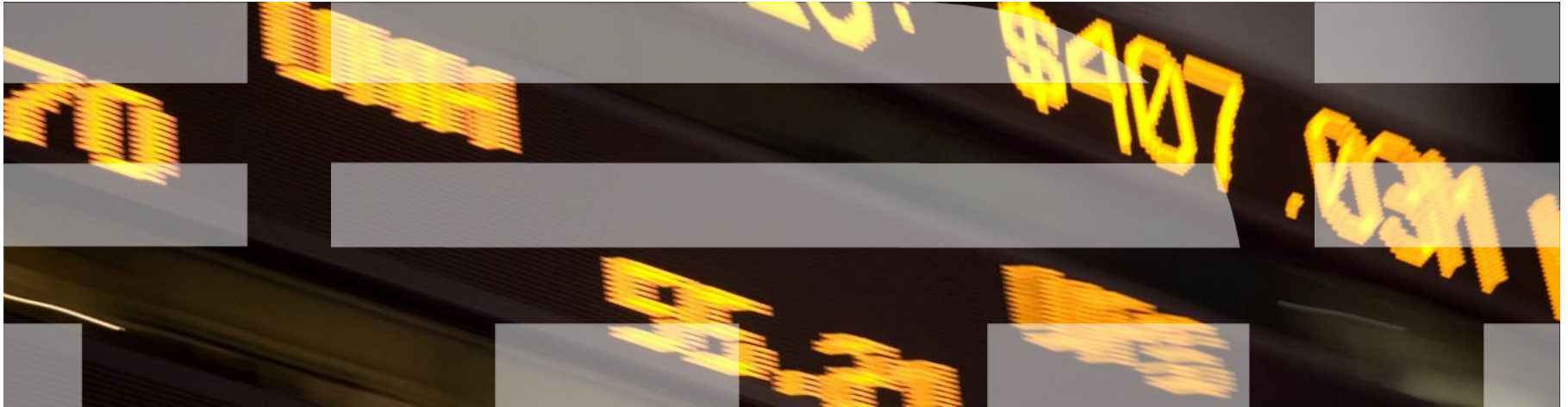IBM

Precision Time Protocol, and Sub-Microsecond Synchronization

Mike Kravetz
IBM Linux Technology Center
kravetz@us.ibm.com

Agenda

- Background/History leading to Precision Time Protocol

- Results of Experimentation

- Building out a Complete PTP Solution

IBM LTC Low Latency Team Involvement With PTP

Customer Queries

RealTime Linux Suitable for Financial Environments?

High Frequency Trading?

IBM LTC RealTime Team

Real requirement is Low Latency, rather than RealTime
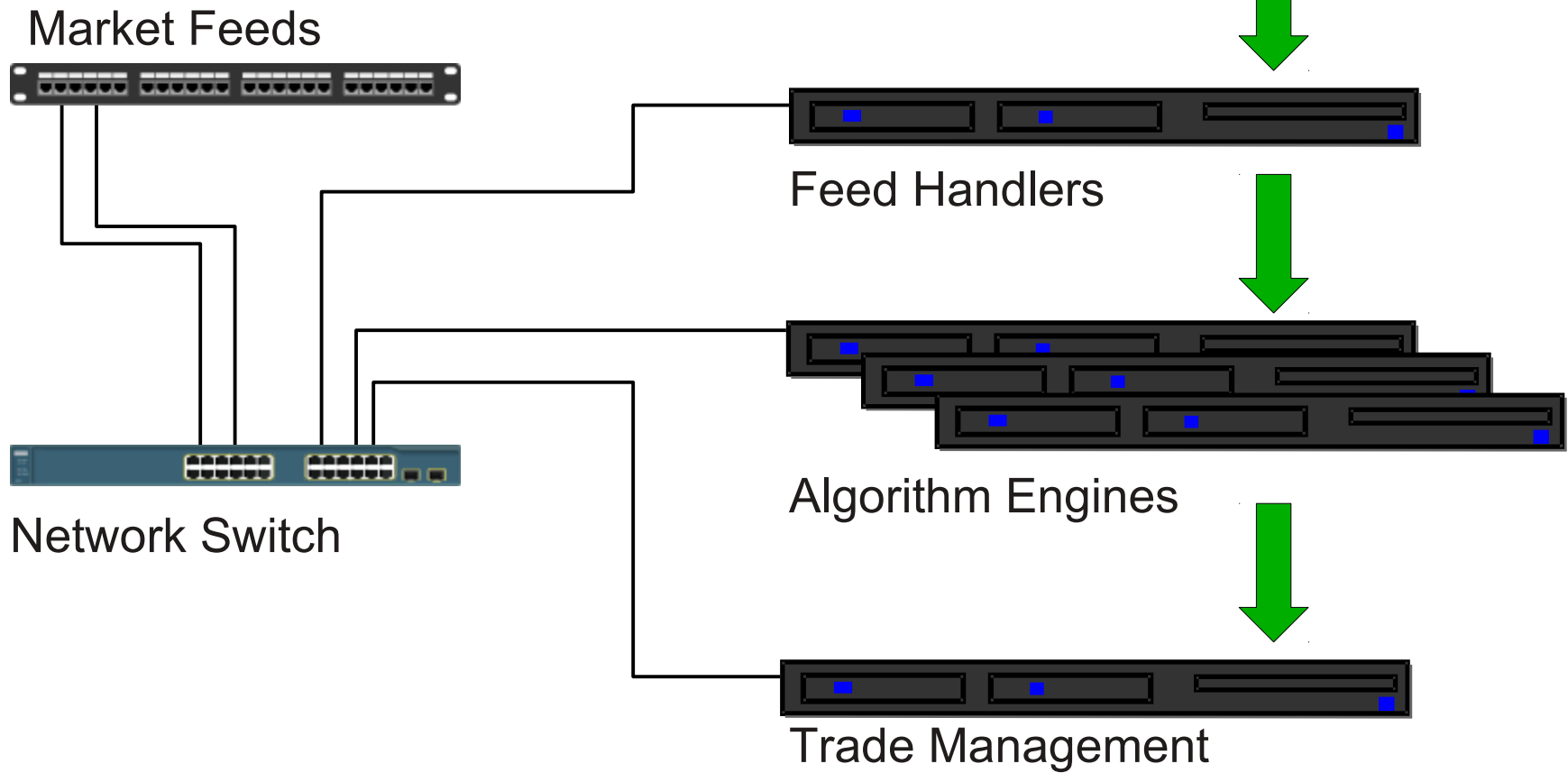
Customer Queries

Time Synchronization in Financial Environments?

IBM Low Latency Team

Exploration of Precision Time Protocol

High Frequency Trading Environment

Market Feeds

Feed Handlers

Network Switch

Algorithm Engines

Trade Management
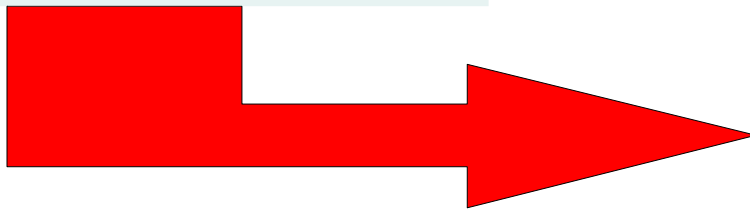
Time Synchronization

Current Methods

Synced External Clocks
- GPS or Similar time source attached to each system
- New Connection Infrastructure
- New Management Issues
- Expense

Network Time Protocol – NTP
- Linux/Unix Standard
  - Works "everywhere"
- Designed for WANs
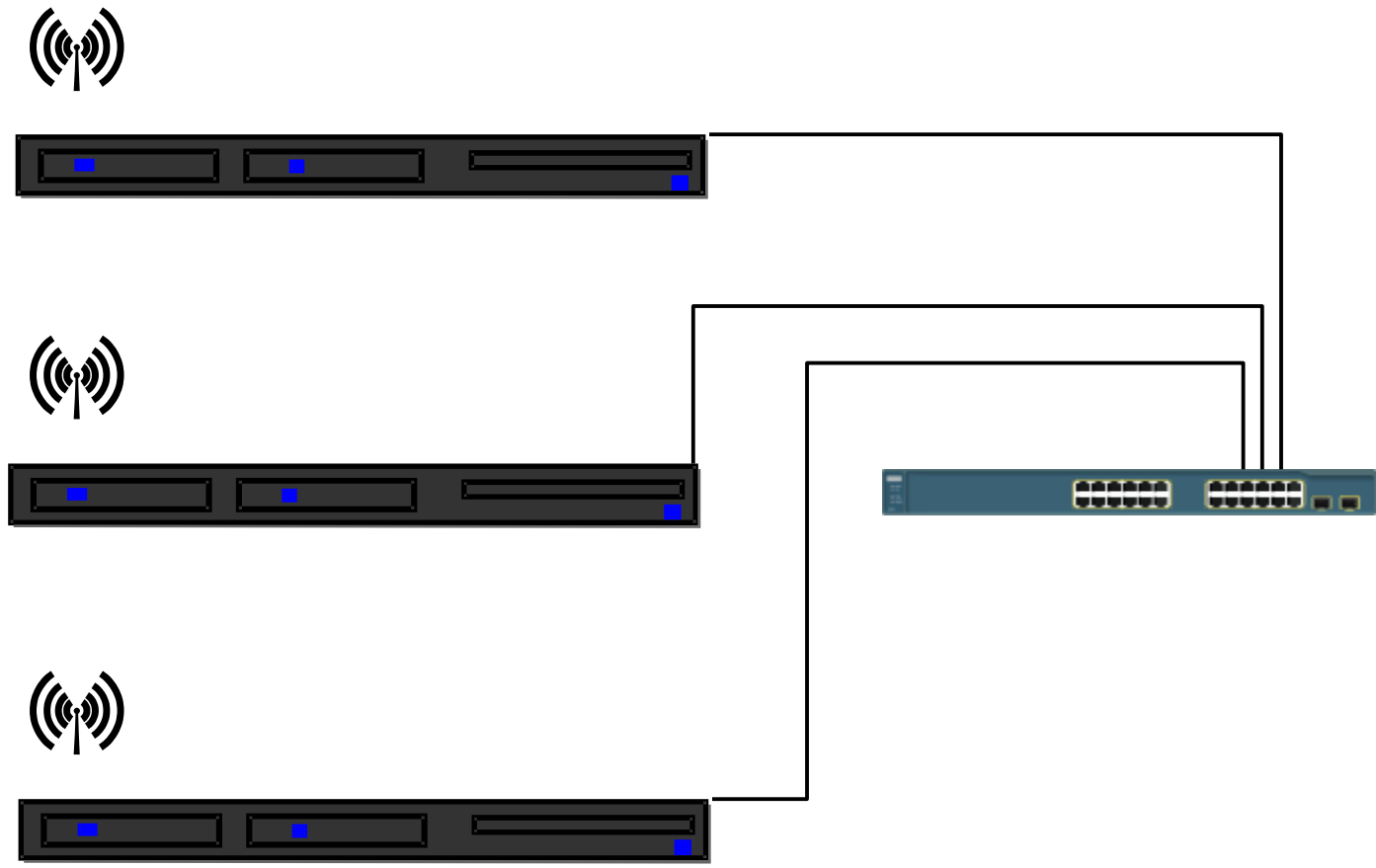- Sync Accuracy may not be good enough for Low Lat environments
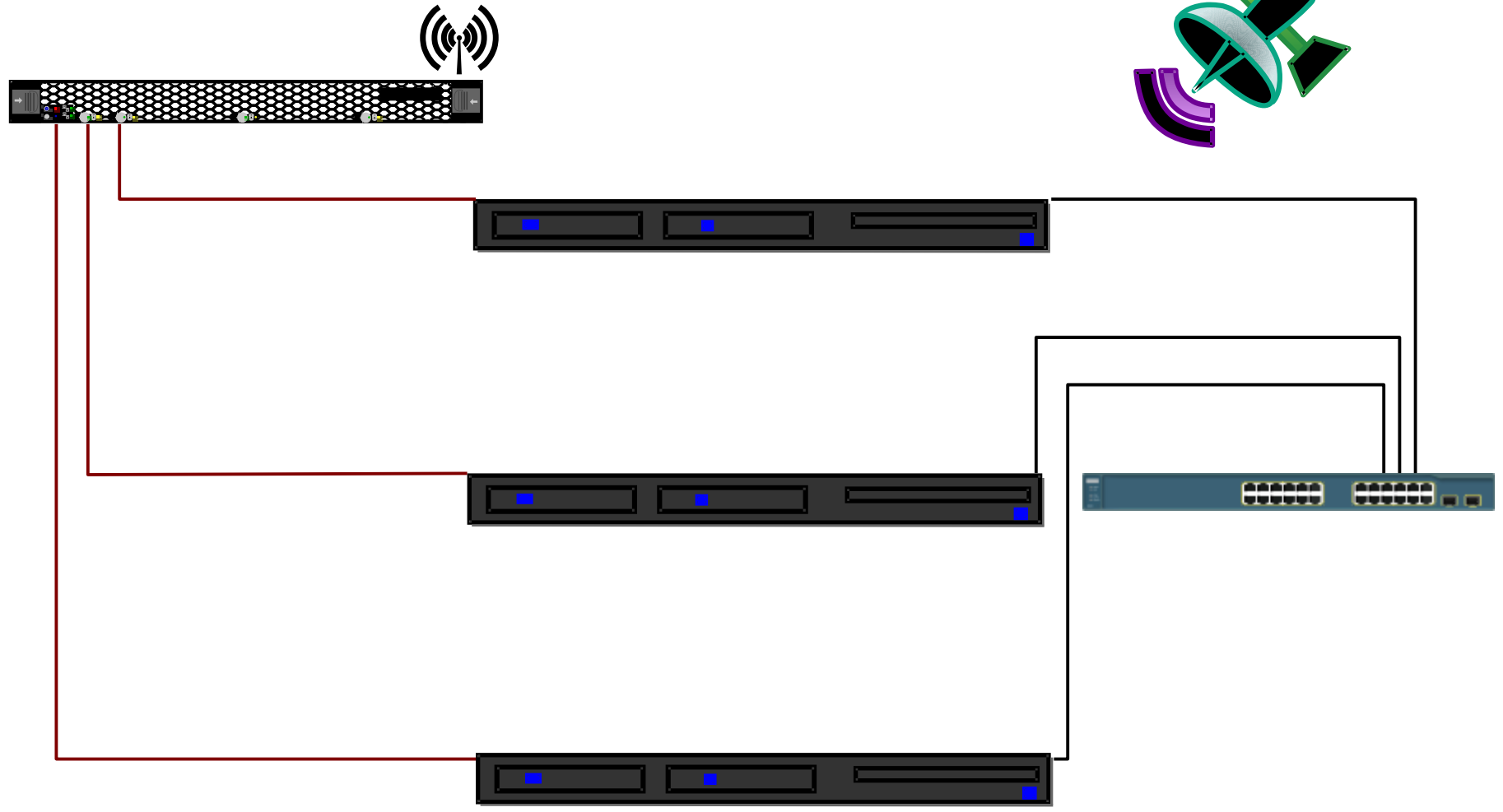
Emerging Method

