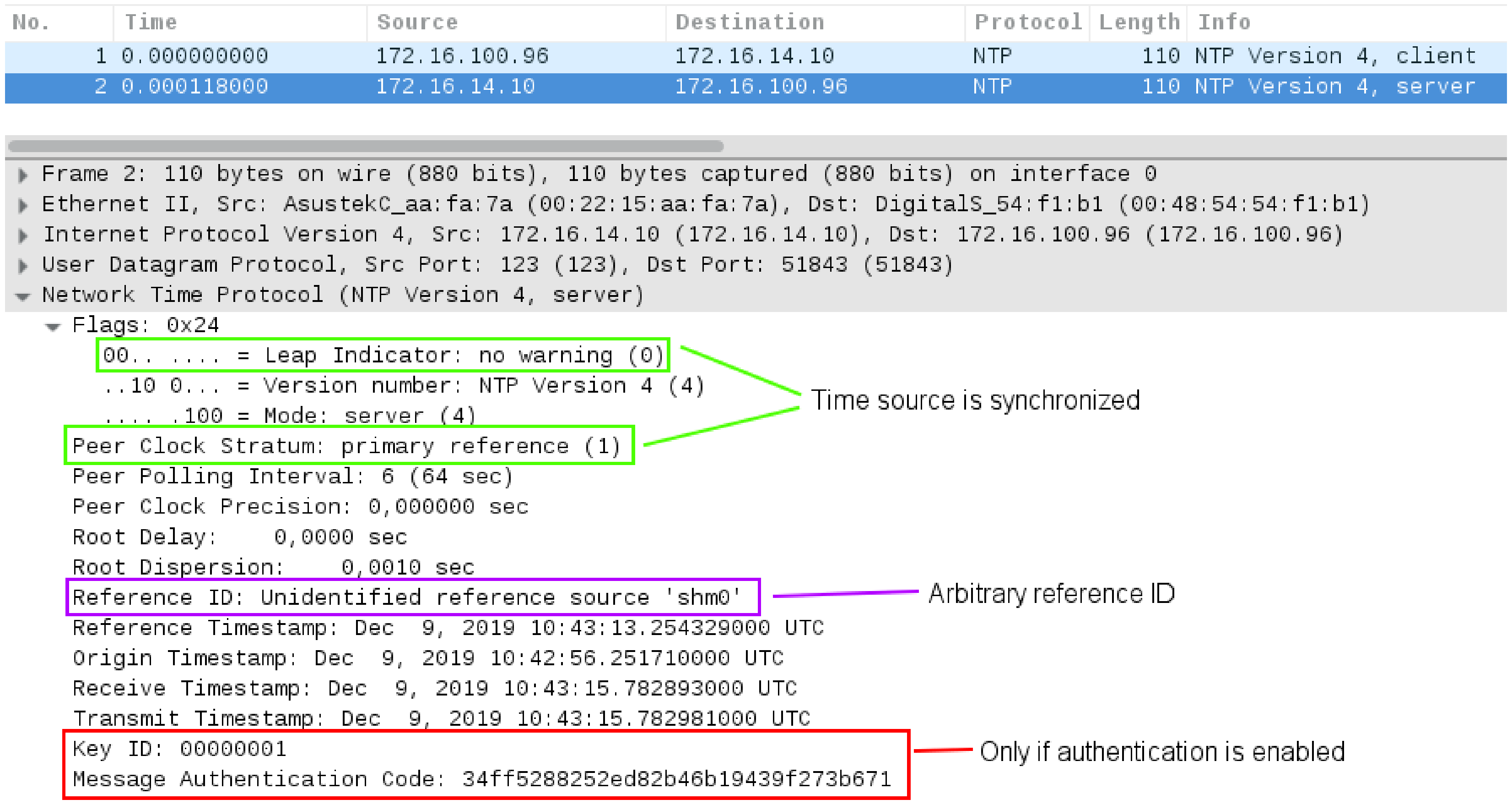
Using external NTP Servers

- Basically, external servers (pool servers, PTB / NIST servers) can be used
- External servers can “overvote” internal time sources
- Polling external servers with intervals less than 2^6 / 64 s is considered abusive

- In the past there have been a number of abuses:
  - Hard-coded IP addresses or hostnames in low cost routers
  - Requests or retries once per second by simple client implementations

