Precision Time Protocol on Linux ~ Introduction to linuxptp

Ken ICHIKAWA
FUJITSU LIMITED.
LinuxCon Japan 2014

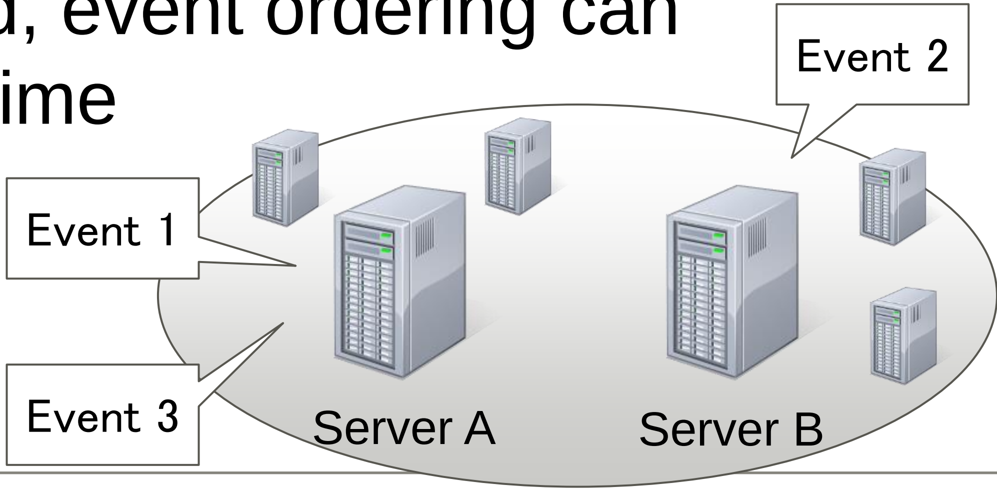
Agenda

- Background
- Overview of Precision Time Protocol (PTP)
- About PTP on Linux
- Tips
- For easy trial or development

Background

- Event ordering is very important
  - for incident analysis, performance analysis and so on
- Event ordering is based on timestamps
- Timestamps are collected from multiple servers

  ⇒Clock synchronization is important
- If precision and accuracy of clock synchronization are bad, event ordering can reverse against actual time



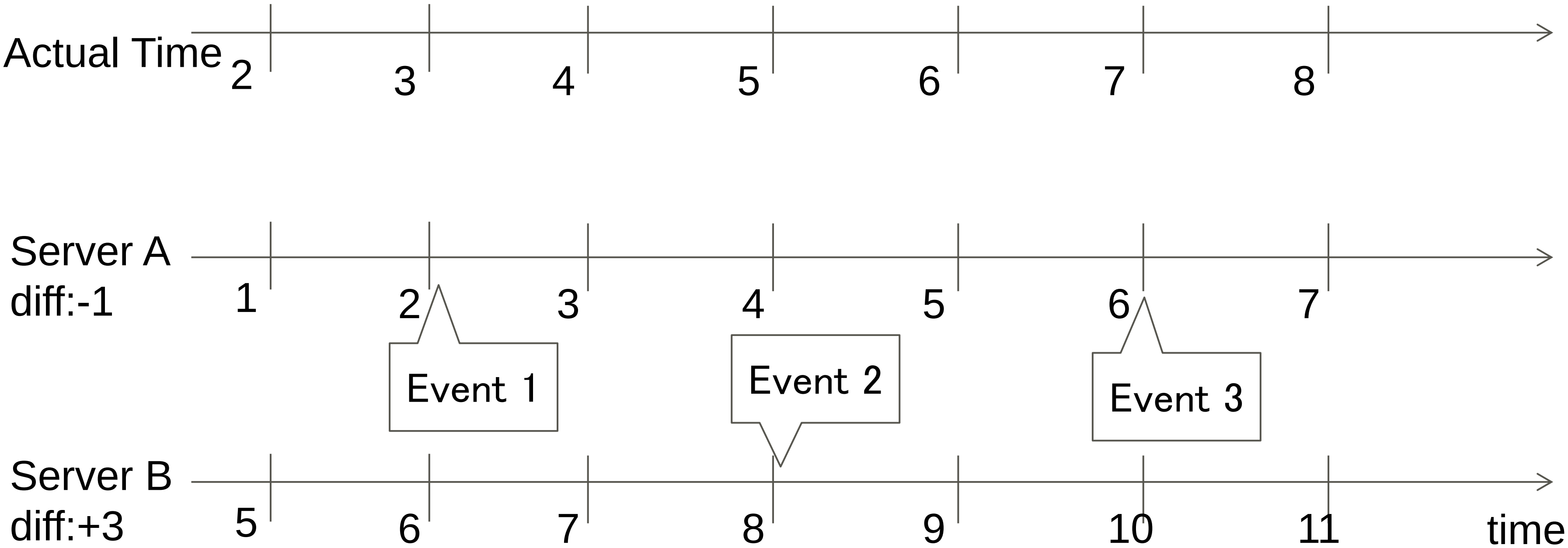
NTP is not enough

- NTP provides millisecond level synchronization
  - Maybe enough for remote machines, but not enough for locally cooperating machines
  - Many events occur in a millisecond in multiple servers

  ⇒Event ordering will frequently reverse

- Need another protocol
  - Higher precision and accuracy
  - Not need to synchronize large area, but local servers and devices

Example of wrong event ordering



Event Ordering based on Actual Time

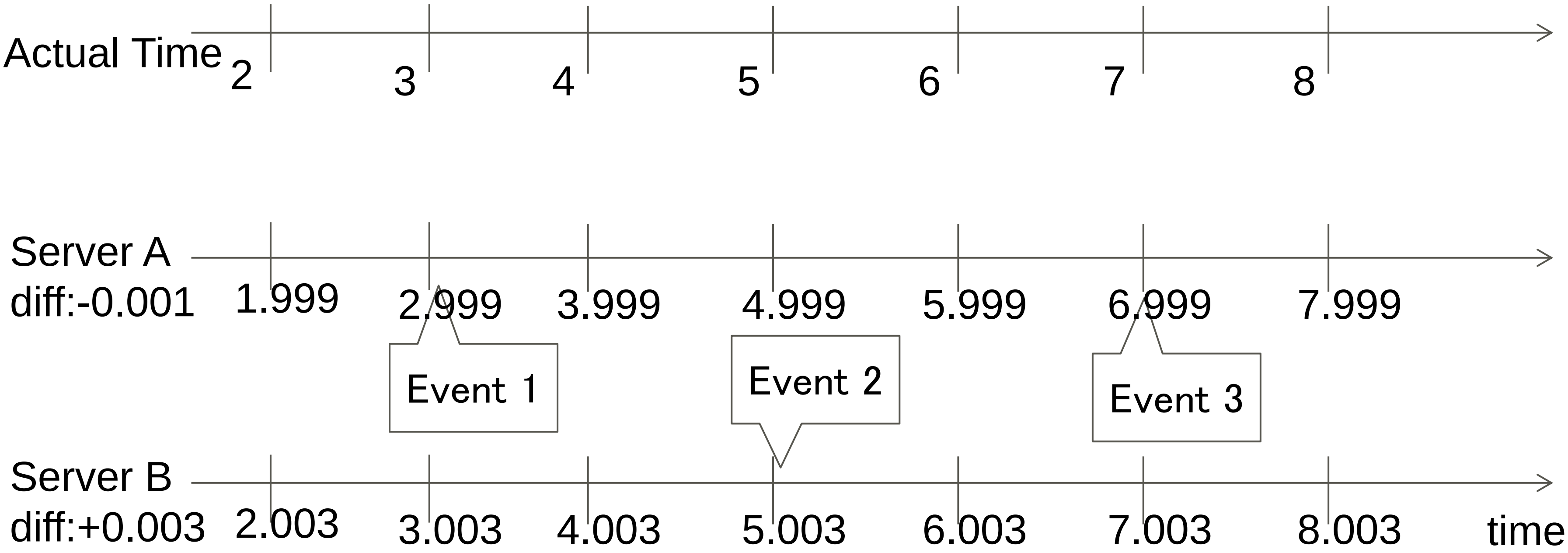
Time	Event
3	Event 1
5	Event 2
7	Event 3

Event Ordering based on Timestamp

Time	Event
2	Event 1
6	Event 3
8	Event 2

← reverse! ←

Example of correct event ordering based on better clock synchronization



Event Ordering based on Actual Time

Time	Event
3	Event 1
5	Event 2
7	Event 3

Event Ordering based on Timestamp

Time	Event
2.999	Event 1
5.003	Event 2
6.999	Event 3

correct!

