Teaching Penguins to Tell Time

Rob van der Heij
IBM Netherlands
rvdheij@nl.ibm.com

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

BookManager*
DB2*
DFSMS/MVS*
DFSMS/VM*
e-business logo*
Enterprise Storage Server
ESCON*
FICON
GDDM*

IBM*
IBM logo*
Language Environment*
Multiprise*
MVS
NetRexx
OpenEdition*
OpenExtensions
OS/390*

Parallel Sysplex*
PR/SM
QMF
RACF*
RAMAC*
S/390*
S/390 Parallel Enterprise Server
VisualAge*
VisualGen*

VM/ESA*
VSE/ESA
VTAM*
z/Architecture
z/OS
z/VM
zSeries

The following are trademarks or registered trademarks of other companies.

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation; LINUX is a registered trademark of Linus Torvalds; Penguin (Tux) compliments of Larry Ewing; Tivoli is a trademark of Tivoli Systems Inc.; Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries; UNIX is a registered trademark of The Open Group in the United States and other countries; Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation; SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC. * All other products may be trademarks or registered trademarks of their respective companies.

Notes: Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

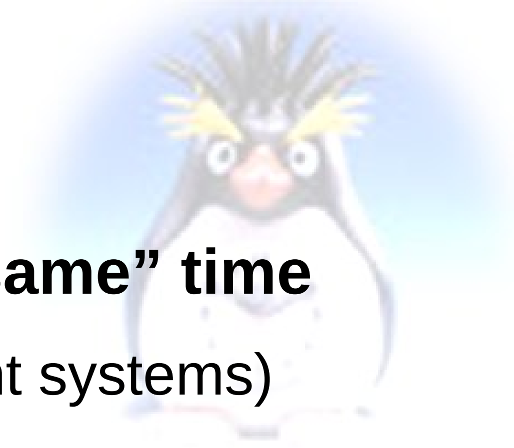
- **Why do we care?**
- **Hardware Clocks**
- **Time zones and UTC**
- **Network Time Protocol**
- **NTP and Linux on z/VM**

Why care about correct time?

- **For isolated systems correct time is convenient**
  - Time stamps on files
  - Reliable backup and restore
  - Applications that can pick up system date and time

```
MSG FROM: [illegible]     --DKIBMVM2 TO: NLX3951 --UITVM1           06/04/97 20:55:45
To:    Rob van der Heij
From: [illegible]                                                     at DKIBMVM2

Hi, Rob.

Next time anyone IPLs DKIBMVM2, could they also set the clock.  It is
now five minutes fast.  That means that I rush down to the train and
stand there waiting :-)

VM1 is about a couple of minutes fast.

    [illegible]                                              [illegible]
```


Why care about correct time ?



- **Distributed applications require the “same” time**
  - File sharing (e.g. edit and make on different systems)
  - Security (e.g. Kerberos tokens)
  - Convenient for debugging and tracking
- **Most systems get the time from someone else**
  - No infinite accuracy possible
  - Required quality of clock depends on application

Who has the correct time?

- **Universal Time 1 (UT1)**
  - Computed from astronomical observations
  - Speeds up and slows down with earth rotation
- **International Atomic Time (TAI)**
  - Based on Cesium-133 radiation
- **Coordinated Universal Time (UTC)**
  - Derived from TAI
  - Adjusted to UT1 with occasional leap seconds

Daytime Protocol

- **Public Internet Service**
  - Provided by several organizations (e.g. NIST)
- **Documented in RFC 867**
  - Query to UDP or TCP port 13
  - Format of response is different per server

```
pipe tcpclient 192.43.244.18 13 linger 1 | deblock linend 0a | xlate a2e | cons
53108 04-04-13 10:08:46 50 0 0 432.8 UTC(NIST) *
```