Precision Time Protocol - PTP
- Uses existing Network Infrastructure
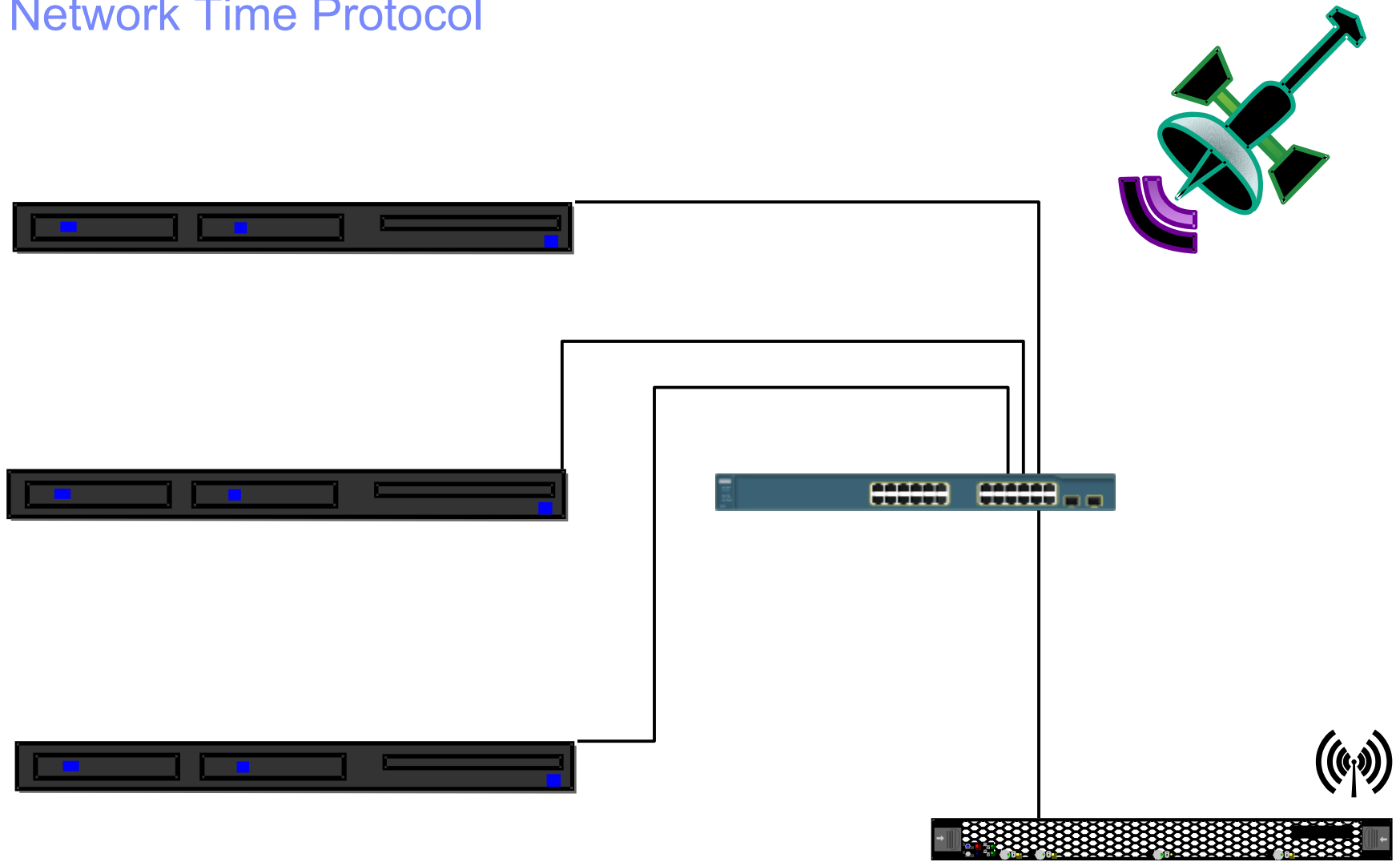- Designed for LANs
- Targeting sub-microsecond accuracy

GPS for Time Synchronization (Receiver per-System)

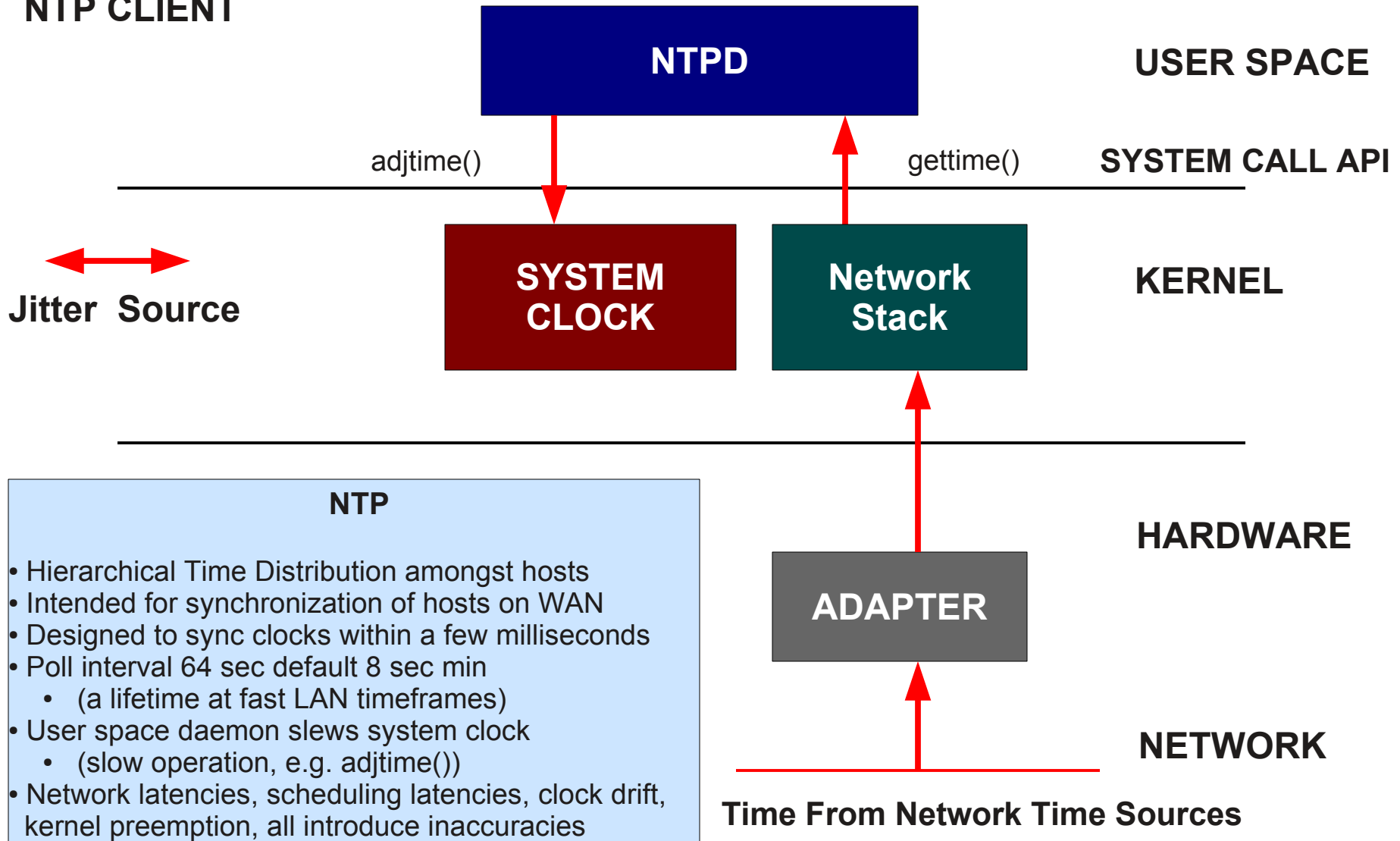# GPS for Time Synchronization (Distribute Signal)
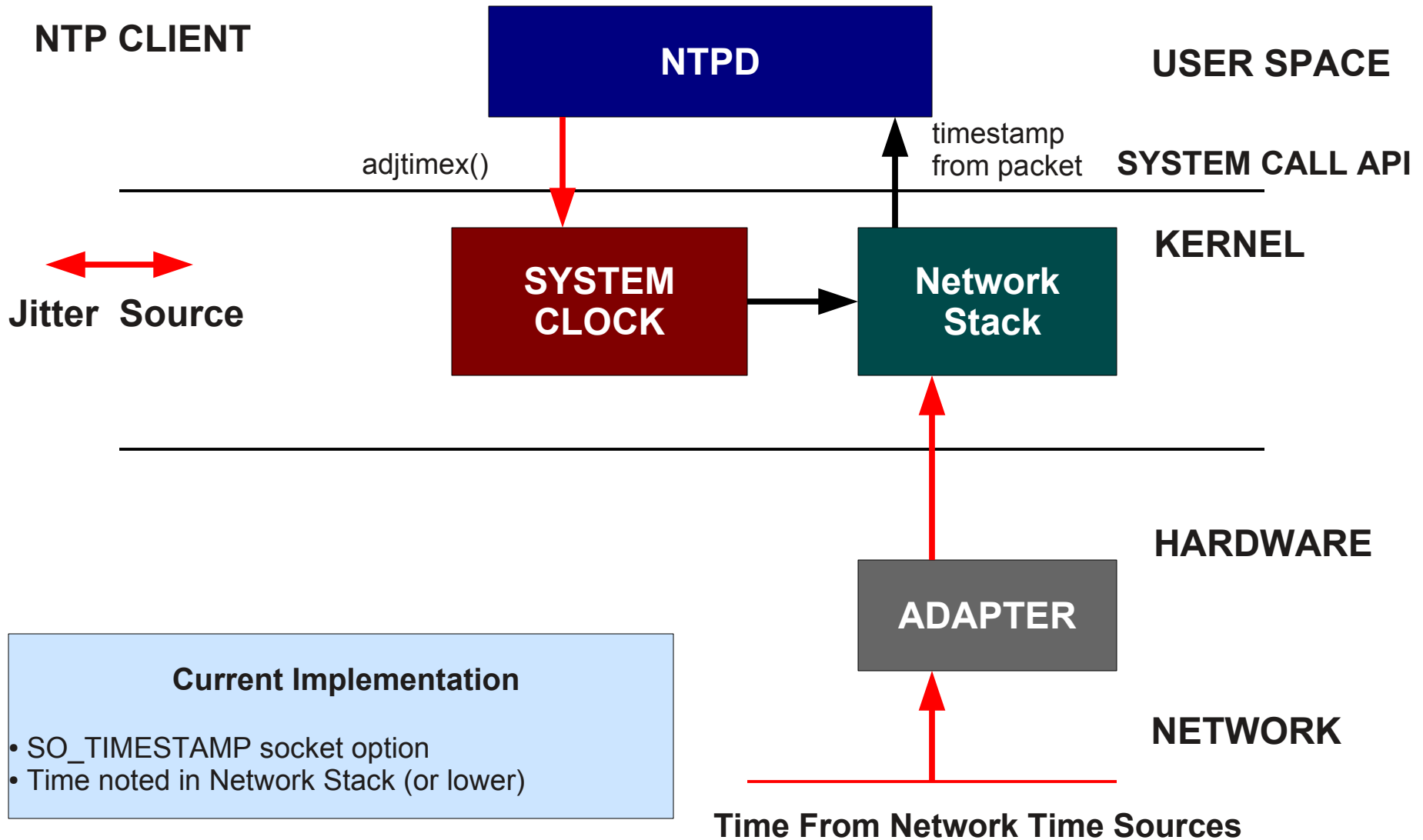
Network Time Protocol