Agenda

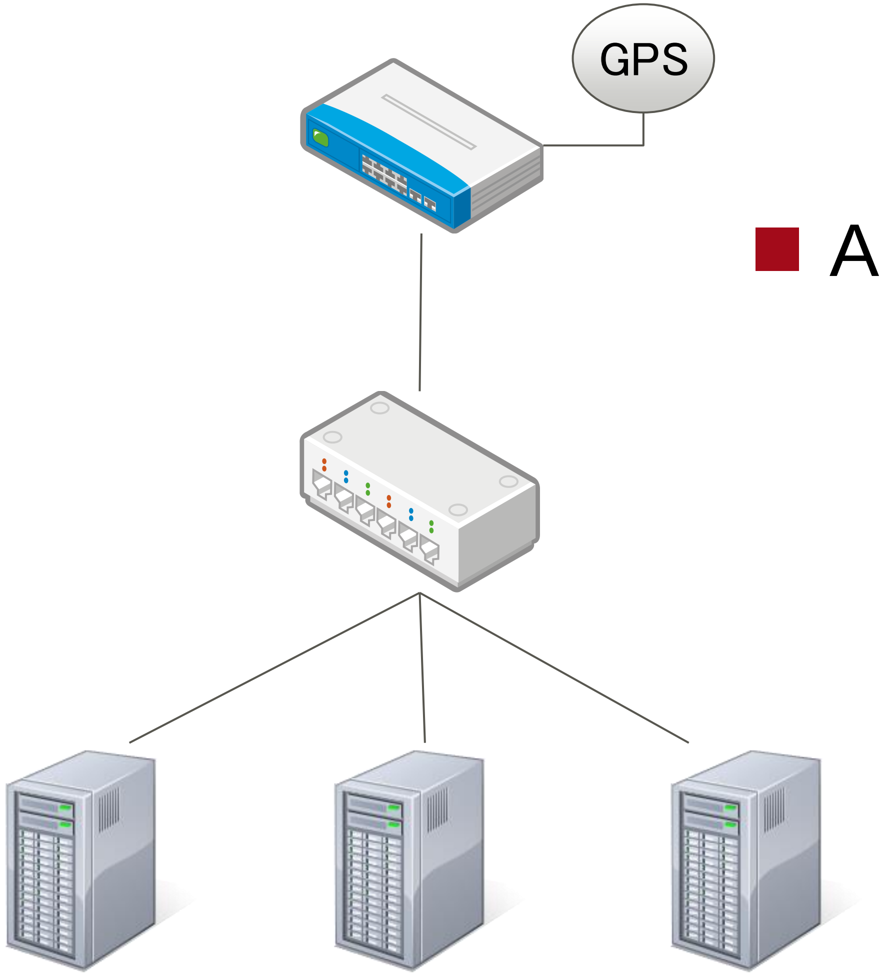
FUJITSU

- Background
- Overview of Precision Time Protocol (PTP)
  - What's PTP
  - Term explanation
  - About packet timestamp
- About PTP on Linux
- Tips
- For easy trial or development

Precision Time Protocol (PTP)

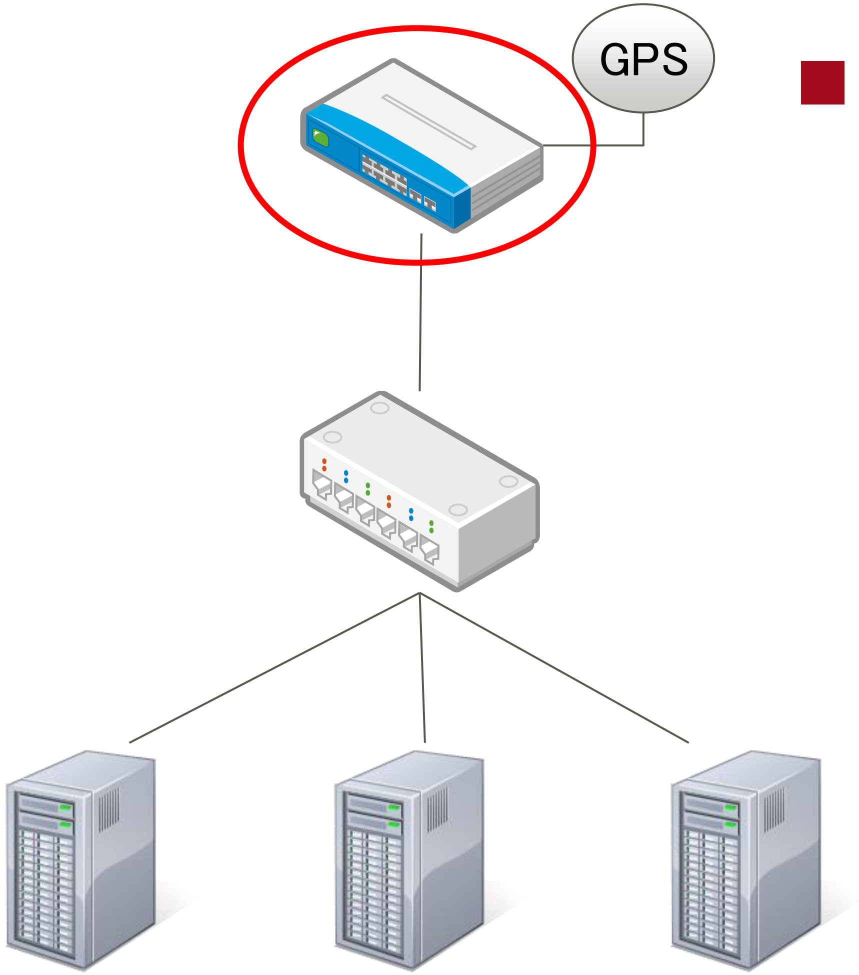
- Standardized protocol, IEEE1588
- Synchronize the clocks in local computing systems and devices
- Microsecond to sub-microsecond accuracy and precision
- Administration free
  - Capability to autonomously decide time server(master)
    - called Best Master Clock Algorithm (BMCA)

Term explanation



- An example of PTP network

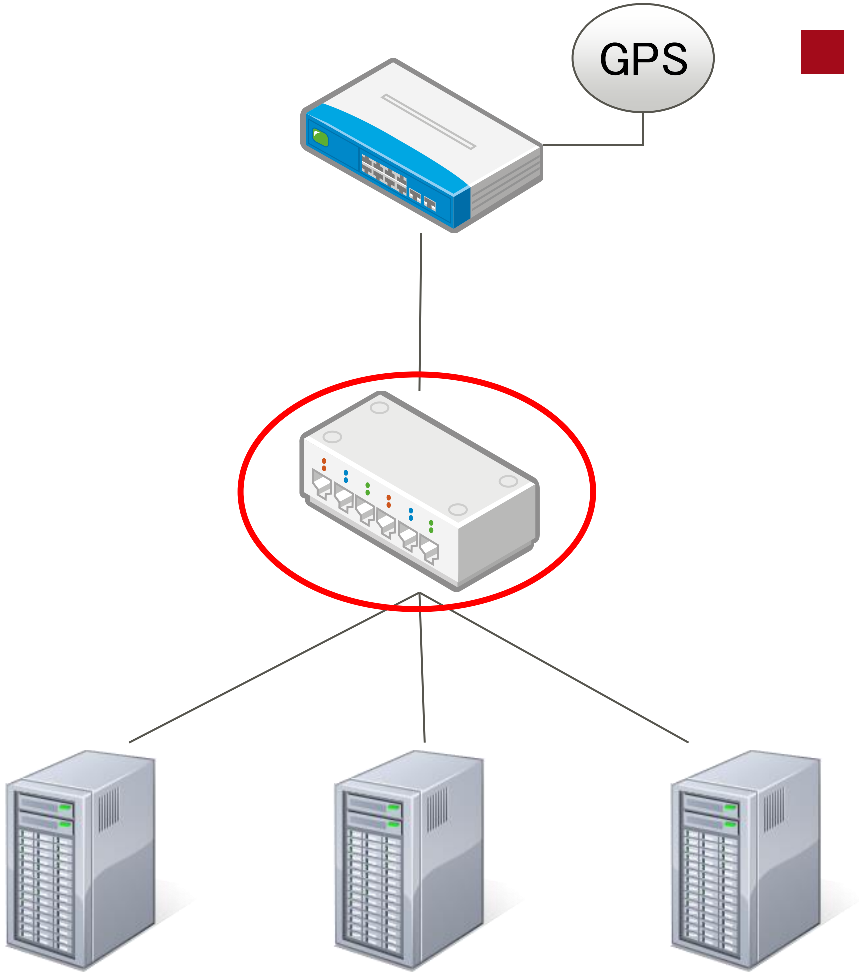
Term explanation



- Grandmaster Clock (Ordinary Clock)
  - Original time source for the PTP network
  - Typically synchronize its clock to external time (GPS, NTP and so on)
  - End point of PTP network is called Ordinary Clock