Julian Day

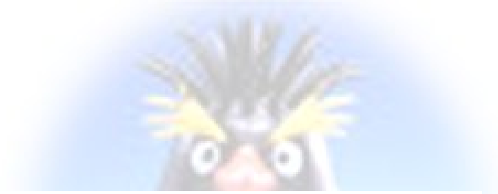
DST

Network delay

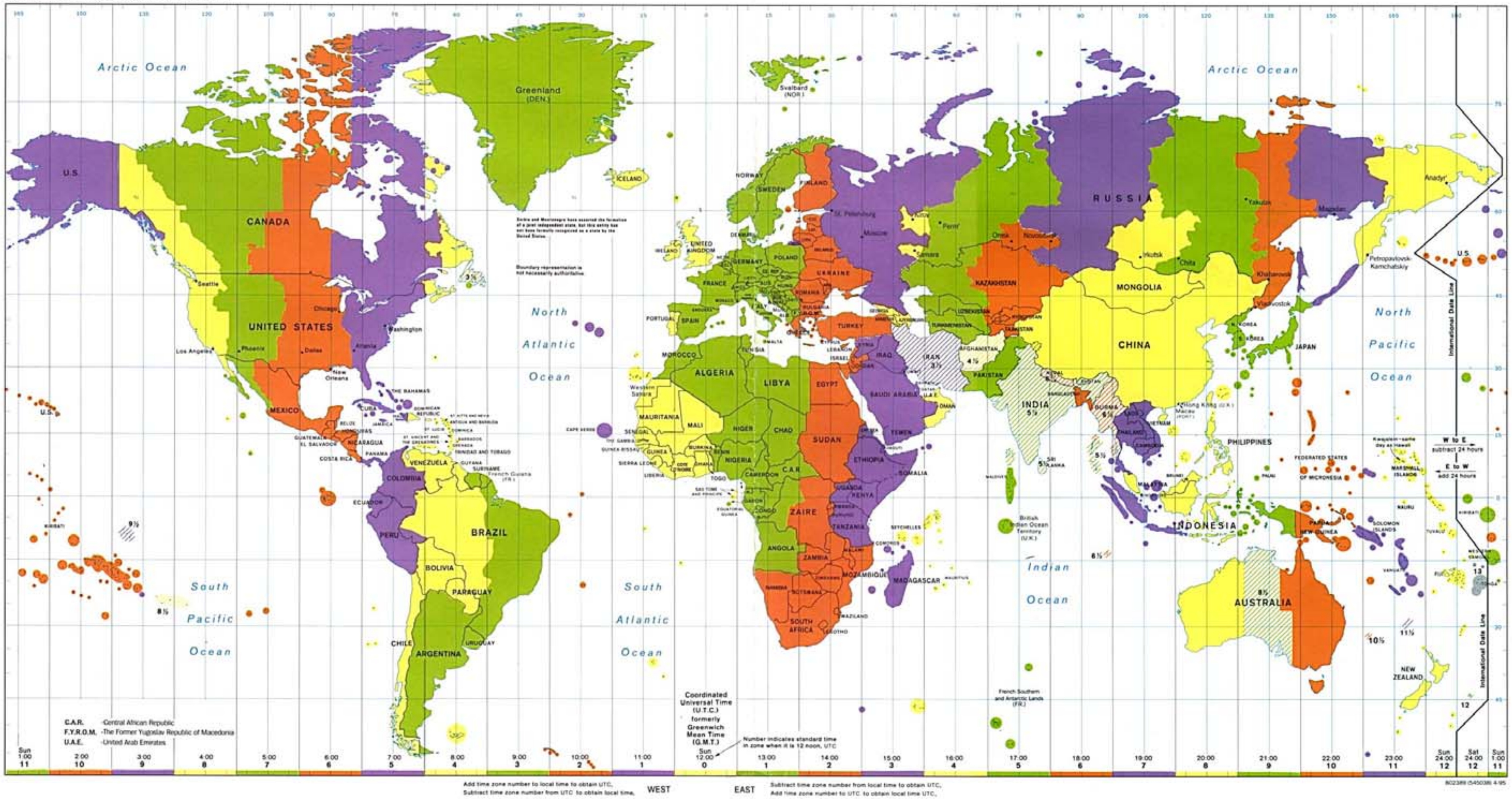
Clock Health (error < 5 s)

Leap seconds

Time Zones



Standard Time Zones of the World



Time zones and UTC

- **Linux system clock is in UTC**
  - Number of seconds since 1970
- **Local time computed from System Time**
  - GNU C Runtime Library routines
  - Defined by zone info in /etc/localtime
    - Offset to UTC
    - Beginning and end of DST

Quality of Clocks

- **System clock is at best approximation of UTC**
  – Delay, jitter, offset, drift
- **High quality time keeping is expensive**
  – Trade-off between various type of error
- **Quartz controlled clocks**
  – Typical frequency offset in the range of 10’s PPM
  – Variation of several PPM due to temperature changes
  – When not corrected, may add up to seconds per day

1 ppm ~ 30 s / yr

The Ideal Broken Clock

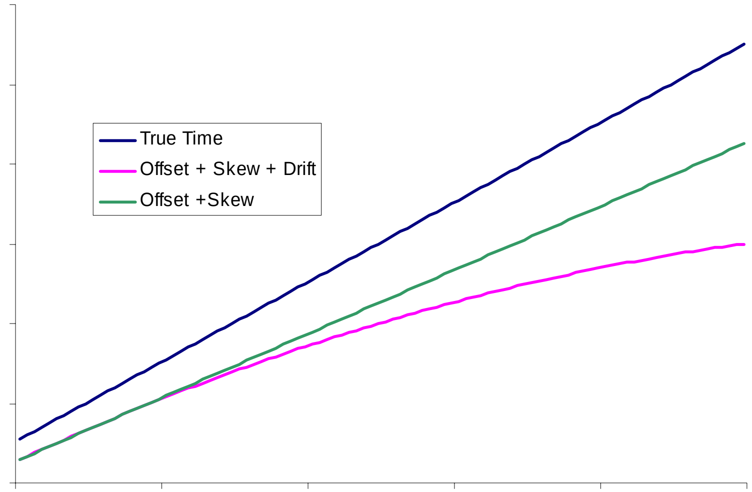
Clock differs from true time

Constant base offset

Different frequency (skew)

Drift due to aging

Jitter

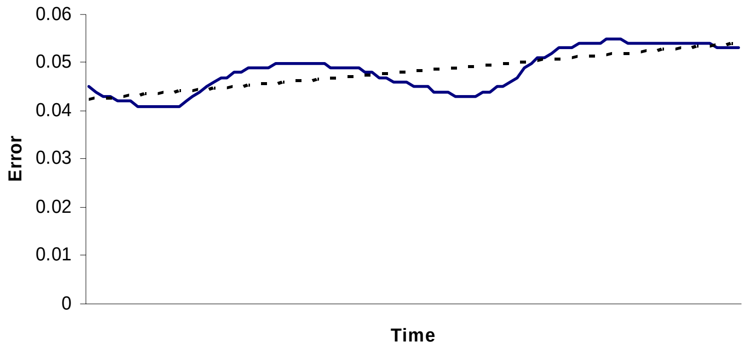


PC Clock

- **Real Time Clock with battery backup**
  - Keeps time while PC powered off
  - Cheap and not very accurate
    (e.g. several seconds per day off)
  - Low precision (one second)
    - Not suitable for system timing