Network Time Protocol – The Original Standard

NTP CLIENT

USER SPACE

NTPD

adjtime()   **SYSTEM CALL API**   gettime()

Jitter Source

SYSTEM CLOCK

Network Stack

KERNEL

HARDWARE

ADAPTER

NETWORK

NTP

- Hierarchical Time Distribution amongst hosts
- Intended for synchronization of hosts on WAN
- Designed to sync clocks within a few milliseconds
- Poll interval 64 sec default 8 sec min
  - (a lifetime at fast LAN timeframes)
- User space daemon slews system clock
  - (slow operation, e.g. adjtime())
- Network latencies, scheduling latencies, clock drift, kernel preemption, all introduce inaccuracies

Time From Network Time Sources

Network Time Protocol – The Current Standard

IBM

NTP CLIENT

NTPD

USER SPACE

adjtimex()

timestamp from packet

SYSTEM CALL API

Jitter Source

SYSTEM CLOCK

Network Stack

KERNEL

HARDWARE

ADAPTER

NETWORK

Current Implementation

• SO_TIMESTAMP socket option
• Time noted in Network Stack (or lower)

Time From Network Time Sources

© 2012 IBM Corporation

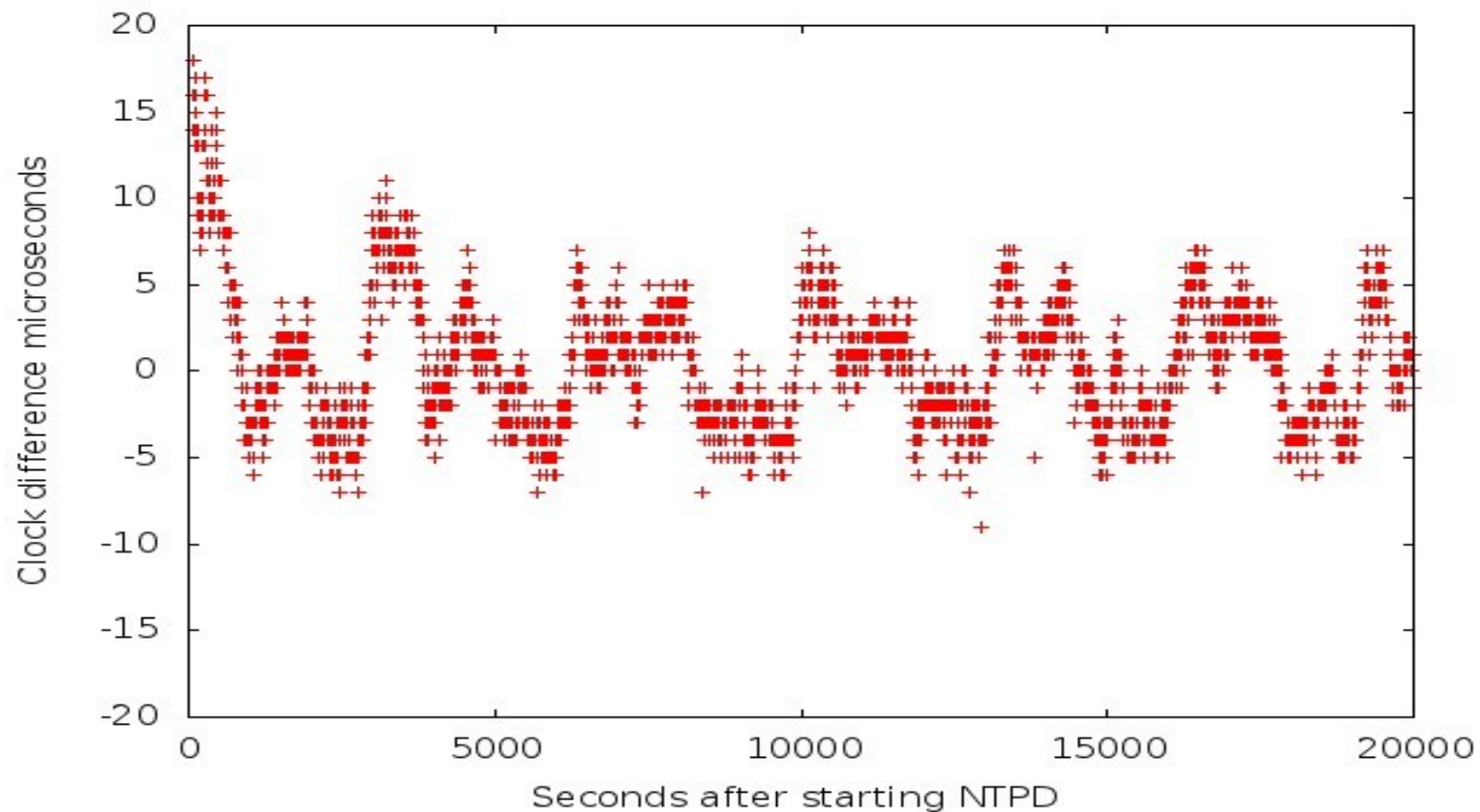# Measuring Clock Deltas – A Example of Perfect
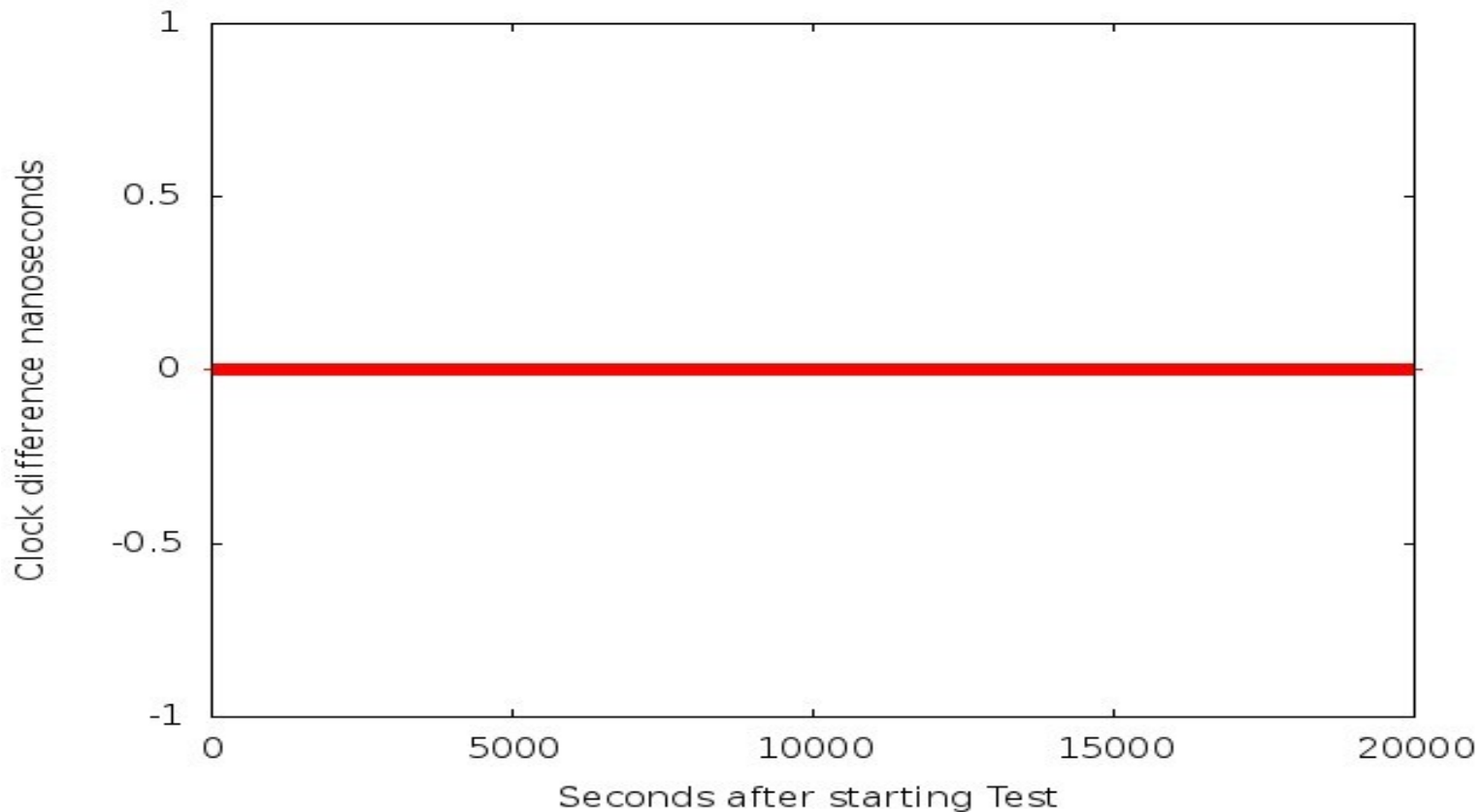
Stats