Term explanation

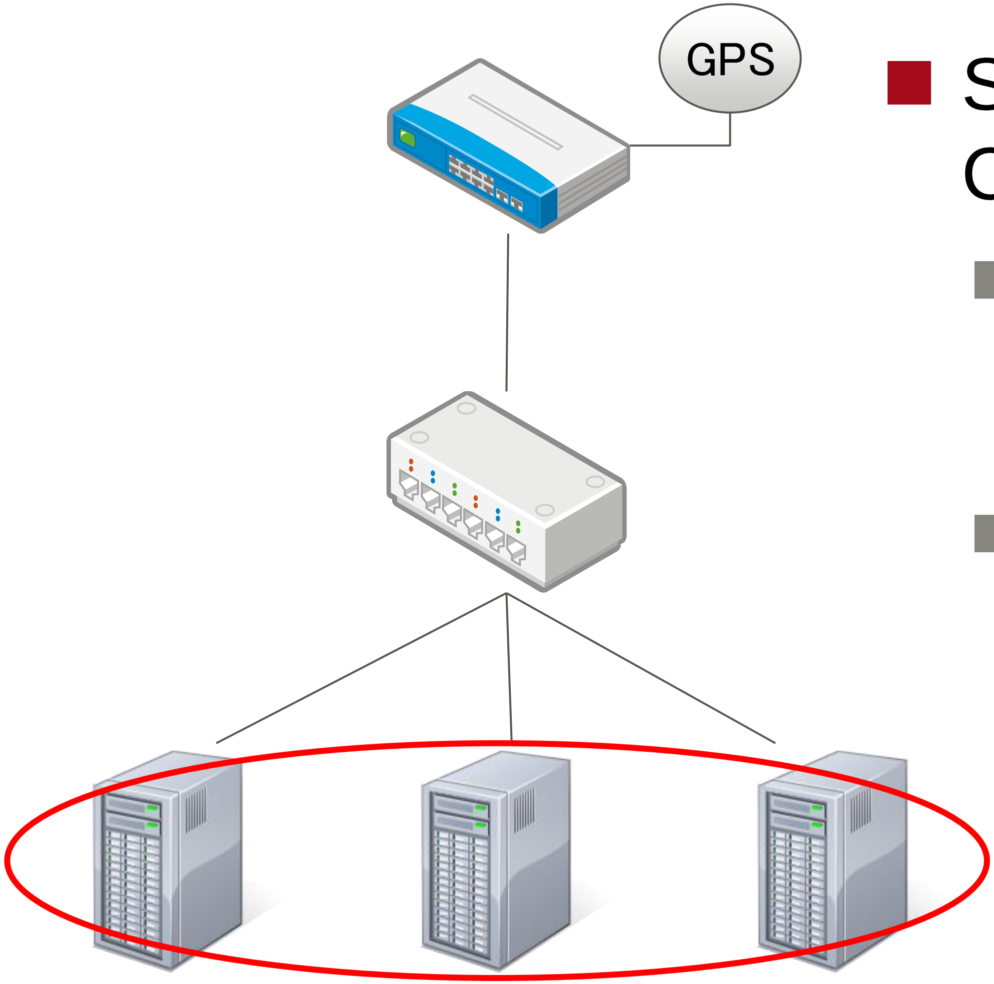
FUJITSU



- Boundary Clock
  - Typically it's switch
  - Synchronize its clock to a master
  - Serve as a time source to other (slave) clocks
  - May become Grandmaster clock if current Grandmaster is lost

- Master: serve as a time source
- Slave: synchronize to another clock

Term explanation



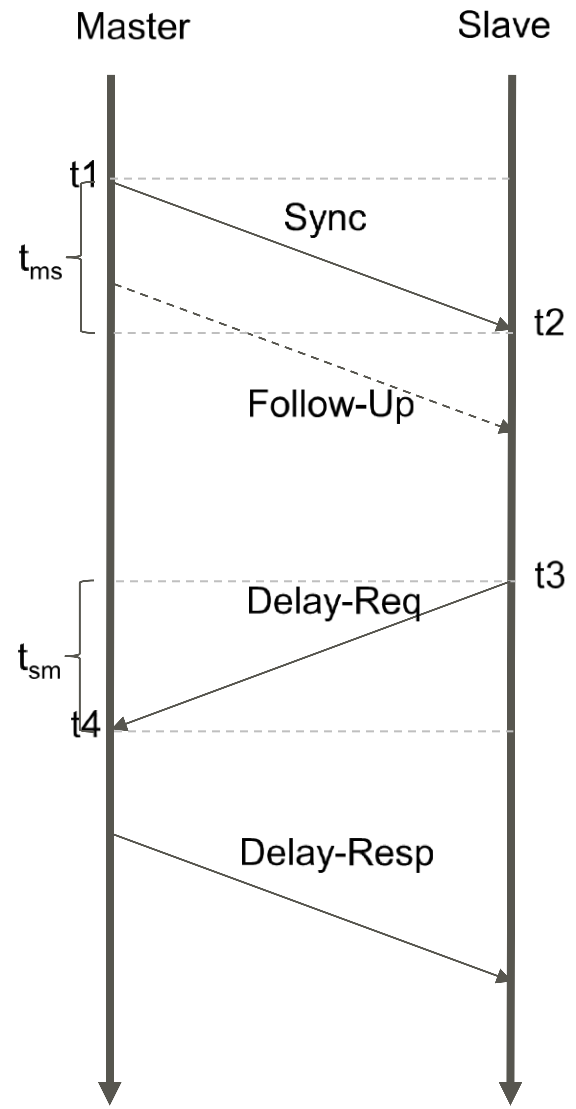
- Slave Clock (Ordinary Clock)
  - Synchronize its clock to a master (to the boundary clock in this example)
  - May become Grandmaster clock if current Grandmaster is lost

Packet timestamp

- Time offset between master and slave clocks is calculated based on timestamps at packet sending and receiving

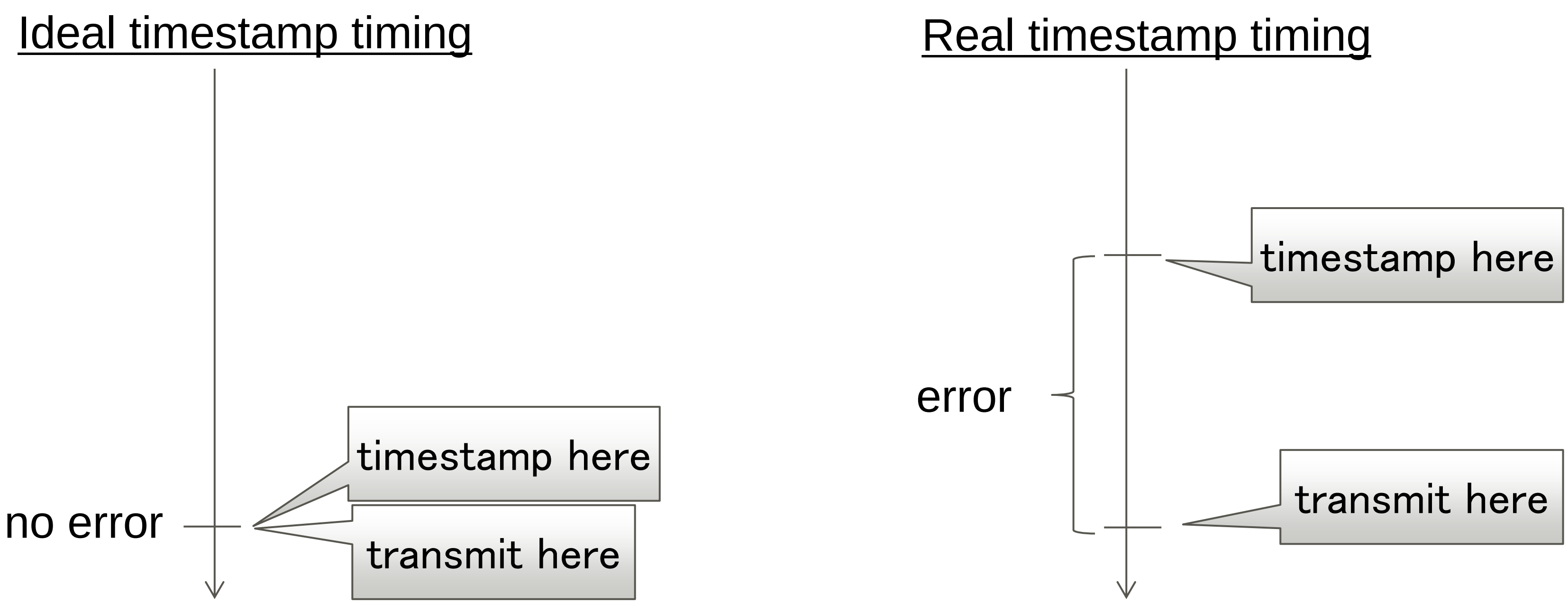
$$offset = t2 - t1 - \frac{1}{2}(t_{ms} + t_{sm})$$
$$= t2 - t1 - \frac{1}{2}\{(t4 - t1) - (t3 - t2)\}$$

- Packet timestamp accuracy is important for PTP



Timestamp timing

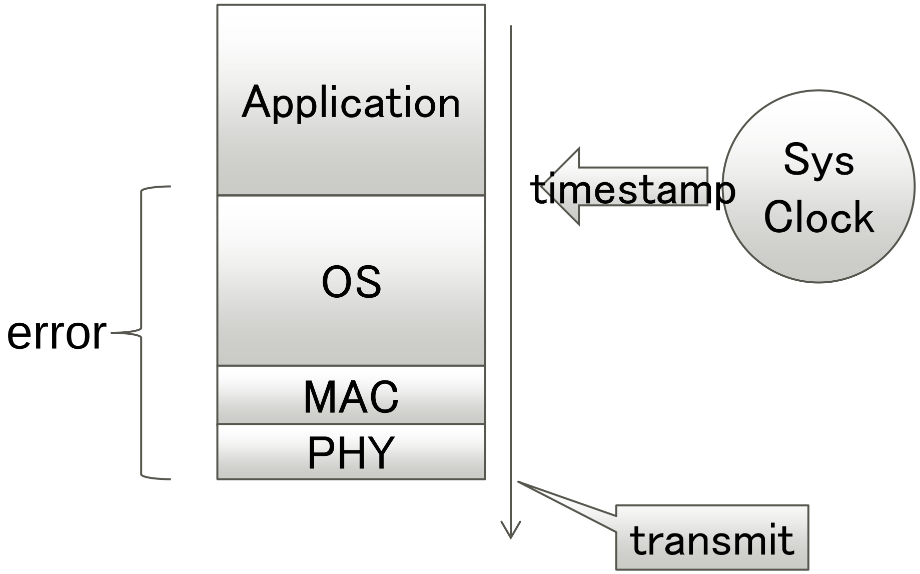
- Ideally, we want timestamps of the time just sending (or receiving) packet
  - But in reality, there is deference between timestamp timing and packet sending (or receiving) timing