NTP Packets in Wireshark

Wireshark: an utility program to inspect network packets



```
No.   Time          Source          Destination     Protocol Length Info
    1 0.000000000   172.16.100.96   172.16.14.10    NTP         110 NTP Version 4, client
    2 0.000118000   172.16.14.10    172.16.100.96   NTP         110 NTP Version 4, server

▸ Frame 2: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
▸ Ethernet II, Src: AsustekC_aa:fa:7a (00:22:15:aa:fa:7a), Dst: DigitalS_54:f1:b1 (00:48:54:54:f1:b1)
▸ Internet Protocol Version 4, Src: 172.16.14.10 (172.16.14.10), Dst: 172.16.100.96 (172.16.100.96)
▸ User Datagram Protocol, Src Port: 123 (123), Dst Port: 51843 (51843)
▾ Network Time Protocol (NTP Version 4, server)
  ▾ Flags: 0x24
      00.. .... = Leap Indicator: no warning (0)
      ..10 0... = Version number: NTP Version 4 (4)
      .... .100 = Mode: server (4)
    Peer Clock Stratum: primary reference (1)
    Peer Polling Interval: 6 (64 sec)
    Peer Clock Precision: 0,000000 sec
    Root Delay:    0,0000 sec
    Root Dispersion:    0,0010 sec
    Reference ID: Unidentified reference source 'shm0'
    Reference Timestamp: Dec  9, 2019 10:43:13.254329000 UTC
    Origin Timestamp: Dec  9, 2019 10:42:56.251710000 UTC
    Receive Timestamp: Dec  9, 2019 10:43:15.782893000 UTC
    Transmit Timestamp: Dec  9, 2019 10:43:15.782981000 UTC
    Key ID: 00000001
    Message Authentication Code: 34ff5288252ed82b46b19439f273b671
```

Time source is synchronized

Arbitrary reference ID

Only if authentication is enabled

Redundancy with NTP

- Full-featured NTP clients can evaluate time from several servers
- Falsetickers are detected and discarded
- When a time source becomes unavailable, a different one is automatically selected
- Usage of redundant reference time sources (e.g. GPS, GLONASS, PZF)
- Redundant modules in devices: redundant receivers, redundant power supplies, redundant network interfaces, bonding
- MRS feature of the LANTIME devices
- Cluster feature of the LANTIME devices for simple NTP clients

MRS – Multi Reference Source

- Specific feature of the Meinberg LANTIME servers with redundant input sources as reference signal
- Available for LANTIME M300 / M400 / M600 / M900 models and the new IMS series
- A highly stable disciplined oscillator (OCXO HQ)
- Automatic switchover between sources based on availability and quality
- The same range of output signals as GPS models (e.g. 1 PPS, 10 MHz, IRIG DCLS / AM, Serial Time Strings, Programmable Pulses, ...)
- For all MRS models, GPS / GNS, NTP, IRIG DCLS / AM, 1 PPS, 10 MHz can be configured as reference source
- Optional reference sources: PTP/IEEE 1588, E1/T1telecom signals, longwave radio signals (DCF77 / MSF / WWVB, ...)
- Selection and switchover of time source by the receiver module, only single signal passed to *ntpd*

MRS – How does it work?

- Configurable priority of the available input signals, e.g.:
  GPS -> NTP
  GPS -> IRIG -> NTP
  PPS -> IRIG -> NTP

- System oszillator (preferably OCXO HQ) disciplined by selected reference signal

- When the selected reference signal fails, the next available signal is selected after a delay interval that depends on the precision of the next time source
  → IRSA algorithm