- Beginning 15 minutes after start
- Range : 0 to 0 nanoseconds
- Mean : 0.0
- StdDev: 0.0

Environment

- Protocol Used
- Load

Network Time Protocol – LAN (minimal load)

Stats

- Beginning 15 minutes after start
- Range : -9 to 11 microseconds
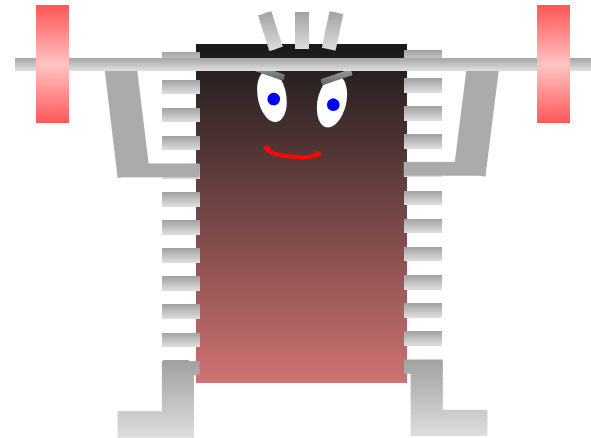- Mean   : 0.154792
- StdDev: 3.352142

Environment

- Simple 10Gb LAN for NTP traffic
- Poll interval 8 sec
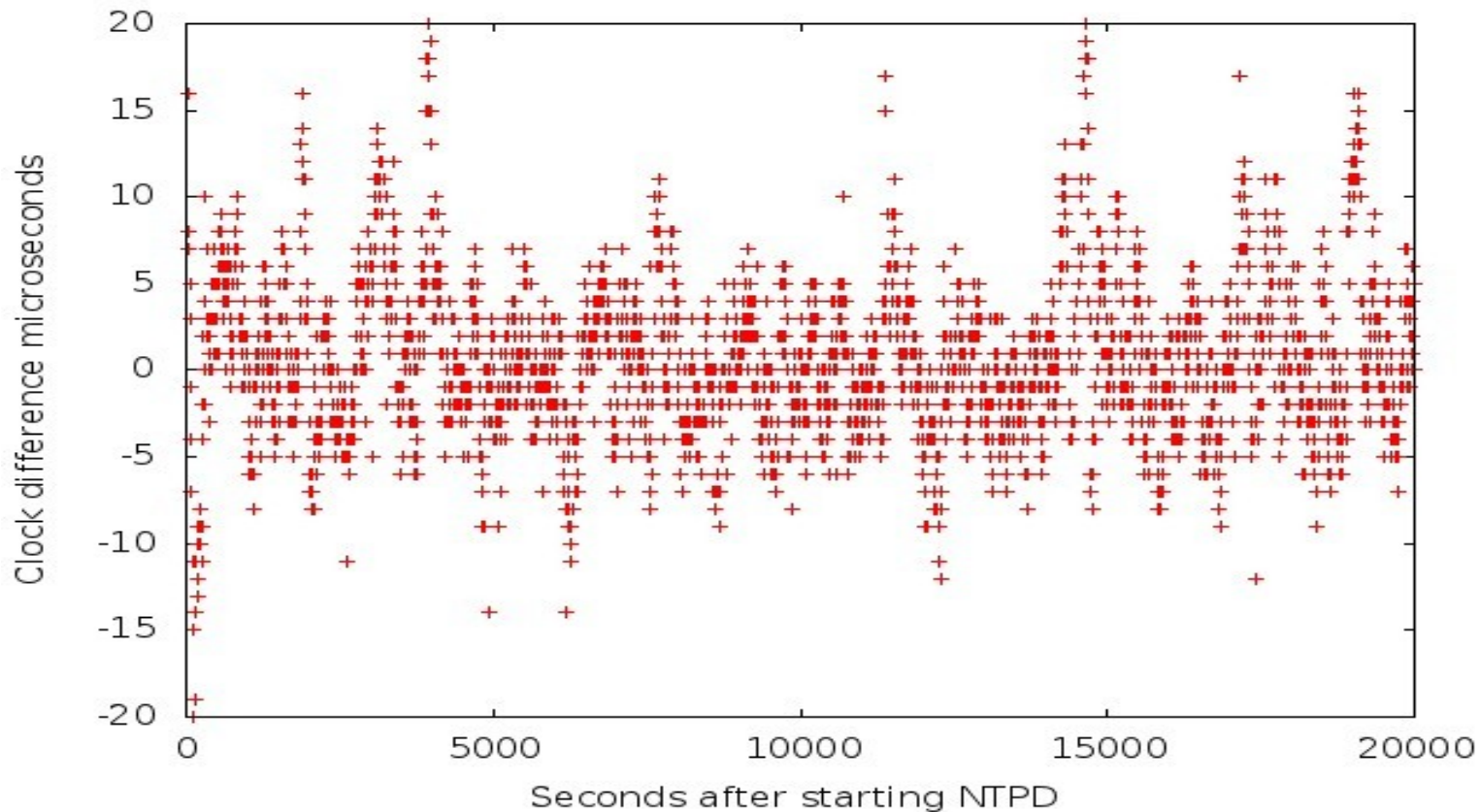- Minimal CPU and Network Activity

Adding CPU Load

Kernel Build

```
while true; do
      make clean
      make -j 32
done
```
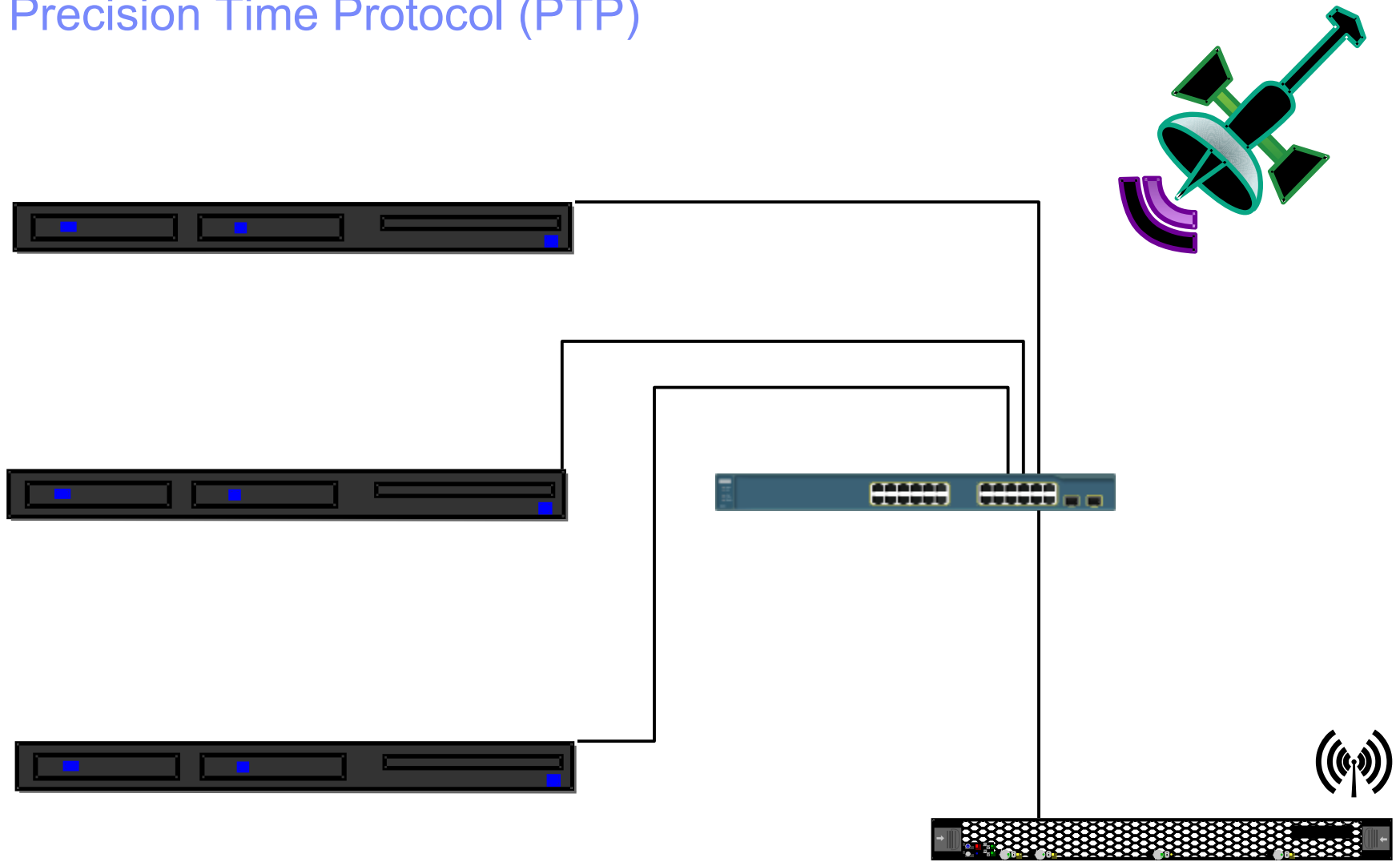
Network Time Protocol – LAN (CPU load)

Stats

- Beginning 15 minutes after start
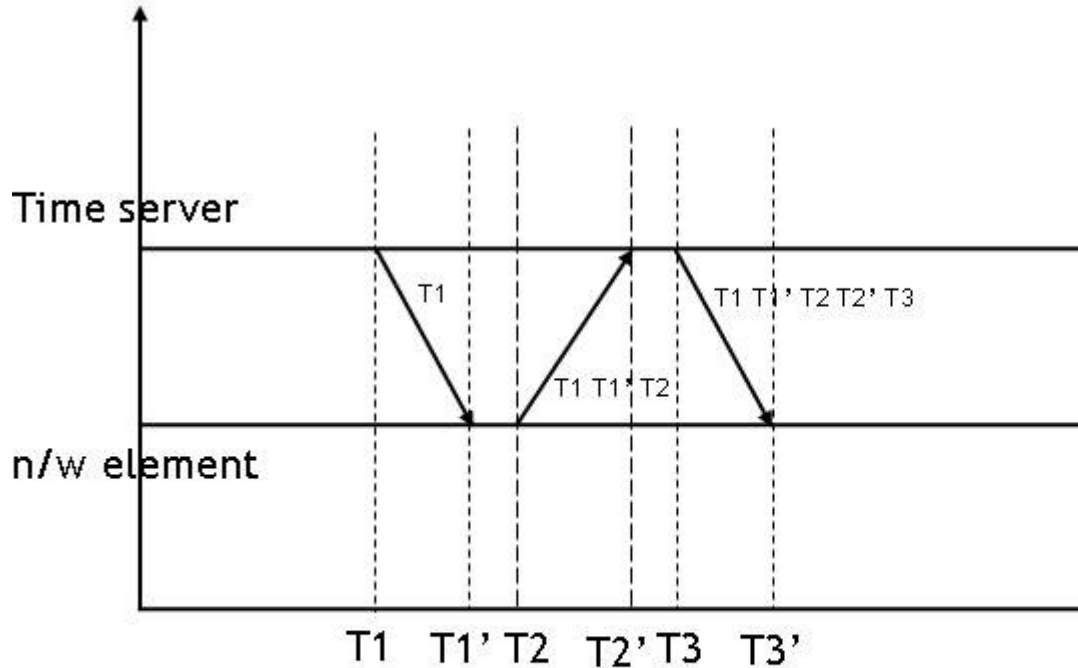- Range : -108 to 21 microseconds
- Mean : 0.335034
- StdDev: 6.050032

Environment

- Simple 10Gb LAN for NTP traffic
- Poll interval 8 sec
- Heavy CPU Activity,
  Minimal Network Activity

Precision Time Protocol (PTP)

Precision Time Protocol (PTP)

Time server

T1

T1 T1' T2 T2' T3

T1 T1' T2

n/w element

T1   T1' T2   T2' T3   T3'

Assumptions

- Master to Slave Transit time
        = equals =
  Slave to Master Transit Time
- Delays (Transit times) Remain Constant
- Master and Slave can accurately compute
  time to send or receive messages.