Type of timestamping

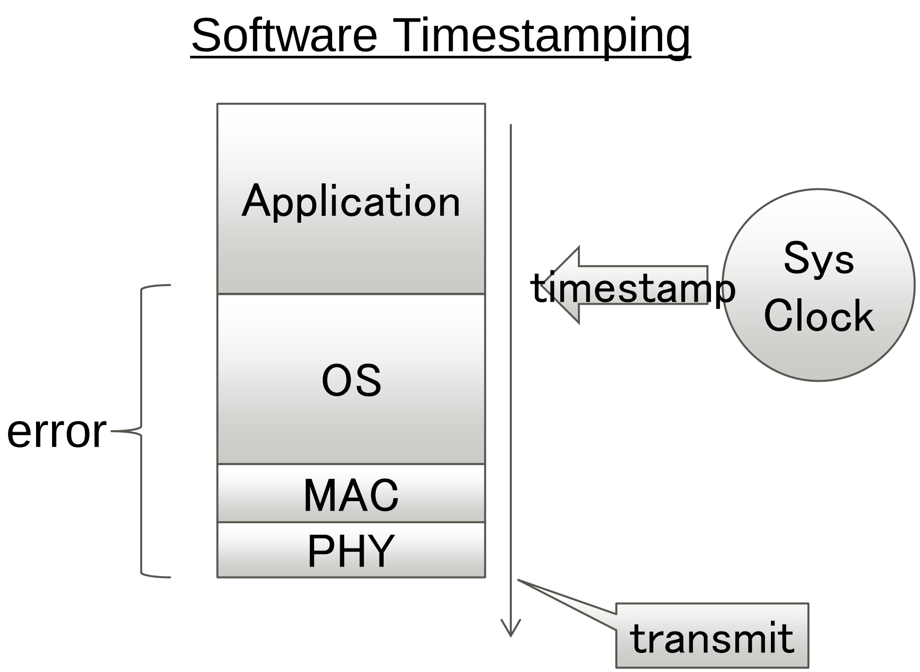
- Software timestamping
  - Timestamp at Application or OS layer
  - Get time from system clock
  - Error is relatively huge

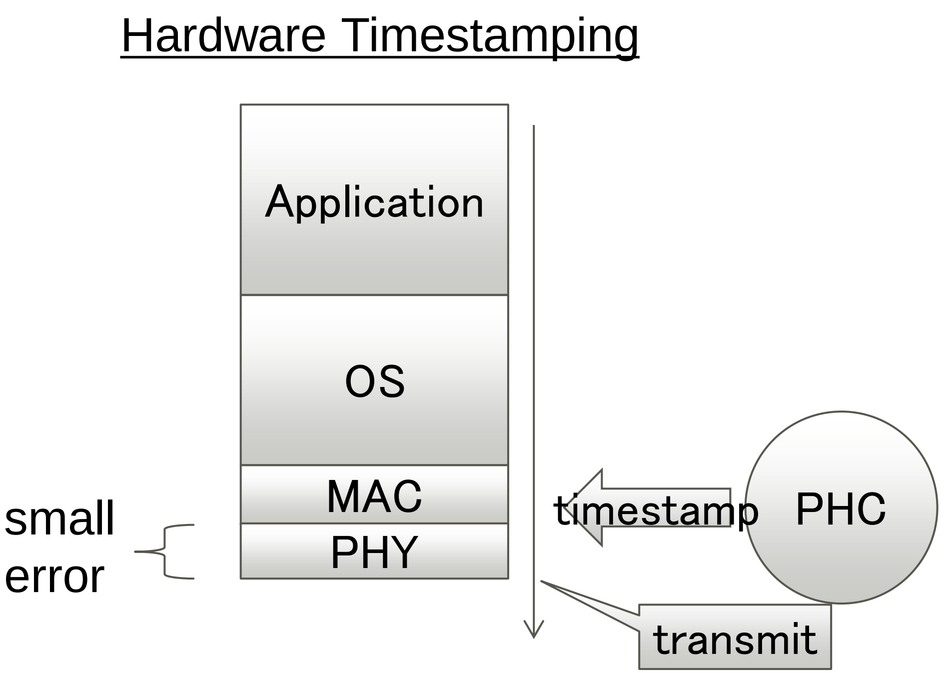
Software Timestamping



To Achieve High Precision

- Hardware Timestamping
  - Hardware assisted timestamp at PHY or MAC layer
  - Get time from PTP Hardware Clock (PHC) on NIC
  - Minimize error





Agenda

- Background
- Overview of Precision Time Protocol (PTP)
- About PTP on Linux
  - Kernel features
  - User-land application: Linuxptp
- Tips
- For easy trial or development

Linux kernel assists PTP

- The protocol itself is implemented on user-land

- Kernel features for PTP
  - Socket option SO_TIMESTAMPING for packet timestamping
  - PHC subsystem
    - Allow to access PHC via clock_gettime/settime/adjtime system calls
  - Some drivers support Hardware and/or Software timestamping (e.g. e1000e, igb, ixgbe, and so on)

The Linux PTP Project

- Project developing user-land applications for PTP
- http://linuxptp.sourceforge.net/
- Maintainer: Richard Cochran
  - He has implemented many Linux kernel features for PTP

  ⇒Linuxptp is reliable and correctly using the kernel features for PTP
- Red Hat, Intel, SUSE, Fujitsu, etc. have been participating the development

Linuxptp Applications

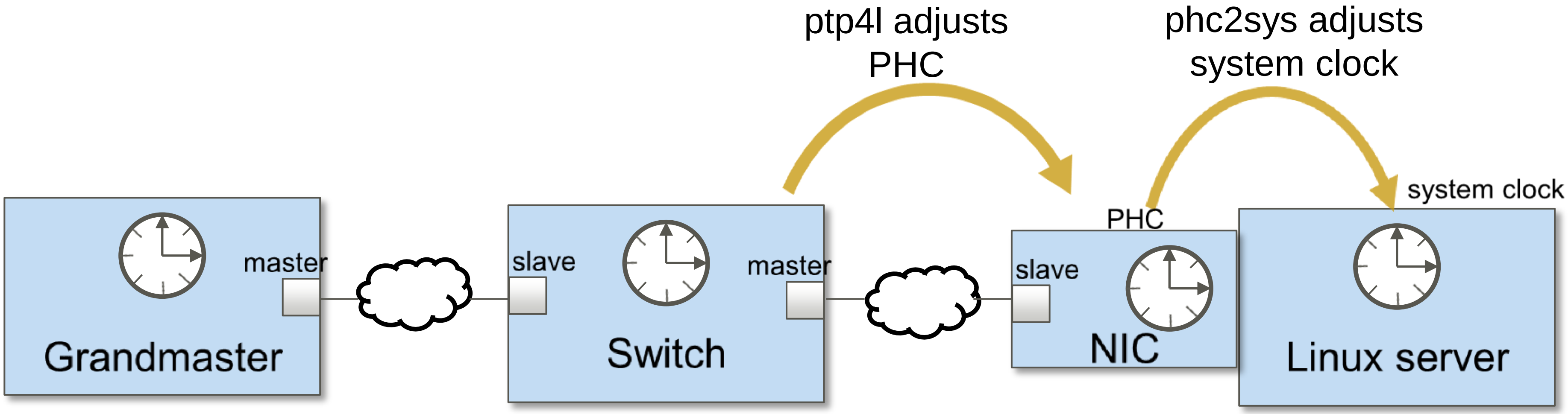
- ptp4l
  - Implementation of PTP (Ordinary Clock, Boundary Clock)
- phc2sys
  - Synchronize two clocks (typically PHC and system clock)
- pmc (PTP Management Client)
  - Send PTP management messages to PTP nodes

ptp4l

- Implementation of PTP
  - Ordinary / Boundary clock
  - Hardware / Software timestamping
  - Delay request-response / Peer delay mechanism
  - IEEE 802.3 (Ethernet) / UDP IPv4 / UDP IPv6 network transport