PC Clock

- **Software Clock in Linux**
  - Initialized from the RTC during boot process
  - Incremented by timer interrupts
  - High resolution based on CPU clock frequency
    - Exploits CPU cycle counter
  - Can be adjusted through NTP
  - Used to compensate RTC drift
    - Be careful with adjusting RTC



PC Clock

- **Software Clock in Windows**
  - Initialized from RTC during boot
  - Incremented by timer interrupts
  - Corrected frequently using RTC
  - RTC and Software Clock in local time rather than UTC
    - Complications with DST

Dual Boot PC

- Linux must be aware RTC is in local time

zSeries Clock

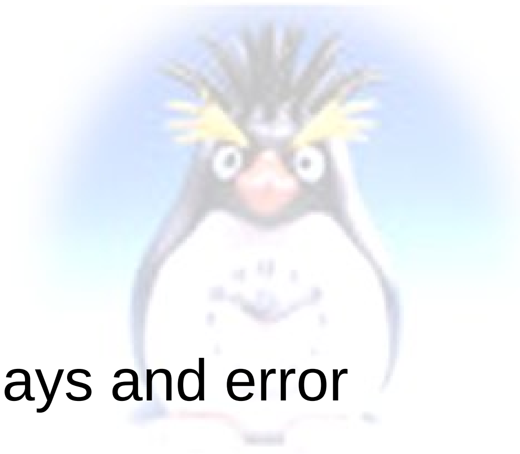
- **Hardware Time-of-Day (TOD) Clock**
  - Reasonable stable and high precision
  - Very cheap to read (STCK instruction in SIE)
  - Frequently set manually by Operator at IPL
    - Not synchronized to official time

- **Linux Software Clock**
  - Initially based on the TOD Clock
  - Even with a stable clock – wrong all day
  - Attractive to synchronize with external source

Network Time Protocol

- **Framework to synchronize clocks**
  - Defined in RFC 1305 (v3) and RFC 2030 (v4)
  - Format allows for very high precision
  - Formalizes quality of time reference
    - Stratum: Number of hops to a primary reference
    - Precision: Accuracy of the clock
  - Implementations available for many platforms
  - Many high quality public time servers

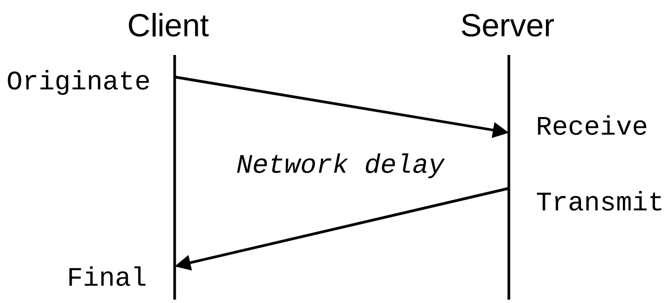
Network Time Protocol



UDP Packet to port 123

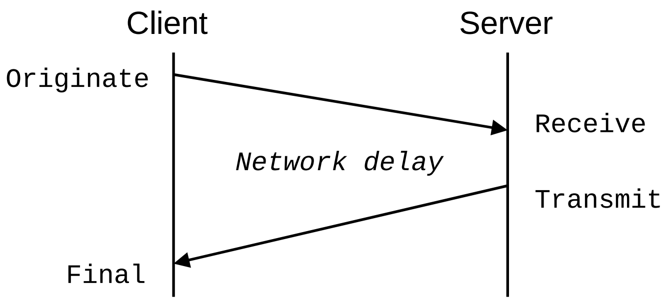
- Contains several timestamps to compute delays and error
- Timestamps in 32 + 32 bits (0.2 ns resolution)

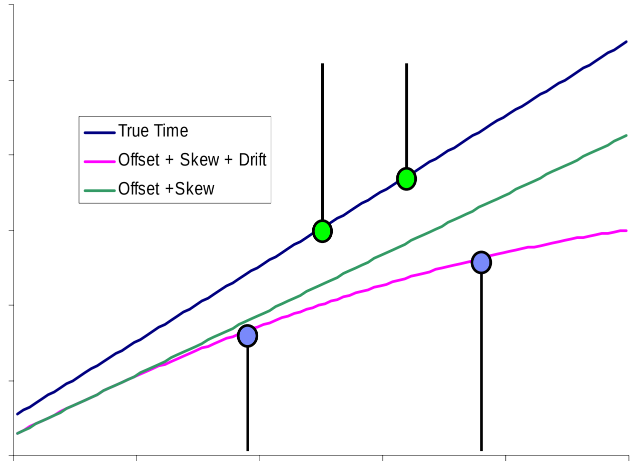
- Allows client to compute the error
- Network delay is assumed to be symmetrical and constant



Network Time Protocol

- **Using the timestamps**
  - Estimate the correct time
  - Compute the maximum error
- **A proper model for the delay**
  - Interpolate time for high precision
  - Extrapolate for long term measurement





NTP on Linux

- **Provisions in the kernel to adjust time**
- **The ntpd daemon and additional binaries**
  - SuSE xntp package
  - Red Hat ntp package
  - Download from www.ntp.org
- **Client configuration is trivial**
  - Single entry in /etc/ntpd.conf to list the time server
  - Finding the right server is harder