Precision Time Protocol (PTP) – SW Only Implementation

PTP CLIENT

Jitter Source

SYSTEM CALL API

PTPD

USERSPACE

KERNEL

Network Stack

SYSTEM CLOCK

HARDWARE

ADAPTER

NETWORK

**Time From PTP Master on network**

PTP – SW Only

- Intended for host synchronization in LANs
- Hierarchical Time Distribution
- Slaves sync to Master
- Master syncs to GPS or similar time source
- Query for time sync default is 1 sec
- Uses timestamps on packets to compute latency information

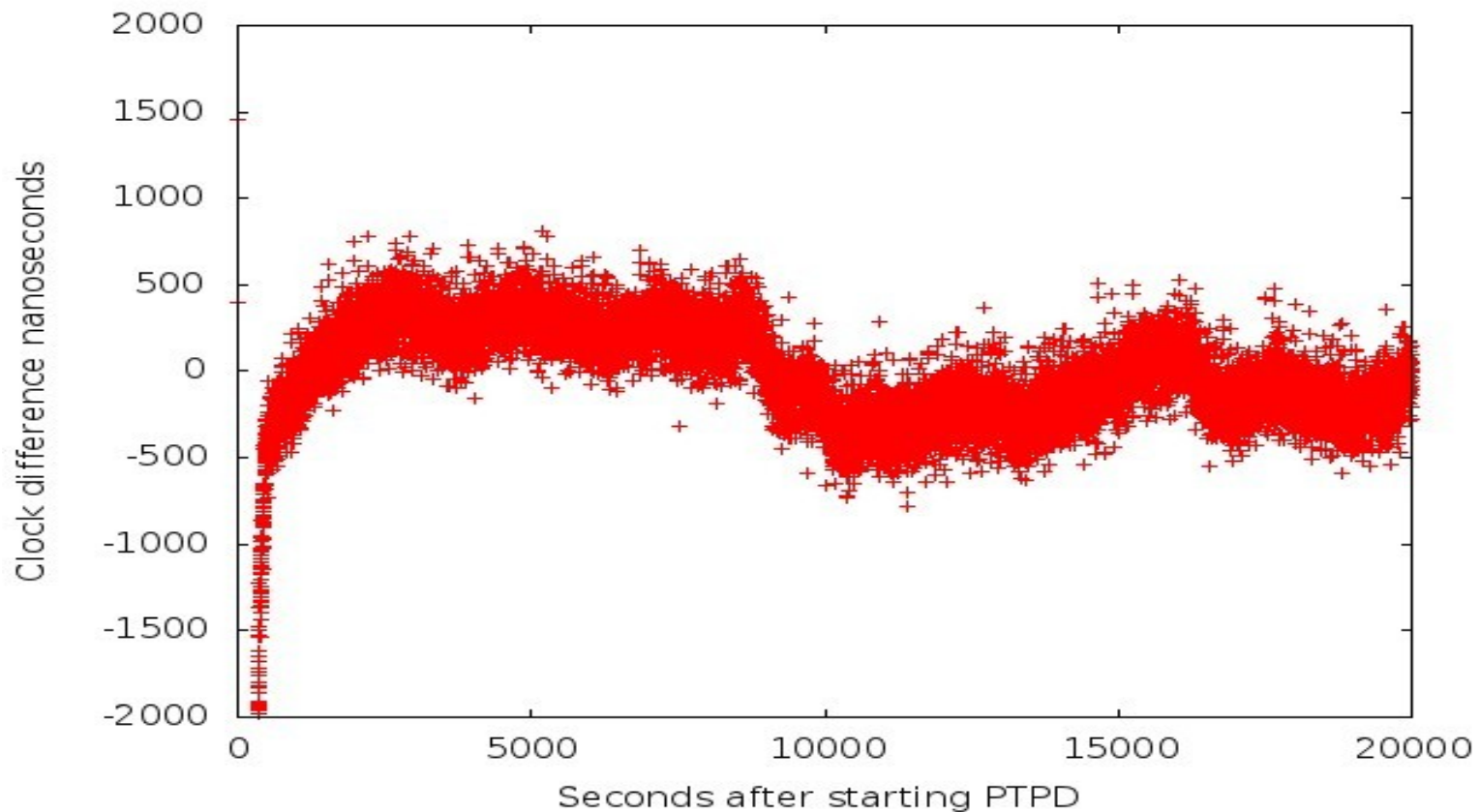**Sources of jitter similar to NTP**
- Packets timestamped in Network stack
- Socket option `SO_TIMESTAMP`

Requires
- Linux kernel 2.6.30 or later
- OpenSource PTP daemon
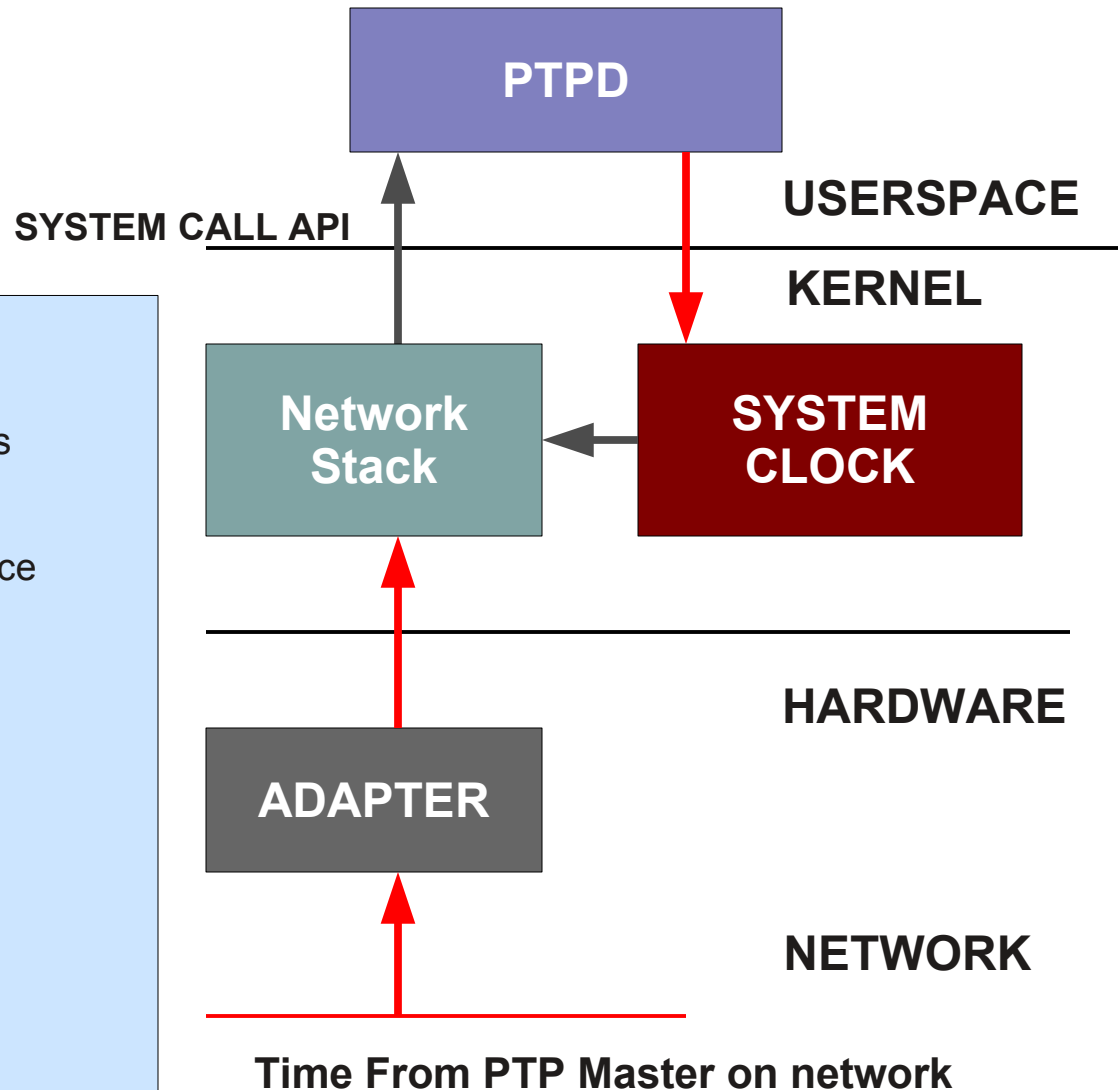
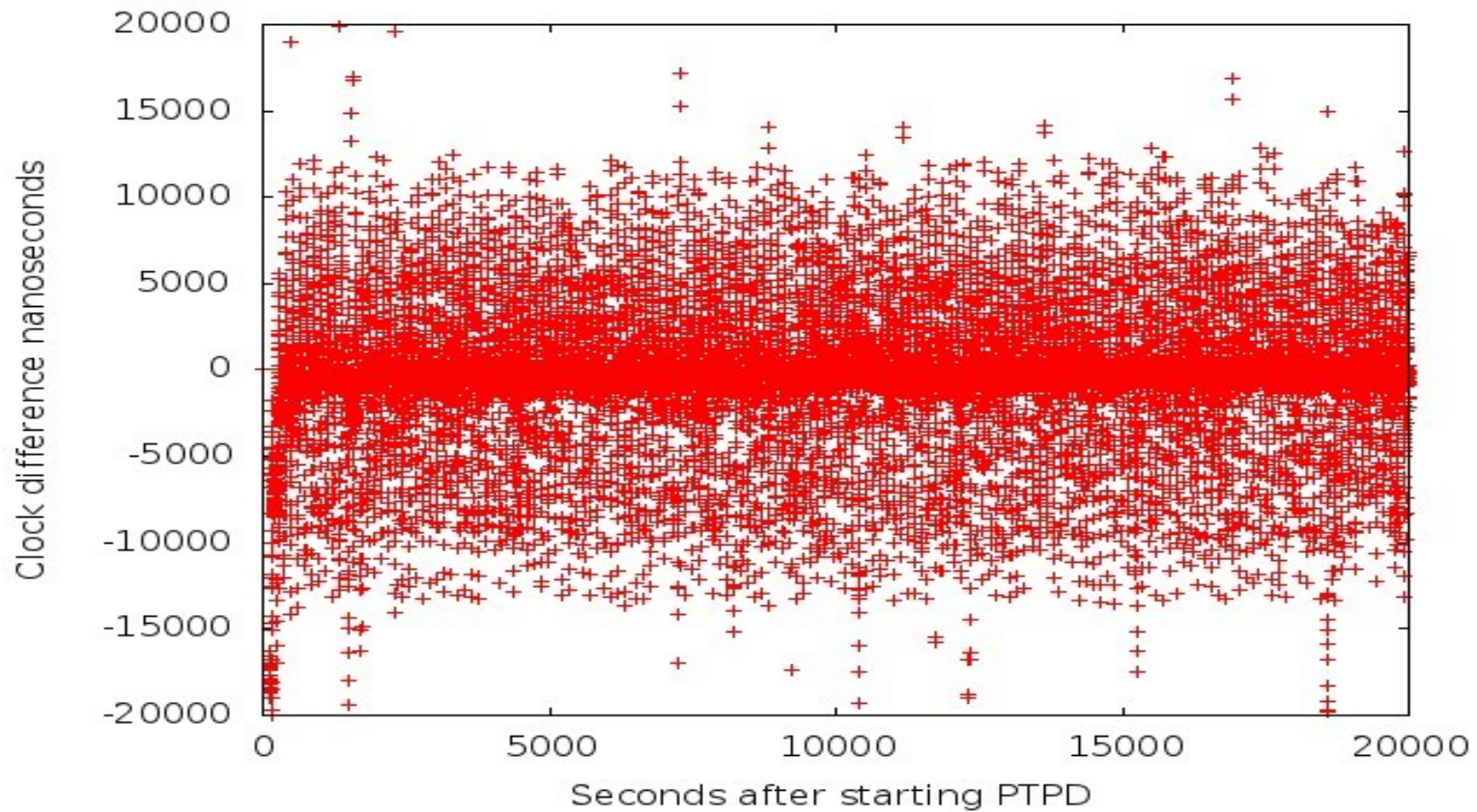# Precision Time Protocol SW Only – LAN (Minimal Load)