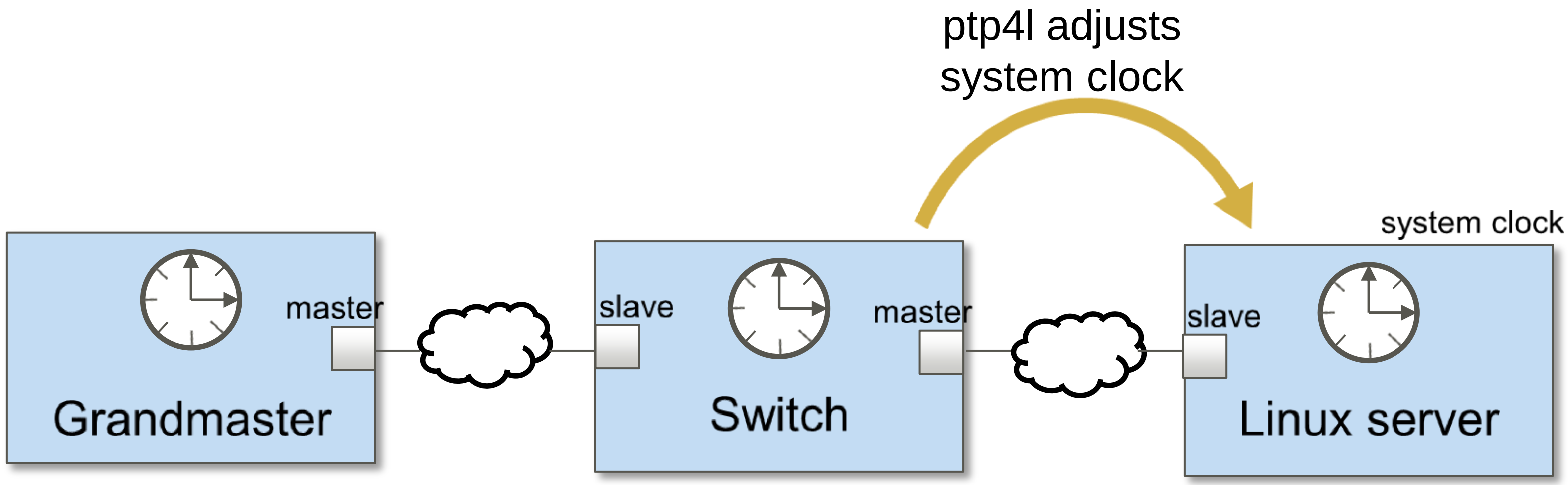
phc2sys

- Synchronize two clocks (typically PHC and system clock)
- When you are using Hardware timestamping:
  - ptp4l adjusts PHC
  - phc2sys adjusts system clock



How about software timestamping

- When you are using Software timestamping:
  - ptp4l directly adjusts system clock
  - phc2sys is not needed



Typical usage of ptp4l

FUJITSU

- Start as a slave node
- Use eth0 to send/receive messages
- Use /etc/ptp4l.conf as configuration file

Network interface to use

Specify slave only mode.
Otherwise, this node can be master.

```
# ptp4l -i eth0 -f /etc/ptp4l.conf -s
```

Specify configuration file to use.
Otherwise, default configuration is used.

Observe synchronization of ptp4l

- Log is handed over to syslog
- Or, you can print it into stdout by using –m option

```
ptp4l[537888.171]: selected /dev/ptp3 as PTP clock
ptp4l[537888.173]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[537888.174]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[537889.091]: port 1: new foreign master 001999.fffe.807b24-1
ptp4l[537893.091]: selected best master clock 001999.fffe.807b24
ptp4l[537893.091]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[537894.093]: master offset      -3318 s0 freq      +0 path delay       600
ptp4l[537895.093]: master offset      -3343 s1 freq   -8378 path delay       600
ptp4l[537896.093]: master offset      -2344 s2 freq  -10722 path delay       600
ptp4l[537896.093]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[537897.093]: master offset        -18 s2 freq   -9099 path delay       545
ptp4l[537898.093]: master offset        641 s2 freq   -8446 path delay       513
ptp4l[537899.093]: master offset        570 s2 freq   -8324 path delay       533
ptp4l[537900.093]: master offset        389 s2 freq   -8334 path delay       533
```

Offset between Master and Slave(PHC)

Typical usage of phc2sys

FUJITSU

- Adjust system clock based on eth0's PHC
- Wait until ptp4l starts synchronization to the master

By specifying network interface to -s option, related PHC is automatically selected.
Or, you can directly specify PHC like -s /dev/ptp0

Wait until ptp4l's synchronization.

```
# phc2sys -s eth0 -c CLOCK_REALTIME -w
```

Specify the clock you want to adjust.
CLOCK_REALTIME is system clock.

pmc (PTP Management Client)

- Send PTP management messages to PTP nodes
  - GET action: Get current values of data
  - SET action: Update current values of variables
  - CMD action: Initiate some events
- PTP management messages are specified in IEEE1588
- Many PTP devices have not supported management messages yet
  - Also linuxptp has not supported many SET and CMD messages yet

Typical usage of pmc

- Send a message to localhost's node
- Get values of CURRENT_DATA_SET

Indicate to use Unix Domain Socket.
UDS is used to receive PTP management messages from localhost.

Action and Management ID.

```
# pmc -u -b 0 'GET CURRENT_DATA_SET'
```

-b specifies allowance number of boundary hops.
In this case, management messages is sent only localhost.

An example of synchronization between Linux servers

- Directly connect two Linux servers (Grandmaster and Slave)
- Use hardware timestamping

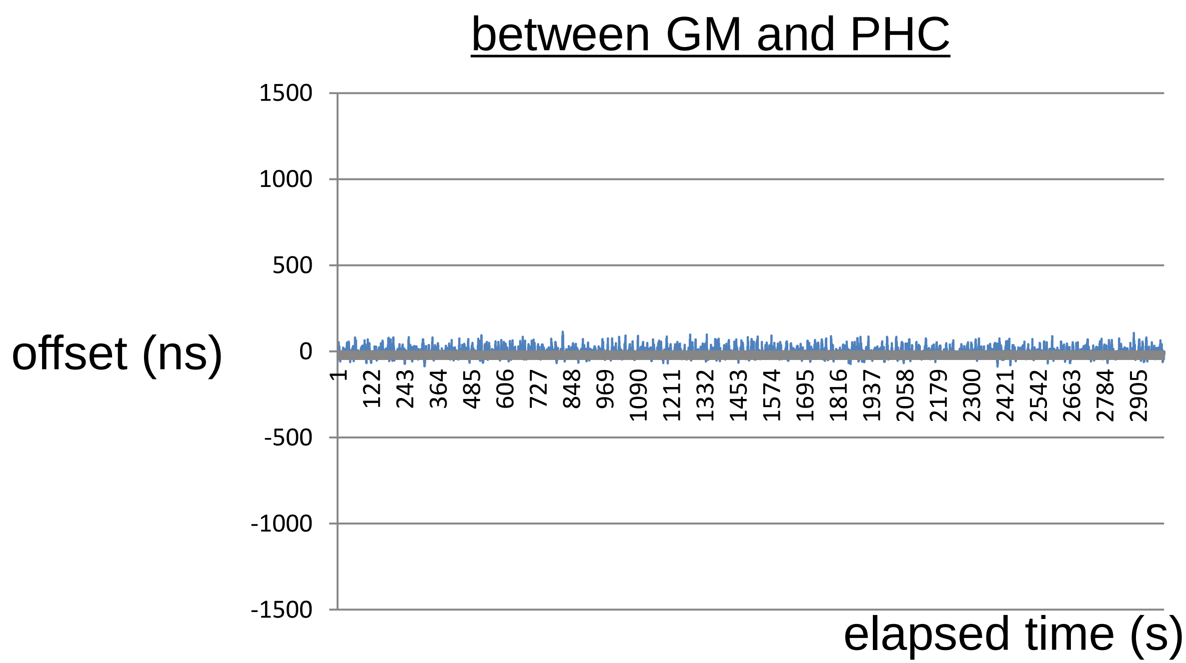
Setup

An example of synchronization between Linux servers

- The offsets between the PHCs, observed by ptp4l

max 116 ns
min -89 ns
RMS 31.19 ns

very stable!



Agenda

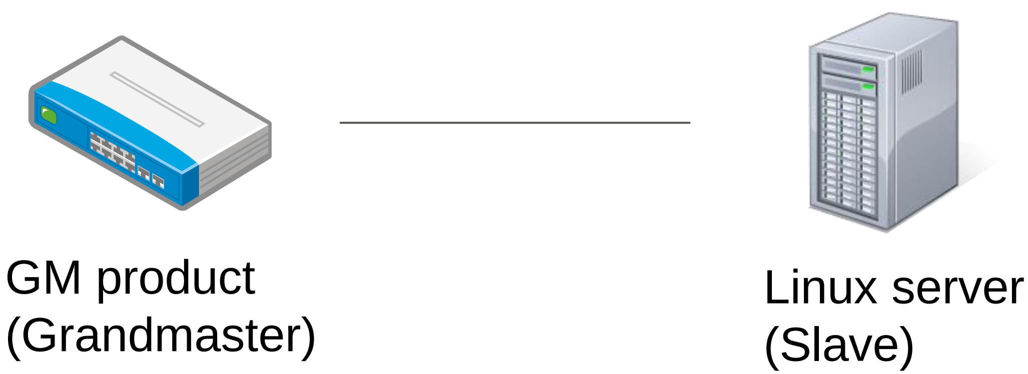
FUJITSU

- Background
- Overview of Precision Time Protocol (PTP)
- About PTP on Linux
- Tips
  - Workaround against bad GM behavior
  - Improve system clock stability
- For easy trial or development

Bad GM behavior

- I encountered a Grandmaster product
- The GM sometimes occurs a few hundred microsecond level errors