Starting ntpd

- **Special adaptation mode**
  - Runs for approximately 15 minutes
  - Exchanges messages with server to adapt oscillator
  - Clock is “stepped” to approach the obtained time
  - An excessive time difference causes ntpd to panic
  - Frequency in ntp.drift maintained

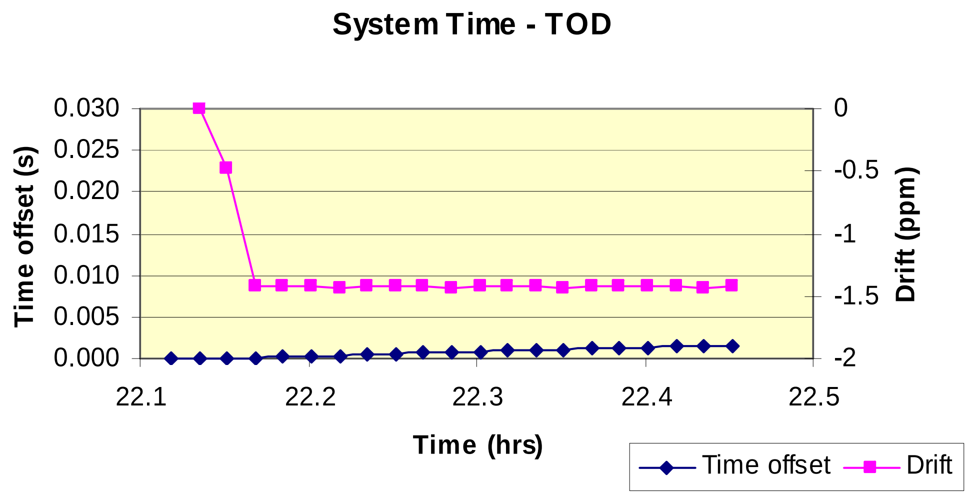
```
Apr  4 12:32:07 box ntpd: ntpd startup succeeded
Apr  4 12:32:07 box ntpd[712]: ntpd 4.1.1c-rc1@1.836 Thu Feb 13 12:17:19 EST 2003 (1)
Apr  4 12:32:11 box ntpd[712]: precision = 9 usec
Apr  4 12:32:11 box ntpd[712]: kernel time discipline status 0040
Apr  4 12:40:46 box ntpd[712]: time set 311.058971 s
Apr  4 12:40:46 box ntpd[712]: synchronisation lost
```

Example of NTP setting the clock

- **Initial frequency offset during adaptation mode**
  - Taken from drift measured during earlier runs
  - Assumes clock is set by hwclock
  - For Linux on zSeries this does not hold

SuSE boot.clock broken when hardware clock not set UTC

```
cat /etc/sysconfig/clock
#
# Set to "-u" if your system clock is set to
UTC, and to "--localtime"
# if your clock runs that way.
#
HWCLOCK="-u"
```



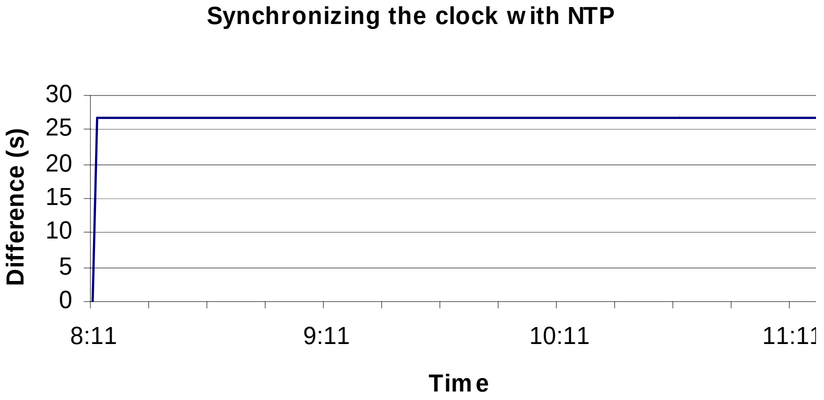
Running ntpdate

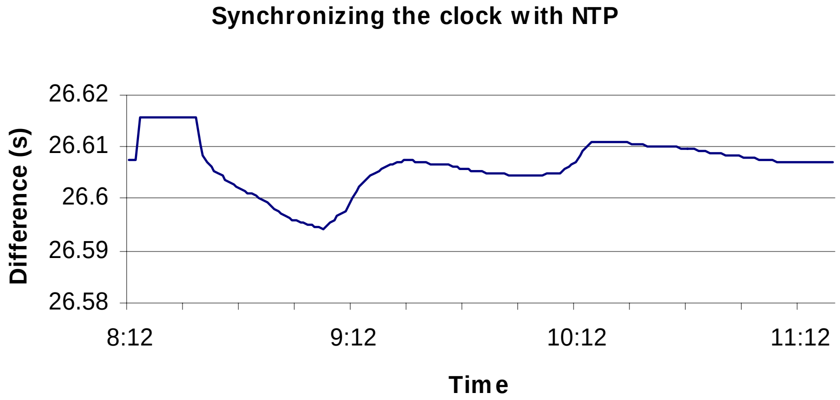
- **Correct time only available after adaptation mode**
  - Not attractive for workstations
  - May be confusing for other daemons started
- **SuSE runs ntpdate once during startup**
  - Obtains an estimate for current time from one server
  - Time difference probably small enough to avoid stepping

```
Apr 14 10:12:18 linux390 ntpdate[9876]: step time server 9.61.40.85 offset 26.607496 sec
Apr 14 10:12:18 linux390 ntpd[9879]: ntpd 4.1.1@1.786 Mon Sep 29 06:44:00 UTC 2003 (1)
Apr 14 10:12:18 linux390 ntpd[9879]: precision = 22 usec
Apr 14 10:12:18 linux390 ntpd[9879]: kernel time discipline status 0040
```