Stats

- Beginning 15 minutes after start
- Range : -783 to 807 nanoseconds
- Mean   : 18.975601
- StdDev: 248.732998

Environment

- Simple 10Gb LAN for PTP traffic
- Sync interval 1 sec
- Minimal CPU and Network Activity

Precision Time Protocol SW Only – LAN (CPU Load)

Stats

- Beginning 15 minutes after start
- Range : -45917 to 199103 ns
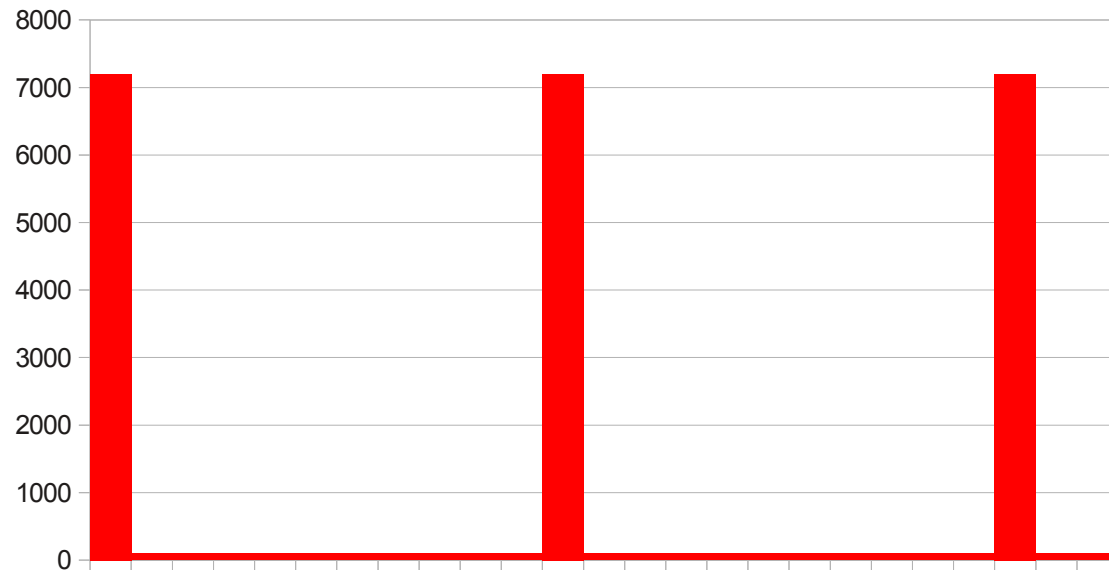- Mean : -48.299971
- StdDev: 5793.242824

Environment

- Simple 10Gb LAN for PTP traffic
- Sync Interval 1 sec
- Heavy CPU Activity,
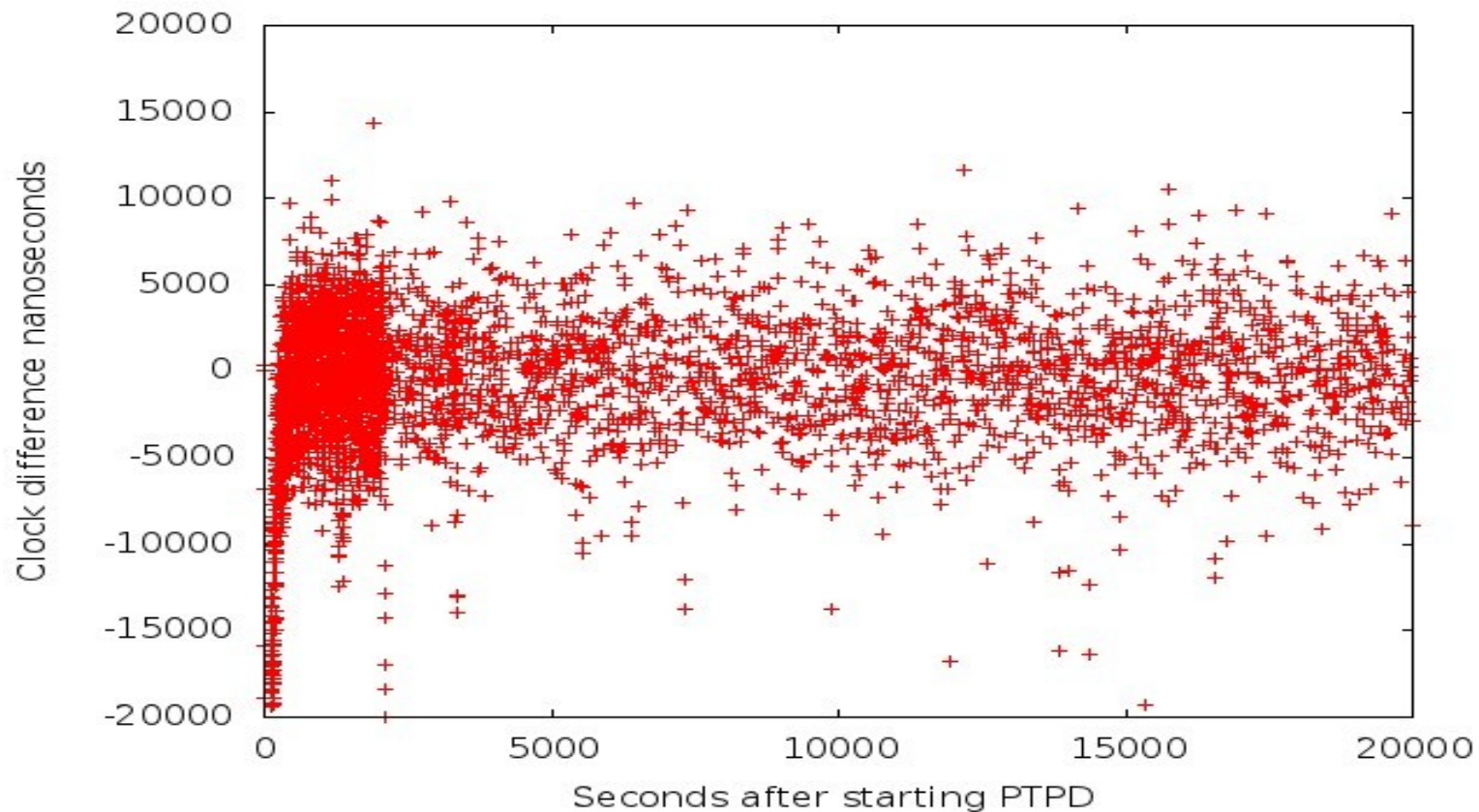 Minimal Network Activity

Adding Network Load

SCP and NETPERF

```
while true; do
        netperf -l 1 -H 10.0.0.11            #  7.2Gb  (1 Second)
        scp -l 102400 to.11 10.0.0.11:work # 100Mb (10 Seconds)
done
```

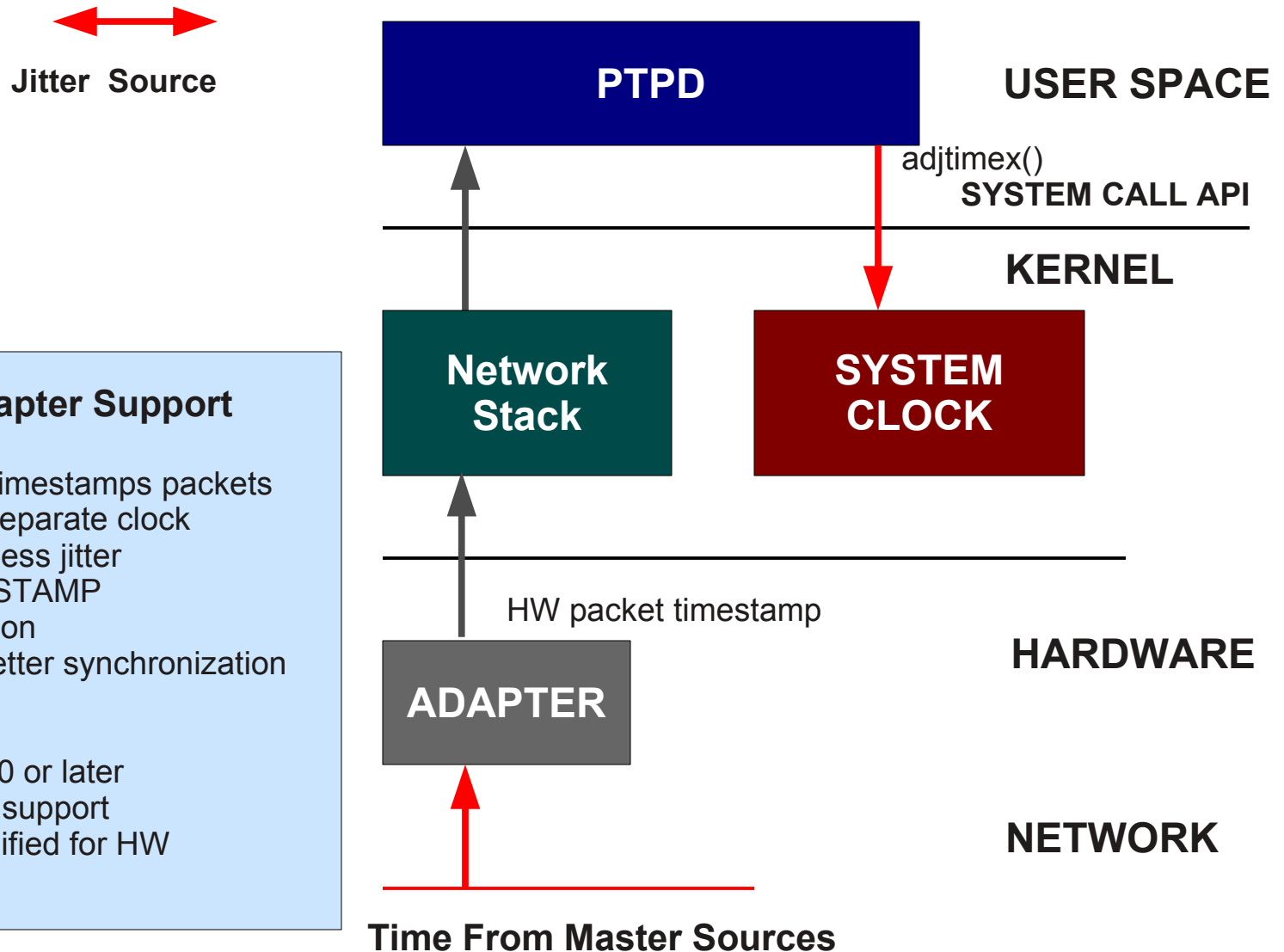# Precision Time Protocol SW Only – LAN (CPU & Net Load)