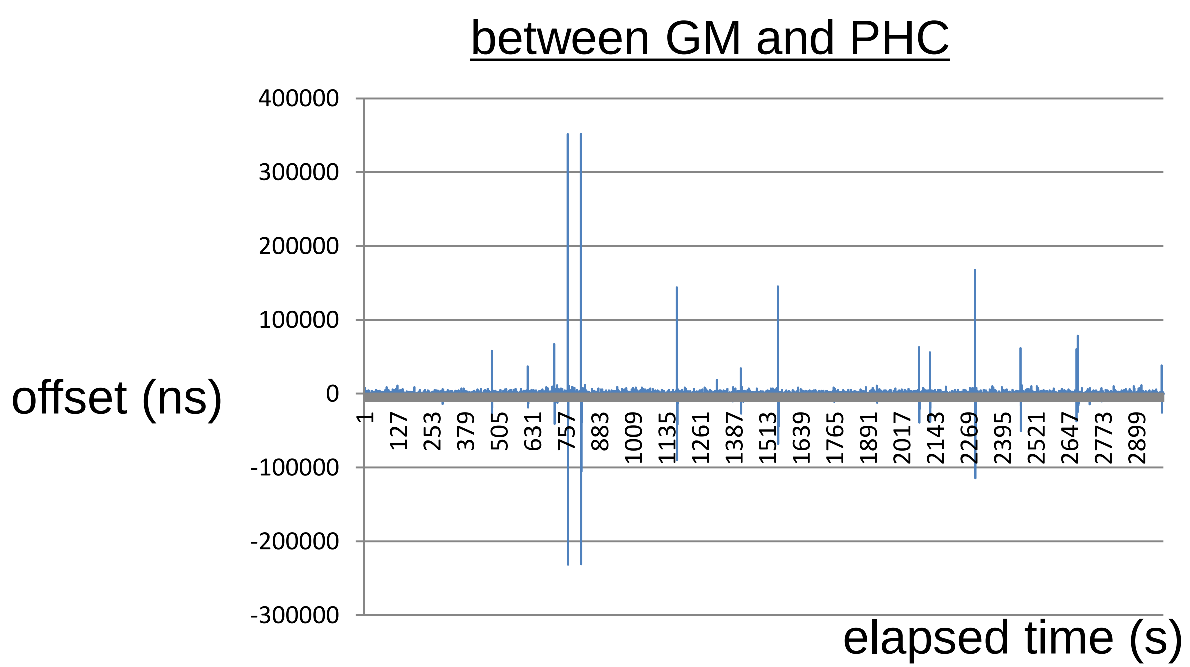
Setup



Bad GM behavior

- The offsets between the GM and the Linux server's PHC, observed by ptp4l

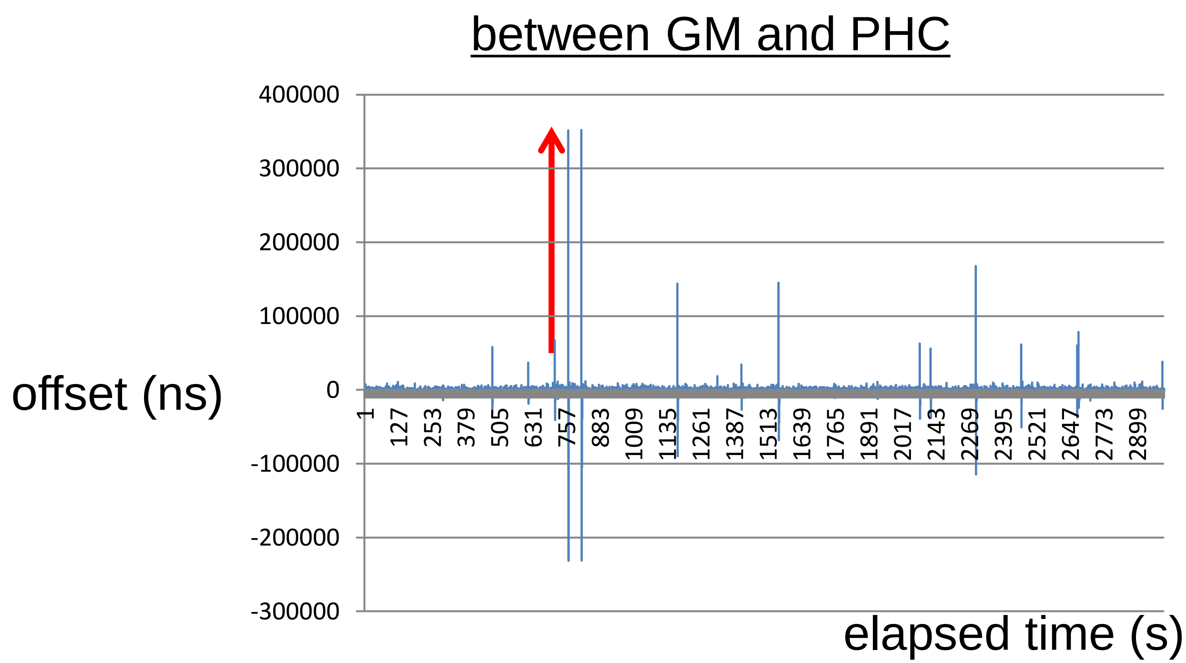
max 352129 ns (= 352 us)
min -231644 ns (= -232 us)
RMS 13664.71 ns (= 14 us)



What's happening?

FUJITSU

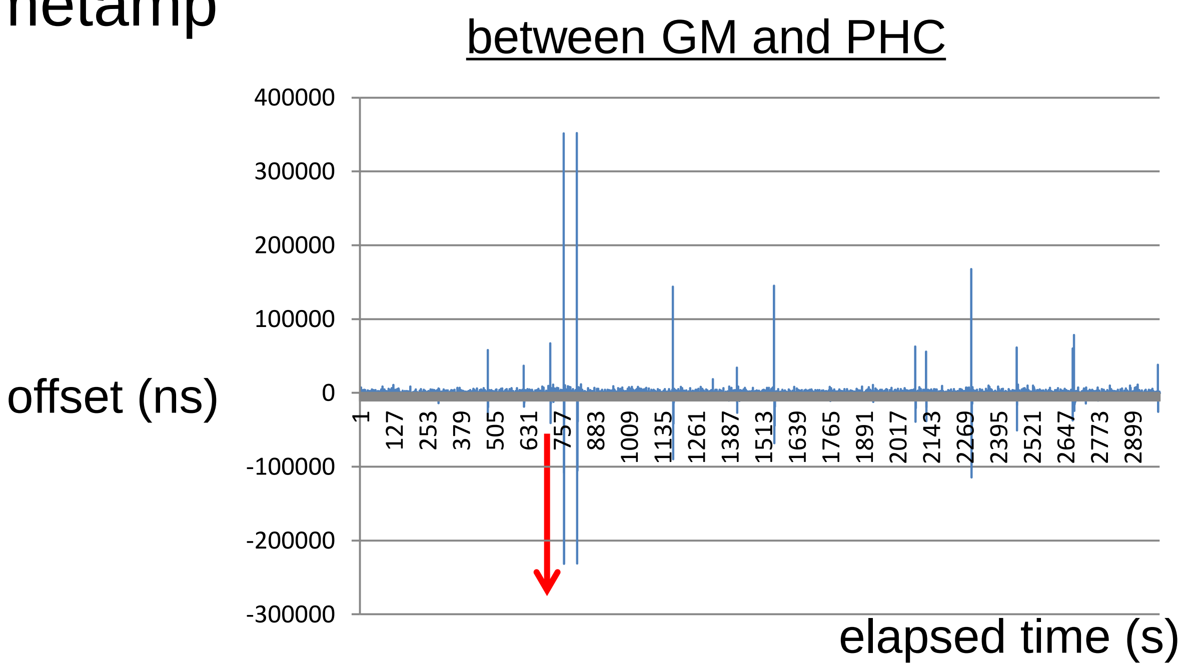
- GM sends a timestamp including huge error
  - It appears as a huge plus offset
- ptp4l changes PHC's frequency too much depending on the offset



What's happening?

- GM's next timestamp does not include so much error
  - But, PHC's frequency was changed too much

⇒ A huge minus offset appears against normal GM's timetamp



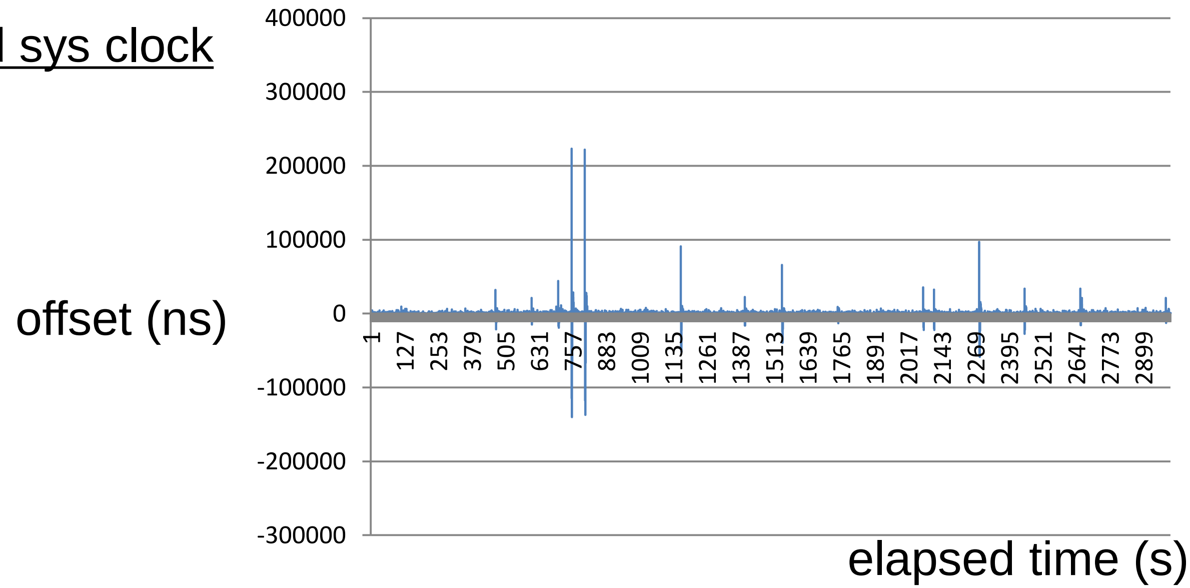
Observe the influence to PHC

FUJITSU

- The offsets between PHC and system clock observed by phc2sys
  - There are similar offsets