Running ntpdate

- **Difference between Linux clock and TOD clock**
  - Initial clock stepping via ntpdate
    - Can be disabled with `XNTPD_INITIAL_NTPDATE` setting in `/etc/sysconfig/xntp`
  - Subsequent tuning of oscillator





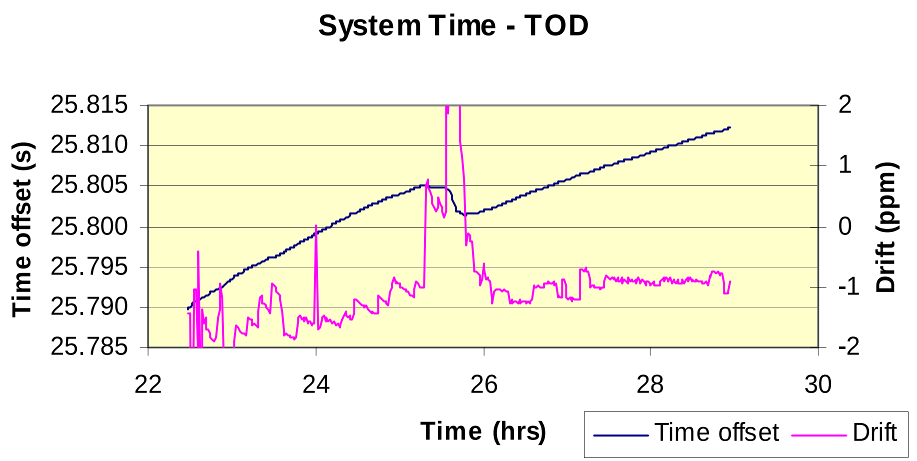
Running ntpd

- **Normal mode**
  - Continuous exchange of messages with servers
  - Delays compared to compute frequency errors
  - Message rate lowered over time to reduce network load

```
linuxgw:~ # ntpq -p -n
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 127.127.1.0     127.127.1.0     10 l    6   64  377    0.000    0.000   0.015
+9.61.40.85      9.61.37.158      2 u   12 1024  377   15.585    1.132 100.497
+9.1.24.204      9.41.0.213       2 u   17 1024  377  104.729    0.079 313.260
*9.41.0.213      .GPS.            1 u  924 1024  377   57.036    0.898   0.185
-9.154.60.2      .hopf.           1 u 1002 1024  377   99.453    8.494   3.280
```

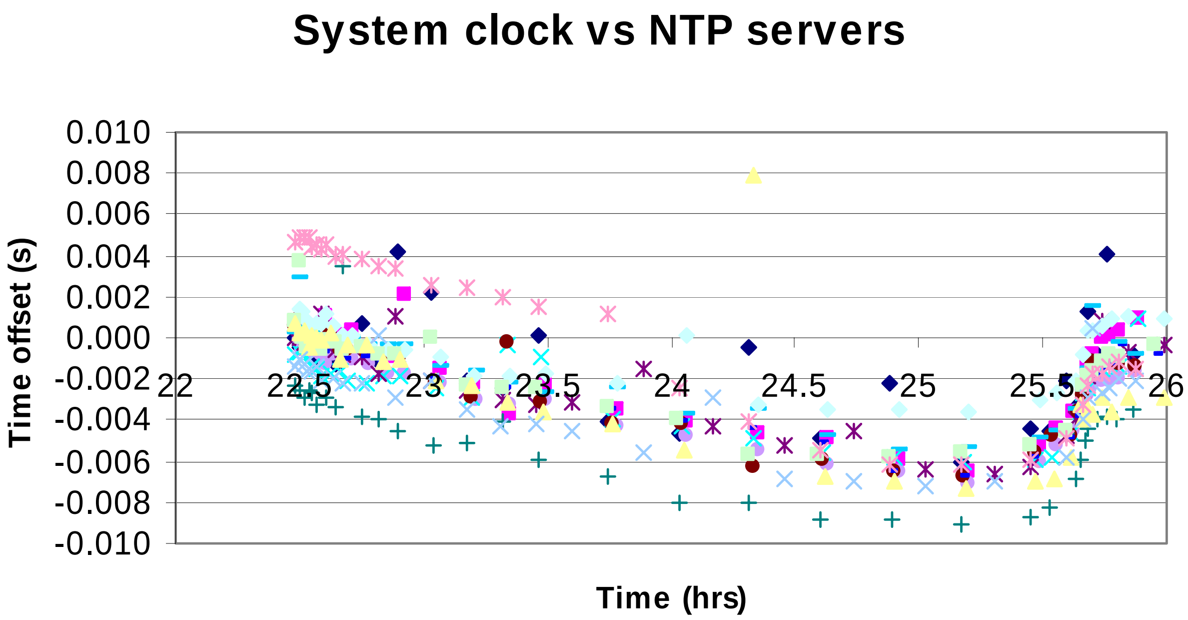
Example of NTP adjusting the clock

- **Over time NTP will tweak frequency to get system clock close to observed true time**
- **You can not have it all**
  - Stable clock with little jitter
  - Clock that follows true time