Stats

- Range : -115106 to 578730 ns
- Mean   : -0.637253
- StdDev: 13670.112245
- **Delay Updates Aborted : 13507**

Environment

- Simple 10Gb LAN for PTP traffic
- Sync interval 1 sec
- Heavy CPU and Network Activity

Precision Time Protocol (PTP) - Adapter TimeStamp Support

Jitter Source

PTPD

USER SPACE

adjtimex()

SYSTEM CALL API
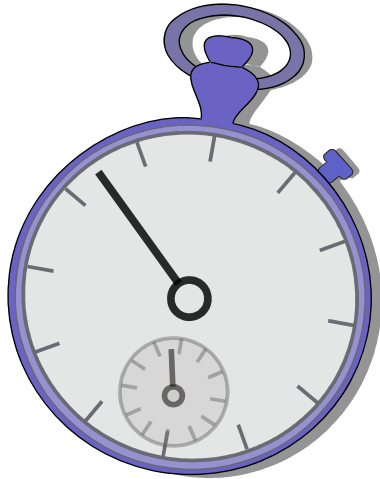
KERNEL

Network Stack

SYSTEM CLOCK

PTP - Adapter Support

- Network adapter timestamps packets
  - Adapter has separate clock
- Closer to events, less jitter
- Socket SO_TIMESTAMP
  - Hardware option
- microsecond or better synchronization

Requires
- Linux kernel 2.6.30 or later
- Adapter with PTP support
- PTP daemon modified for HW timestamping

HW packet timestamp

HARDWARE

ADAPTER

NETWORK

Time From Master Sources

New Problem – Multiple Clocks on Same System

System Clock

Provided view of Time to:
- Kernel
- Applications

Clock on Adapter

Provides TimeStamp for:
- Incoming Packets
- Outgoing Packets

Multiple PTP Domains

Jitter Source

PTPD

USER SPACE

new API

SYSTEM CALL API

adjtimex()

KERNEL

Network Stack

SYSTEM CLOCK

ADAPTER (Clock)

HARDWARE

PTP Adapter Clock Domain

PTP System Clock Domain
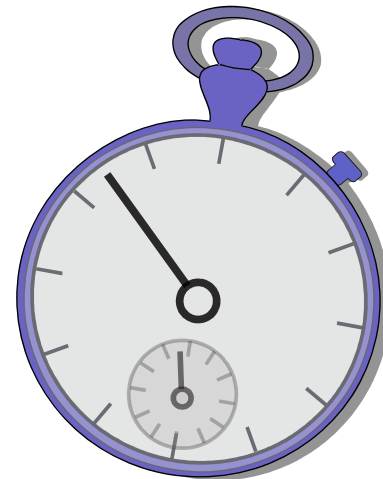
Time From Master Sources
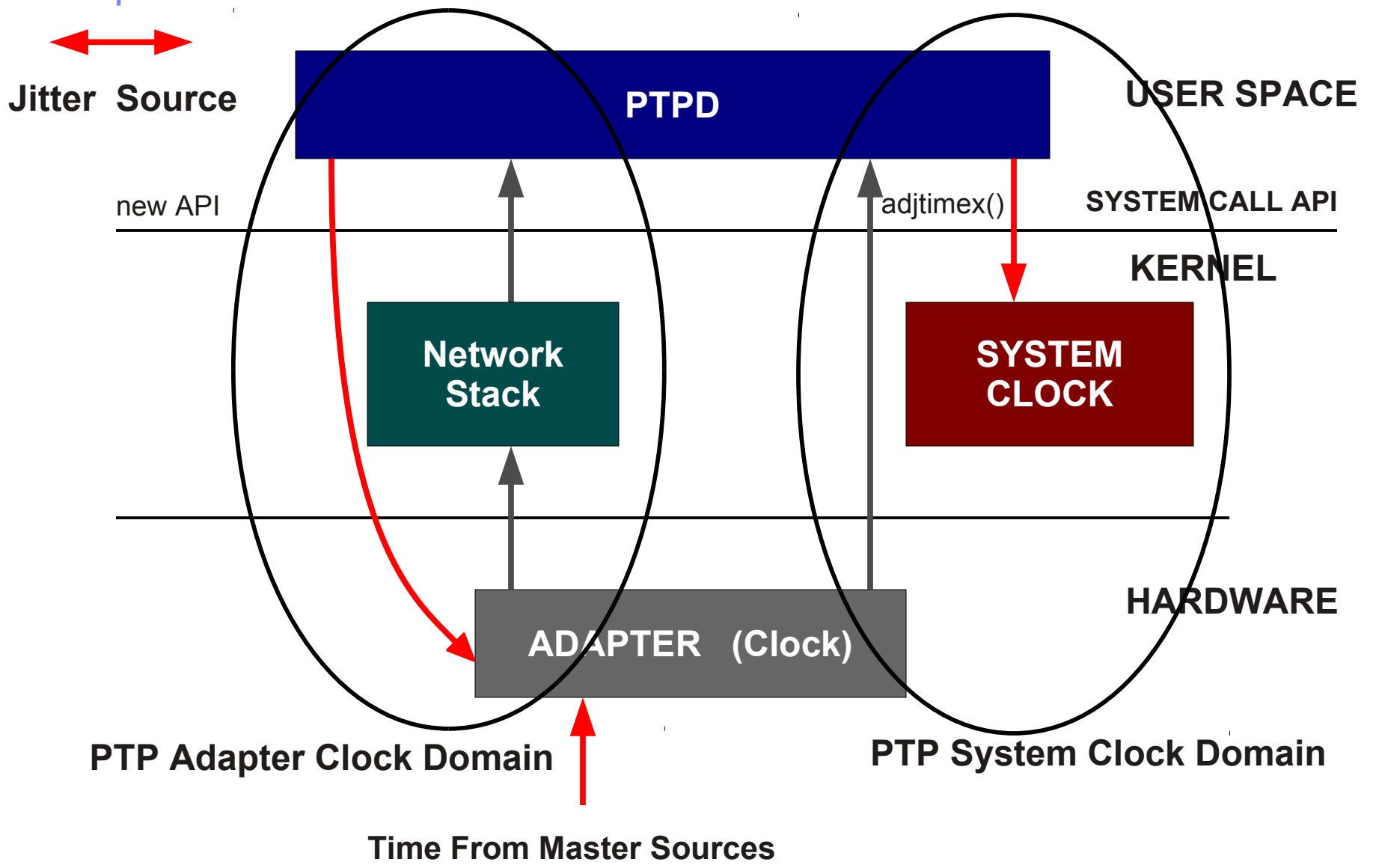
Linux PTP Clock Support - Future Direction

USER SPACE

ANY APPLICATION

PTPD

PTP Clock API

KERNEL

Network Stack

SYSTEM CLOCK

PTP – Linux PTP Clock support

- Framework for Accessing PTP Clocks
- User API for PTP clock access
- PTP daemon access PTP Clocks in standard way
- No need for special daemons
- Applications can use PTP clock

Requires
- Current Linux kernel 3.0.+
- Driver for PTP clock
- PTP daemon modified for PTP Clock

HW packet timestamp

HARDWARE

ADAPTER

NETWORK

Time From Master Sources
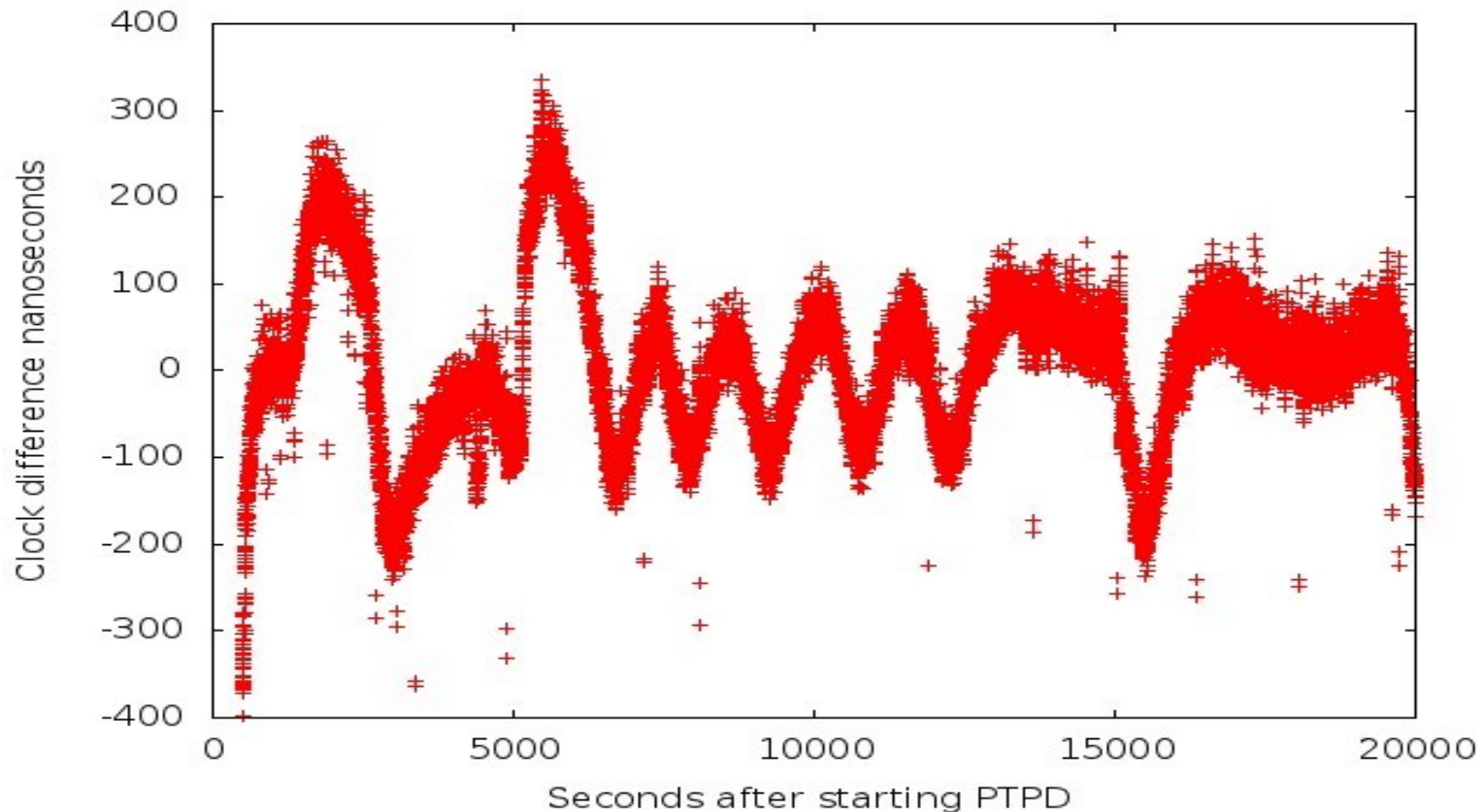
PTP Adapter Clock – LAN (Minimal Load)
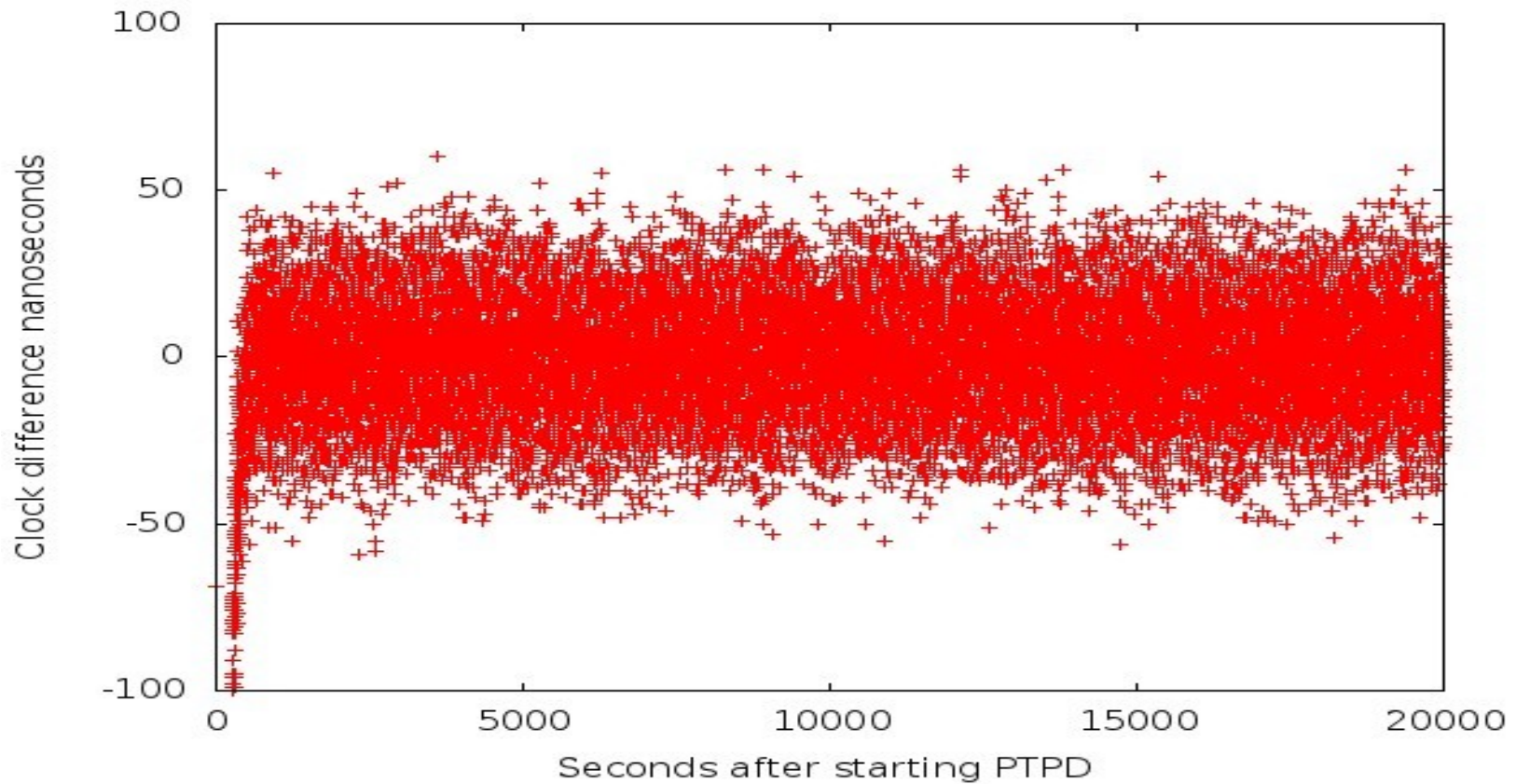
Stats