⇒Introduce worse system clock stability

between PHC and sys clock

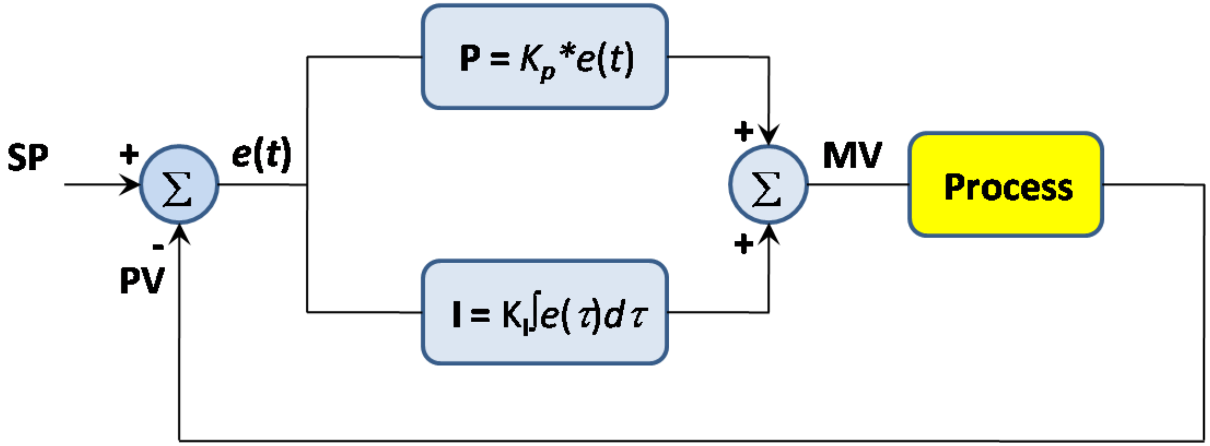


- How to avoid this issue?

ptp4l has servo mechanism

FUJITSU

- ptp4l has PI (proportional-integral) controller servo
  - A kind of feedback loop
  - Determine frequency set to PHC



SP = master's time
PV = slave's time
e(t) = offset
MV = frequency
Process =
change slave's clock freq

Figure: Basic block of Proportional + Integral controller. (excerpt from wikipedia)

  - Kp and Ki are tuning parameters (proportional gain and integral gain)

Tuning sensitivity

- Anyway, by tuning the servo parameters, we can adjust how clock sensitivity react to the offset
- To change the configuration, edit ptp4l's configuration file
  - pi_proportional_const for Kp
  - pi_integral_const for Ki

Default configuration file is `/etc/ptp4l.conf` in Fedora

There are two default configurations

- For hardware timestamping
  - Kp 0.7
  - Ki 0.3

  ⇒Sensitive

  ← Previous result used this configuration

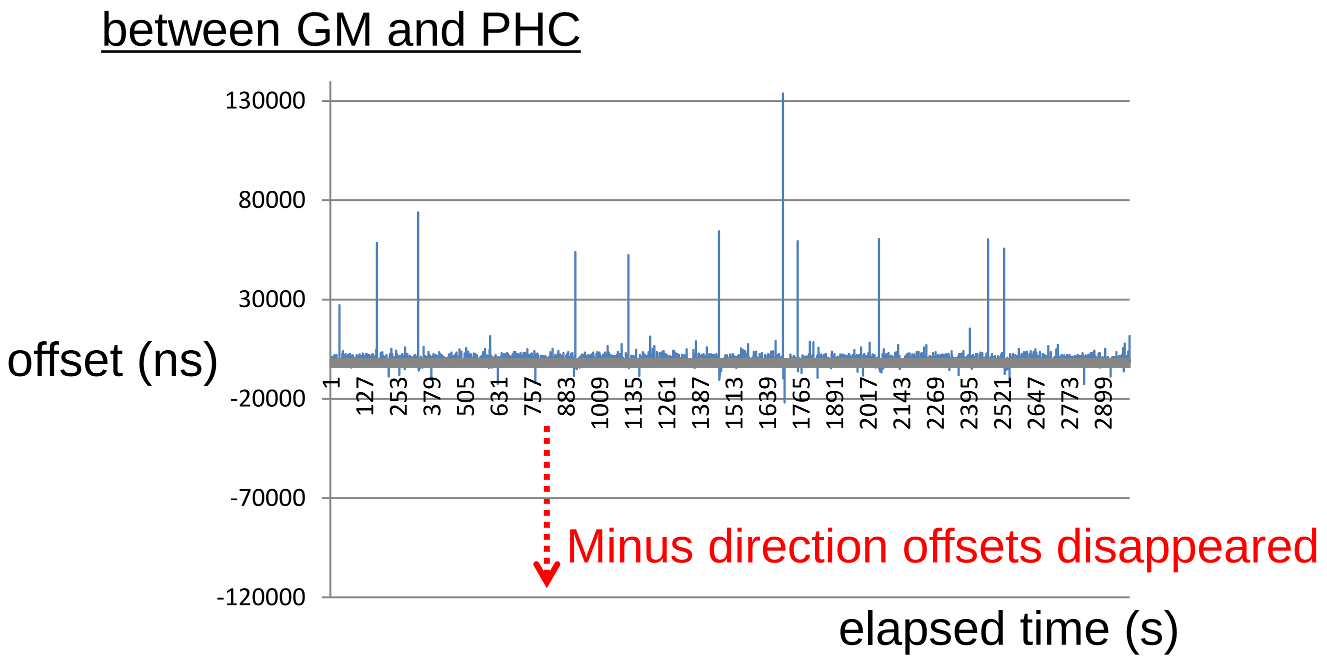
- For software timestamping
  - Kp 0.1
  - Ki 0.001

  ⇒Insensitive

  ← Try this one to prevent over-reacting

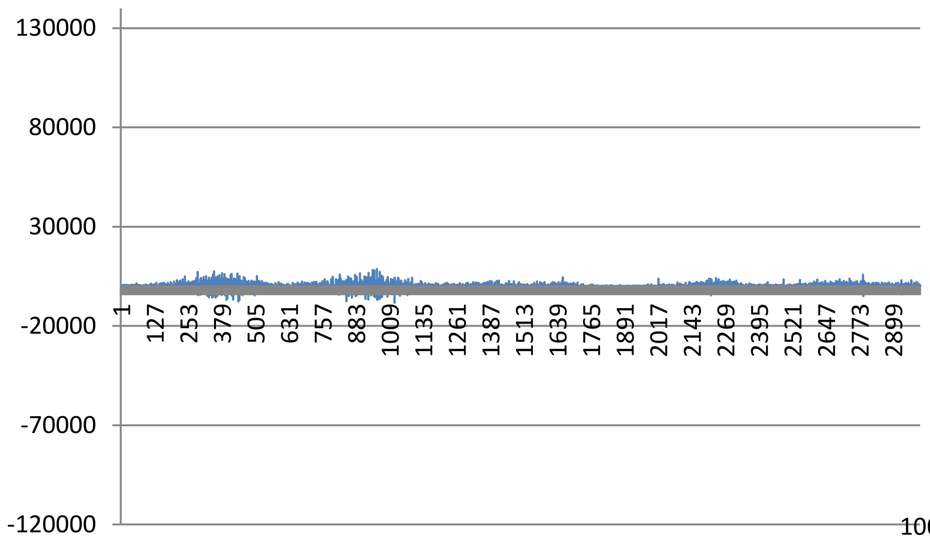
Use software timestamping config

- We tried software timestamping configuration though ptp4l used hardware timestamping
- Minus direction offsets disappeared
  - PHC's frequency is not changed too much



Observe from phc2sys

FUJITSU



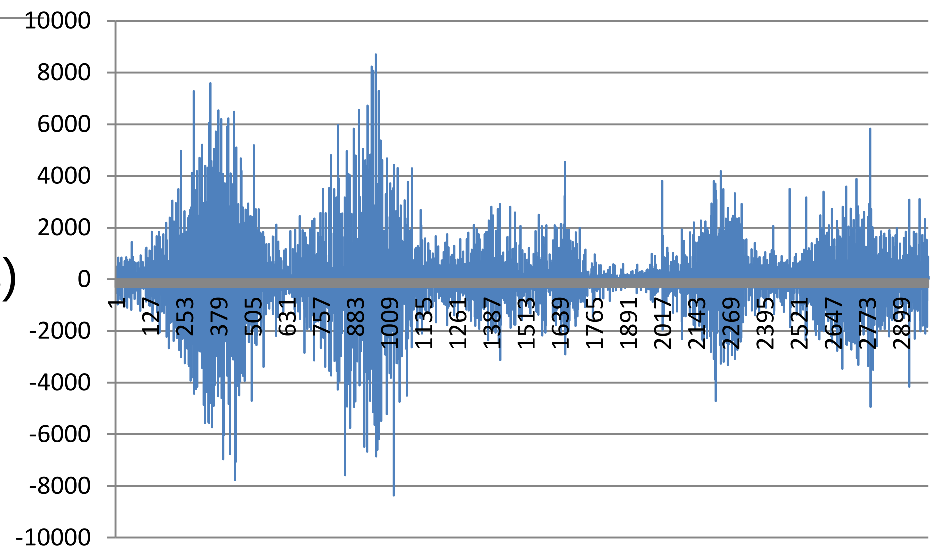
- PHC looks stable

⇒system clock will be also stable

Zoom

offset (ns)

between PHC and sys clock



elapsed time (s)