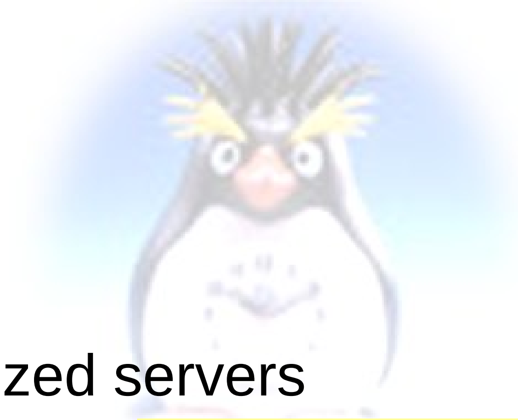


Example of NTP adjusting the clock

- **Jump in time appears to be consistent**
  - Most likely something in the network that changed delay
  - Could be related to system load



Synchronizing Linux with NTP



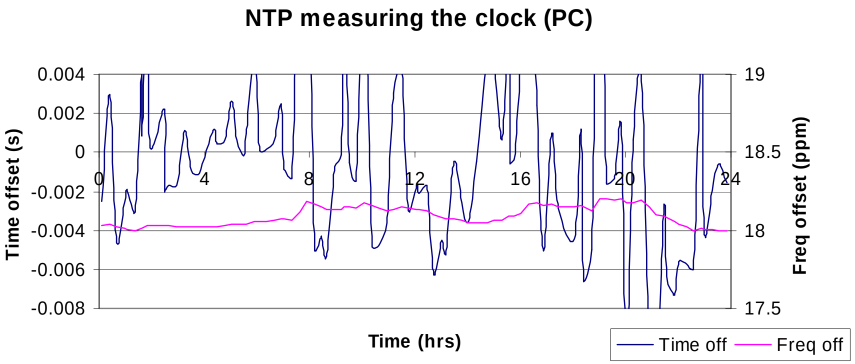
- **NTP works for Linux on zSeries too**
  - Keeps Linux clock close to other synchronized servers
- **Running ntpd is not for free**
  - CPU cost for idle server may increase with 50%
  - May increase memory footprint with 50%
  - Will depend on the number of time servers polled
  - Do your requirements justify the cost?

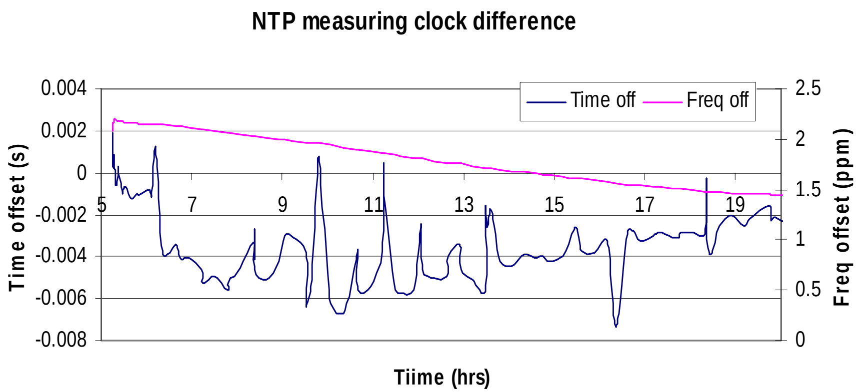
Experiment on z990
Idle: 4.1 ms/min
With ntpd 6.2 ms/min

Experiment:
Idle server 2270 pages
With ntpd 3385 pages

So how close do we get?

- **NTP keeps its own statistics**
  – The loopstats is only what NTP believes
- **How do we know what is true?**
  – The zSeries TOD wrong? Is this bad?
  – NTP getting the time wrong ?

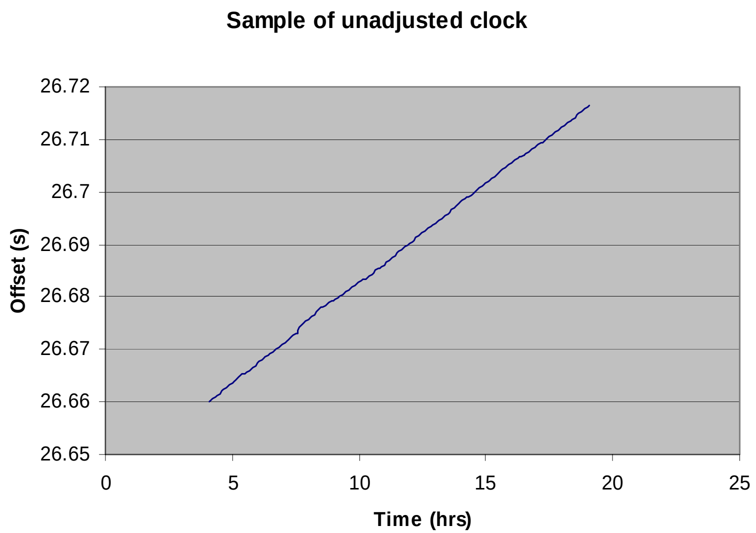




So how close do we get?