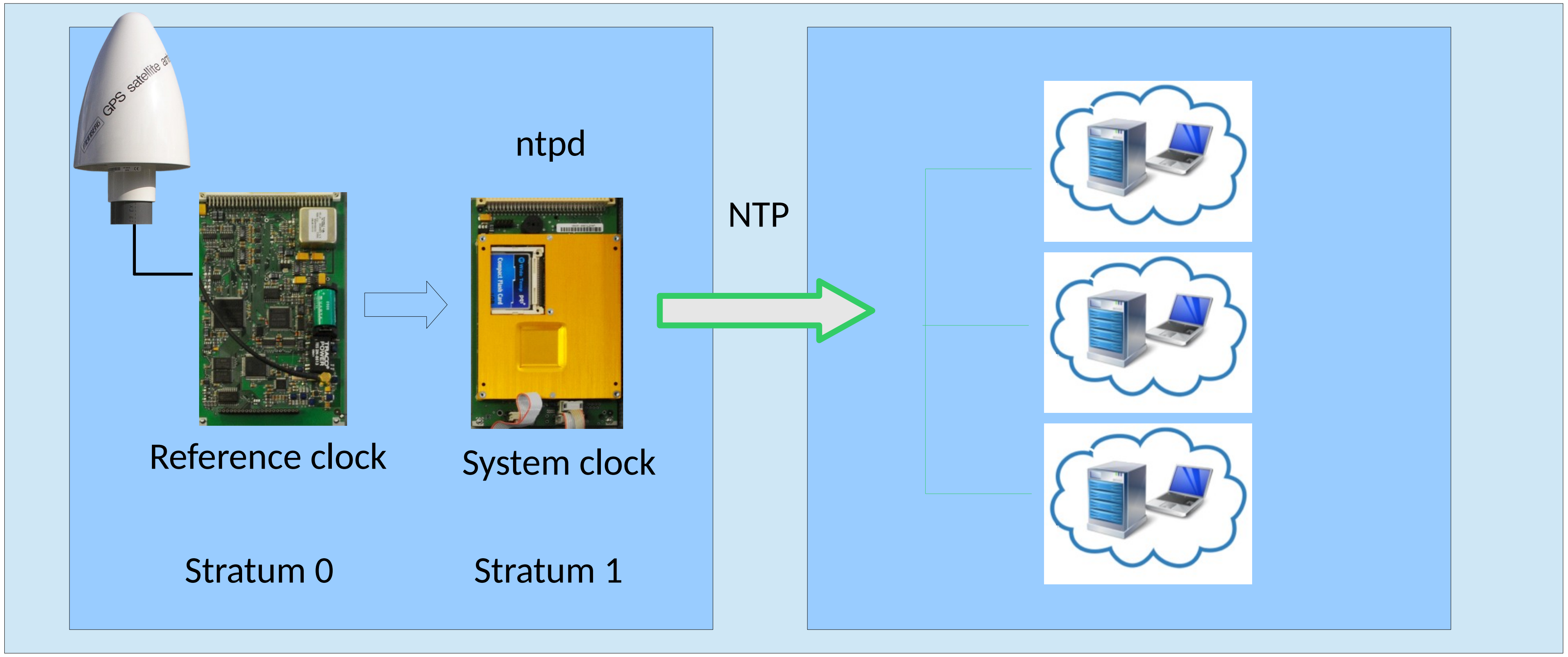
- MRS mode / IRSA example:
  GPS (precision: 50 ns)
  IRIG (precision: 10 µs)
  NTP (precision: 1 ms)

- The system will change to IRIG after a short interval, but switch to NTP as input signal only after a much longer interval