Agenda

- Background
- Overview of Precision Time Protocol (PTP)
- About PTP on Linux
- Tips
  - Countermeasure against bad GM behavior
  - Improve system clock stability
- For easy trial or development

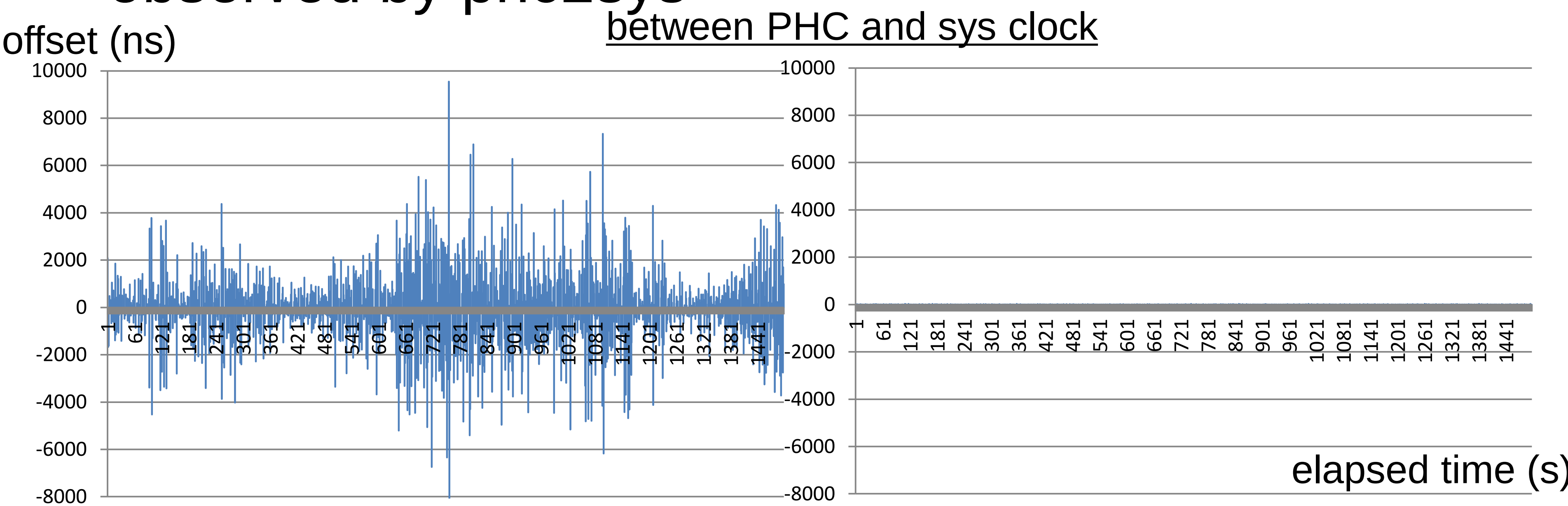
Improve system clock stability

- Dynamic ticks make system clock stability worse
  - Dynamic ticks disable periodic timer tick interrupt
  - It is a useful feature to power saving but…
  - Error correction mechanism in kernel doesn't aware dynamic ticks
- You can disable dynamic ticks
  - Specify nohz=off in kernel boot option
  - (nohz=on is default)

Comparison

FUJITSU

- The offsets between PHC and system clock, observed by phc2sys



nohz=on

max 9550 ns

min -8134 ns

RMS 1589.3 ns

nohz=off

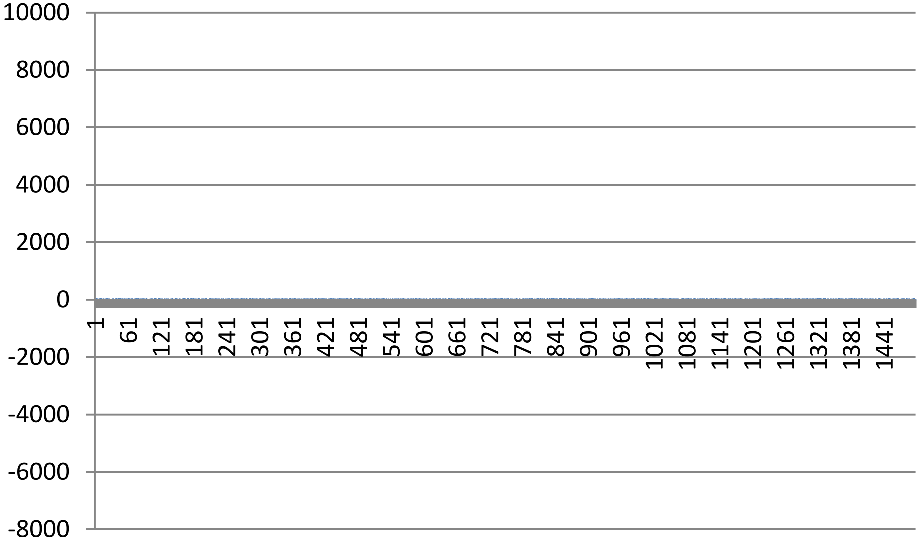
max 32 ns

min -26 ns

RMS 9.3 ns

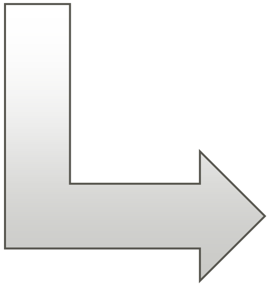
1000 times better!

Zoom graph of nohz=off



<u>nohz=off</u>

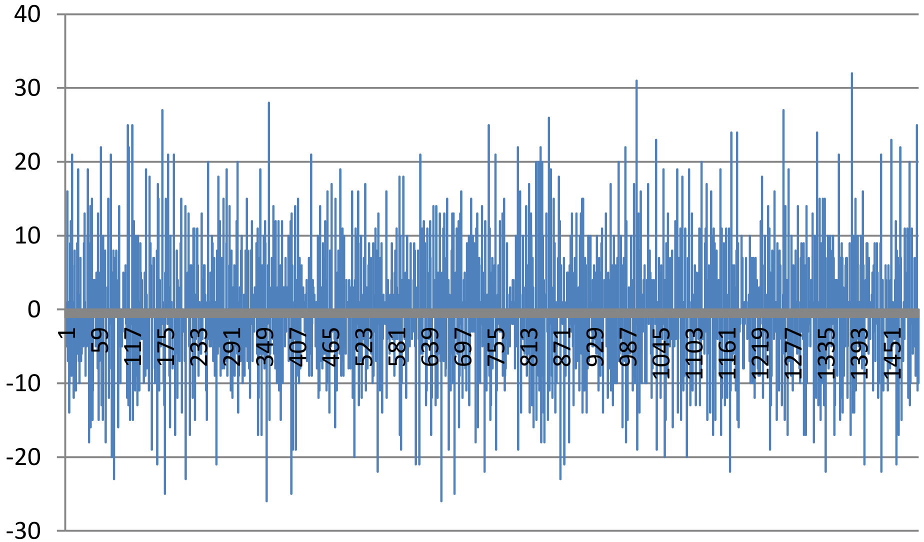
max 32 ns
min -26 ns
RMS 9.3 ns



Zoom

<u>between PHC and sys clock</u>

offset (ns)



elapsed time (s)

Attempt to fix the issue on upstream

- Miroslav Lichvar and John Stultz are working to fix the issue

	title	author	date
1st patch	"[PATCH RFC] timekeeping: Fix clock stability with nohz"	Miroslav	Oct 2013
2nd patch	"[PATCH] [RFC] timekeeping: Rework frequency adjustments to work better w/ nohz"	John	Jan 2014
3rd patch	"[PATCH 0/3] timekeeping: Improved NOHZ frequency steering"	John	Apr 2014

Agenda

- Background
- Overview of Precision Time Protocol (PTP)
- About PTP on Linux
- Tips
  - Countermeasure against bad GM behavior
  - Improve system clock stability
- For easy trial or development
  - Try linuxptp on qemu-kvm

FYI: Running linuxptp on qemu-kvm

FUJITSU

- You can try linuxptp on qemu-kvm without NICs supporting PTP
- Note:
  - Run two Virtual Machines (for GM and Slave)
  - Recommend to use OS supporting linuxptp
    - e.g. recent Fedora, RHEL6.5, 7.0
  - Use virtual NIC emulating e1000
    - e1000 driver supports software timestamping
  - Don't forget to define appropriate firewall rules (or disable firewalls) to allow multi-cast
  - Don't expect high precision and accuracy

FUJITSU

shaping tomorrow with you