- **Sample of a 9672 TOD clock**
  - NTP adjusted system time versus hardware TOD clock
    - Numbers as good as NTP can get
    - Not representative for all CPUs
  - In this case
    TOD clock is 4 ms/hr too slow

    100 ms / day ~ 1 ppm



Cheap way to set the clock

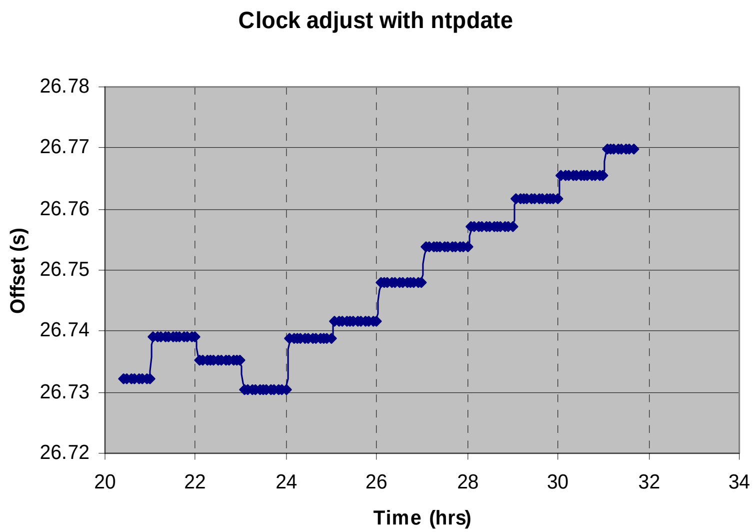
- **Reduce cost by setting the clock once**
  - Compensates for inaccurate set VM clock
  - Drift of zSeries TOD clock probably seconds per month
  - Run ntpdate once during boot process
    - Standard code (use ntpd –q rather than ntpdate)
    - May need the –g and –x options for large adjustments
    - It does not compensate for drift

Cheap way to set the clock

- **Set the virtual machine TOD offset**
  - Can be done in PROFILE EXEC before boot
  - Isolates Linux from the details of the hardware clock
  - No adaptation phase needed, no sudden clock shift
  - With SET VTOD the TOD clock offset can be set equal to the offset of another virtual machine
    - Linux servers with identical clock
    - Clock could be read completely in user space

Cheap way to set the clock

- **Run ntpdate occasionally cron**
  - Not recommended by everyone
  - It does not compensate for drift but jumps the clock
  - May even step your clock back
- **Doing so affects global time**
  - Simultaneous requests will overload the NTP servers
  - Will be removed in the future
    - Instead use ntpd –q –x

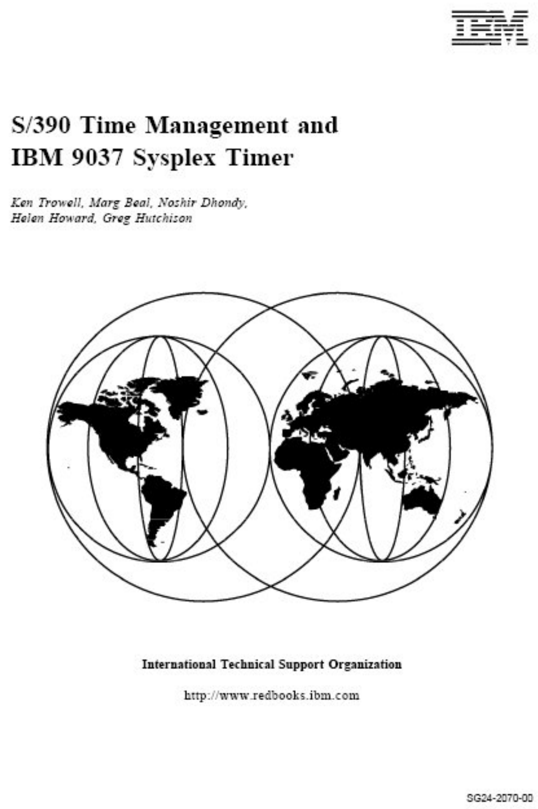


Using External Time Reference

- **IBM 9037-2**
  - Used to synchronized TOD clocks over multiple CECs in parallel sysplex
  - Can also use external reference to synchronize the clock
    - Dial-up to ACTS
    - RF receiver providing timestamp and PPS signal
  - z/VM has no explicit support for ETR but can benefit from a TOD clock kept exact

1000 Linux guests not running ntpd pays for a 9037-2

IBM 9037 Redbook


S/390 Time Management and
IBM 9037 Sysplex Timer

Ken Trowell, Marg Beal, Noshir Dhondy,
Helen Howard, Greg Hutchison

International Technical Support Organization

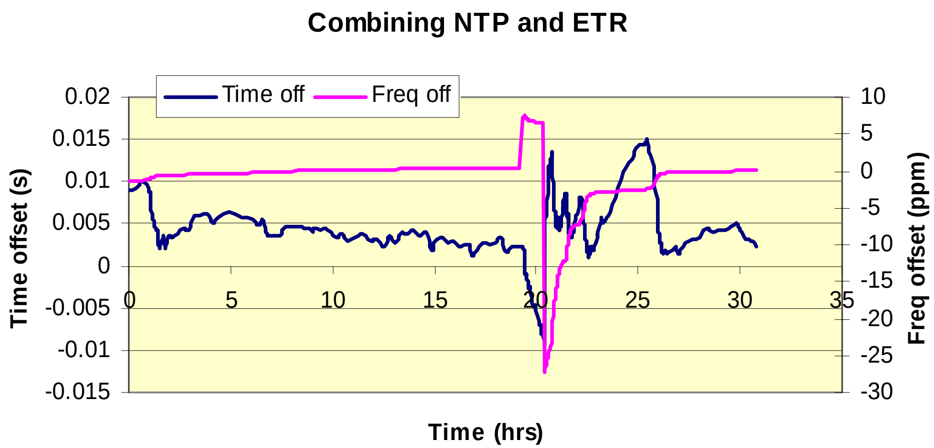
http://www.redbooks.ibm.com

SG24-2070-00

- **ITSO Redbook SG24-2070 S/390 Time Management and IBM 9037 Sysplex Timer**
  - z/OS configuration
  - ETR configuration setup