The Clustering Feature

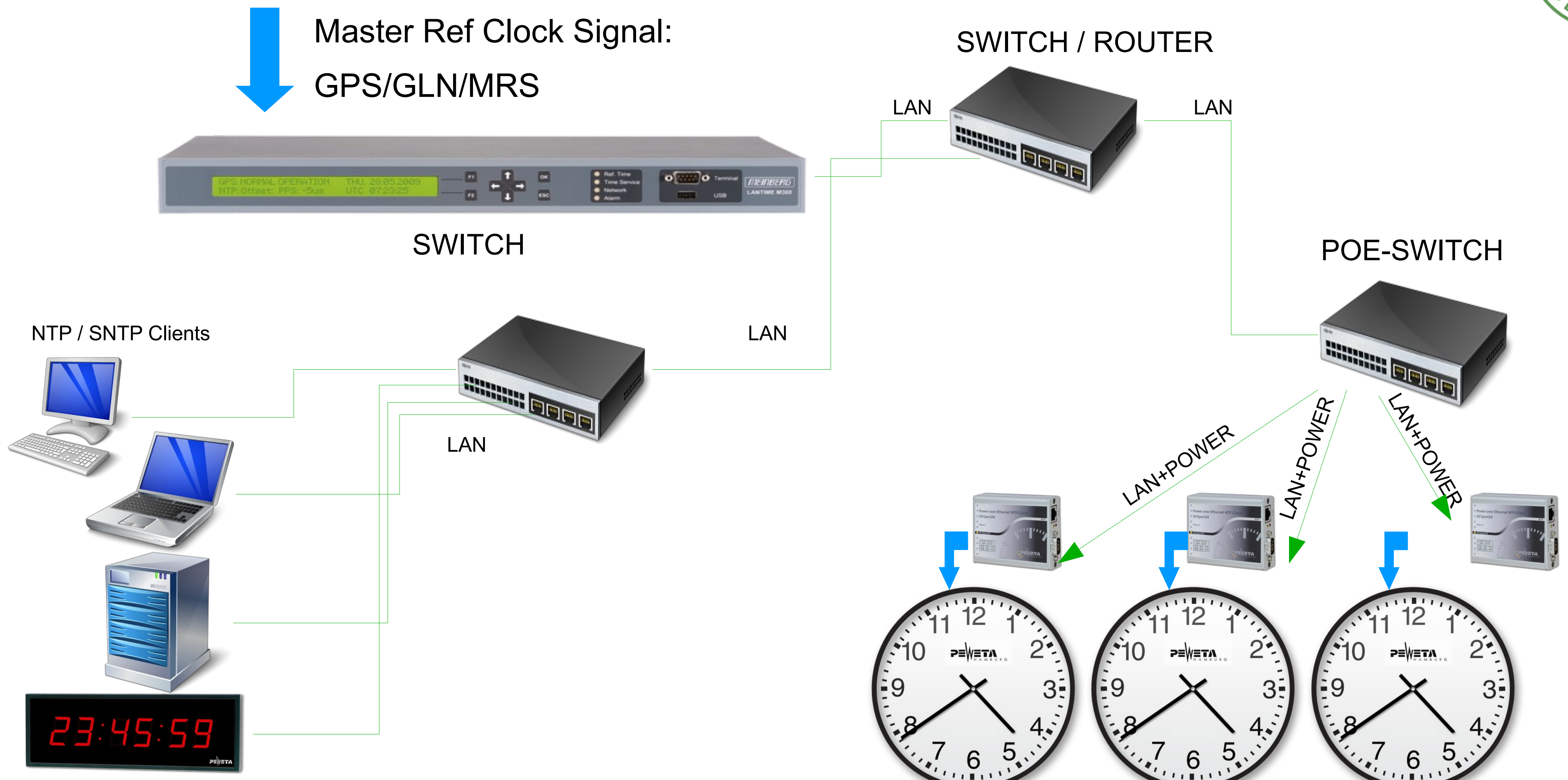
- Preferably for simple/dumb (S)NTP clients
- Several servers share a common, virtual IP address
- Status monitoring via Multicast oder Unicast, specific port
- Only one of the devices makes the service available via the virtual IP address
- When this devices fails, a different device takes over and provides the NTP service via the same virtual IP address
- Clients of the NTP cluster don't even notice an interruption

Core Modules of the LANTIME



NTP Packets

Typical NTP Network Architecture



Time Service Jamming and Spoofing

- Jamming is easy, disrupts reception of time information, detection is easy
- Spoofing tries to provide a wrong time
- Spoofing is always possible, it’s just a question of efforts
- Different attack vectors: GPS/GNSS, network, ...
- Spoofing detection is not easy: different time sources, different networks, times from different geographic locations, plausibility checks
- Monitoring to detect spoofing, e.g. monitoring feature of the LANTIME models
- Holdover mode in case spoofing is deteted → Stable oscillator
- See also: https://kb.meinbergglobal.com/kb/time_sync/time_service_jamming_and_spoofing

Meinberg: What Happens if ...

... all reference signals become unavailable:

- LANTIME starts freewheeling, but immediately generates an alarm via email, SNMP, etc.
- High quality disciplined oscillator still provides very accurate time during the trust time (holdover),
- Time drifts by ca. 22 µs per day or 788 ms per year (depending on oscillator quality, e.g. for OCXO).
- NTP trust time should be configured accordingly to accept the clock in a longer term even if it is not sync (default value is 4 days).
- NTP clients keep the same time, which drifts slowly with the NTP server time
- All NTP clients are still synchronized to each other

  --> **don't just ignore the time from a drifting server**

Meinberg: What Happens if ...

... the LANTIME fails or a connection to it fails:

- Clients are running further without NTP server as a reference.
- Clients drift in general ca. 0.5 s - 1 s per day.
- Each client drifts with its own rate, so time offsets between clients increase over time

Additional Information

Additional information on these basic topics can be found in the Meinberg Knowledge Base:

https://kb.meinbergglobal.com/kb/start

MEINBERG SYNC ACADEMY

EST 2014

Gaining Knowledge from the Experts

MEINBERG