Using External Time Reference

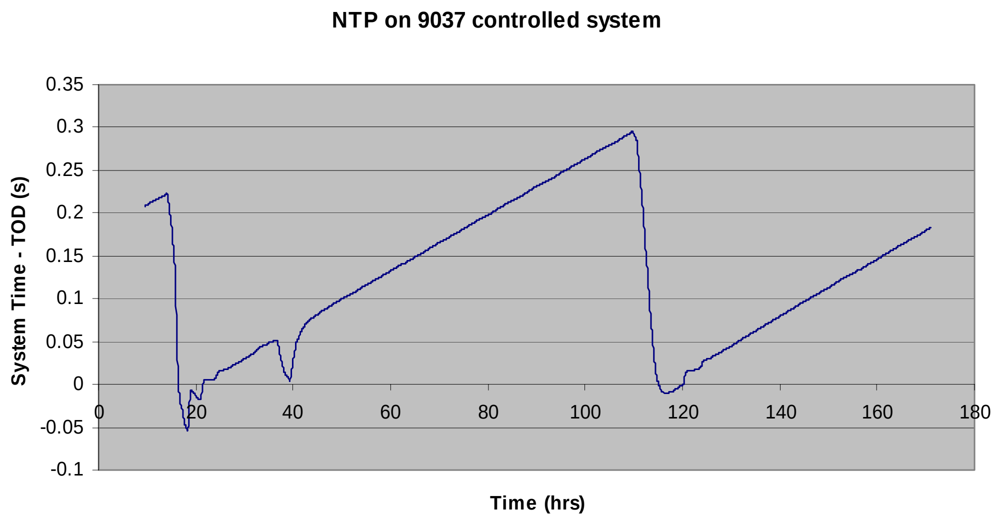
- **Use either ETR or NTP**
  - 9037 is not very subtle
  - NTP algorithms are not tuned for ETR clock steering
- **Combining NTP with ETR results in poor quality**
  - Increased swings
  - Correction overshoot
  - No improved long term stability



Using External Time Reference

- **Another example of clock quality**
  - System time driven by NTP (with 15 time servers)
  - zSeries hardware TOD clock (steered by 9037)

  - Stable drift: ~ -1 ppm
  - Correction: ~ 10 ppm

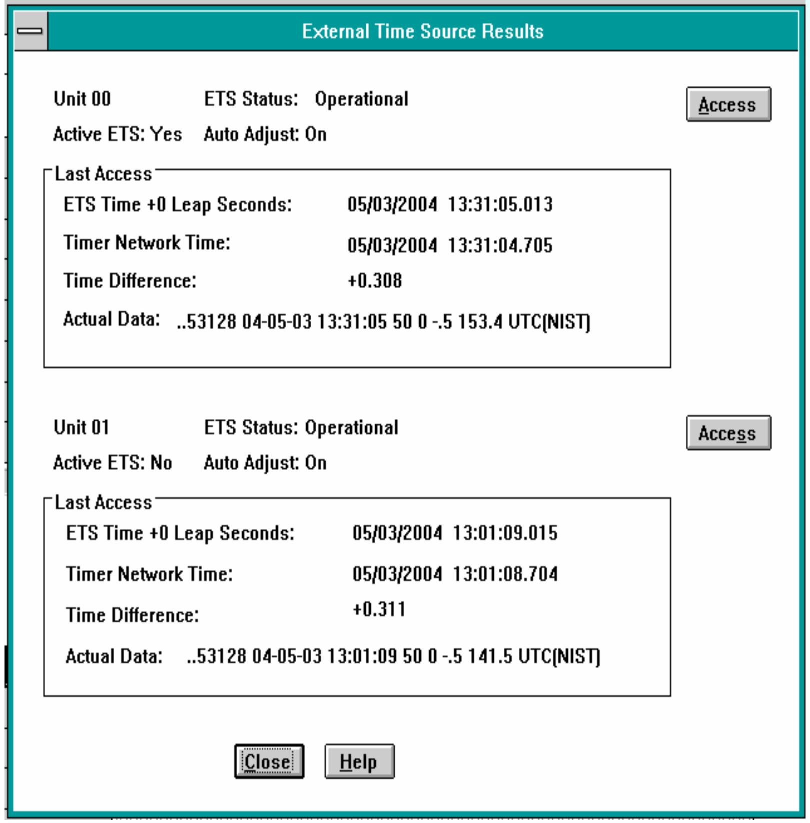


Using External Time Reference

- **Verified NTP measurements with 9037 status**
  - Jumps in the curve match the ETR dial-up times
  - Difference reported by ETR matches the measurement very well

    300 mS in 4 days

    ~ 1 ppm



Conclusion

- **NTP with Linux on z/VM is no silver bullet**
  – Expensive to run, algorithms do not fit completely
- **If you can afford some drift**
  – Set the clock once during boot with ntpd -q or SET VTOD
- **When clock stepping back is acceptable**
  – Compensate drift by running ntpd –q via cron
- **When more accurate time is necessary**
  – Run ntpd as daemon to enable kernel time discipline
  – Check the hwclock issues in the boot scripts
  – Carefully select the time servers to use as reference