- Beginning 15 minutes after start
- Range : -59 to 60 nanoseconds
- Mean   : 0.061463
- StdDev: 16.908153

Environment

- Adapter Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Minimal CPU and Network Activity

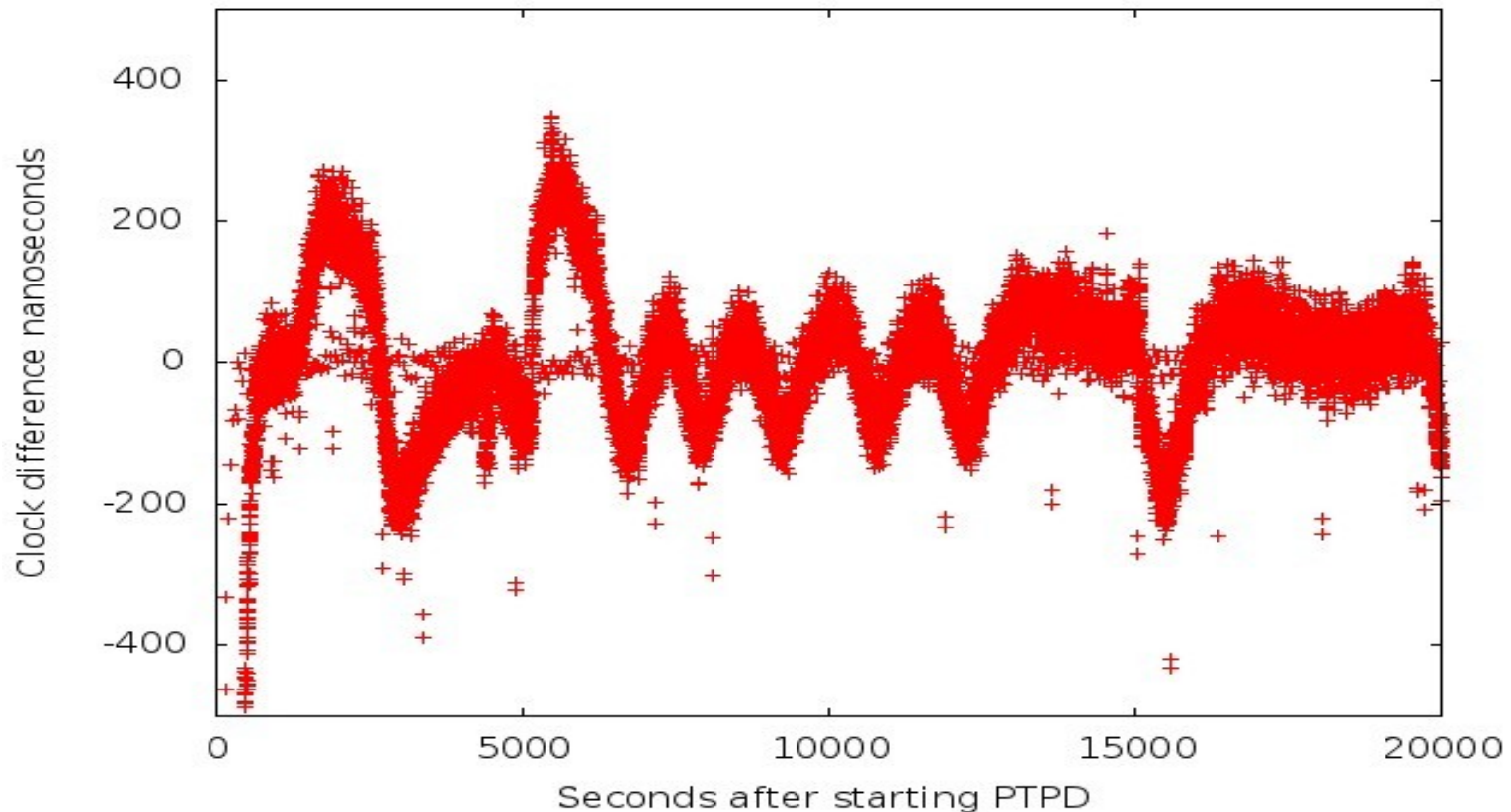# PTP System Clock (from Adapter) – LAN (Minimal Load)

Stats

- Beginning 15 minutes after start
- Range : -438 to 336 nanoseconds
- Mean   : 9.825280
- StdDev: 89.881447

Environment

- System Clock deltas from Adapter
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Minimal CPU and Network Activity

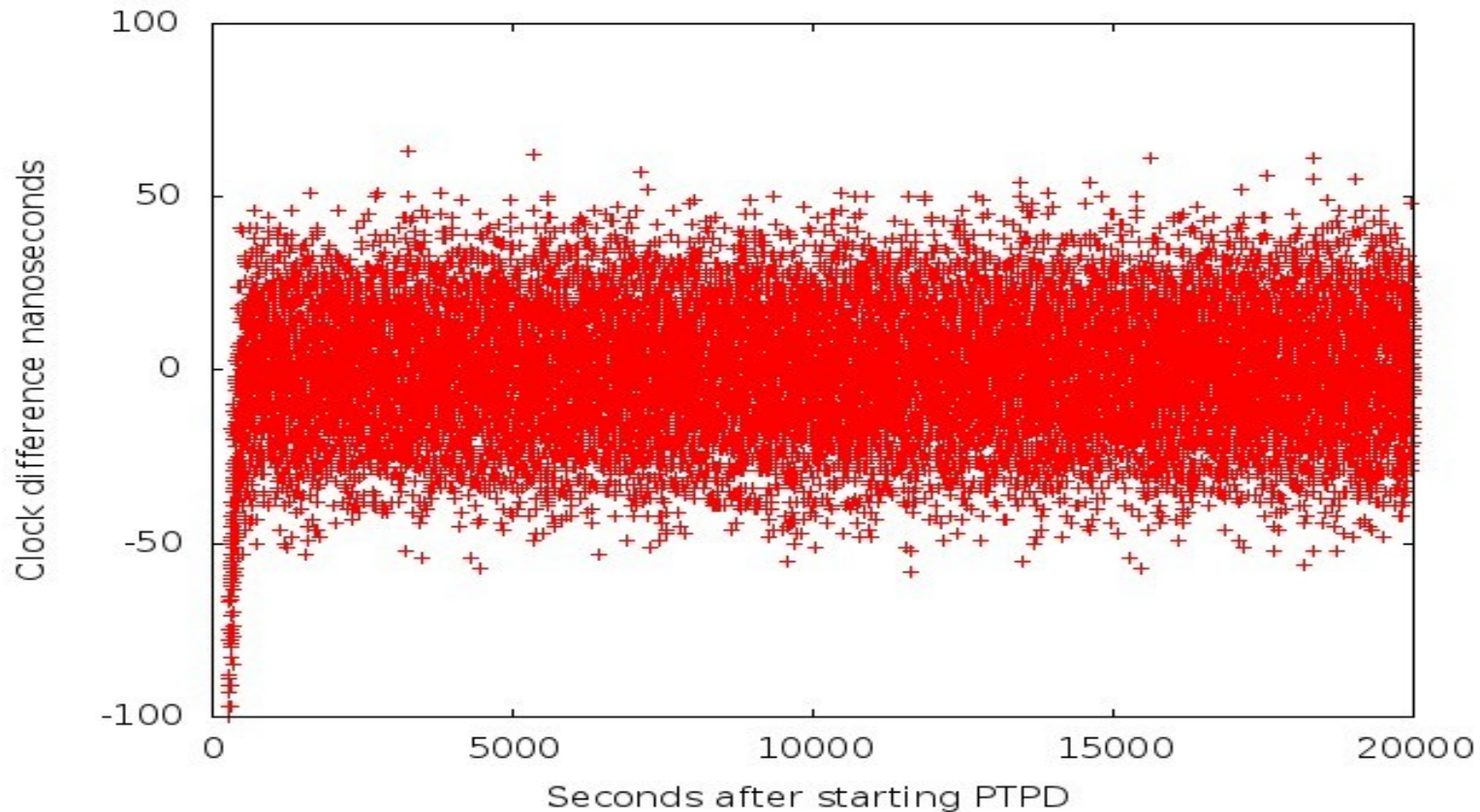# PTP System Clock (from Master) – LAN (Minimal Load)

Stats

- Beginning 15 minutes after start
- Range : -431 to 350 nanoseconds
- Mean : 9.600230
- StdDev: 89.695492

Environment

- System Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Minimal CPU and Network Activity
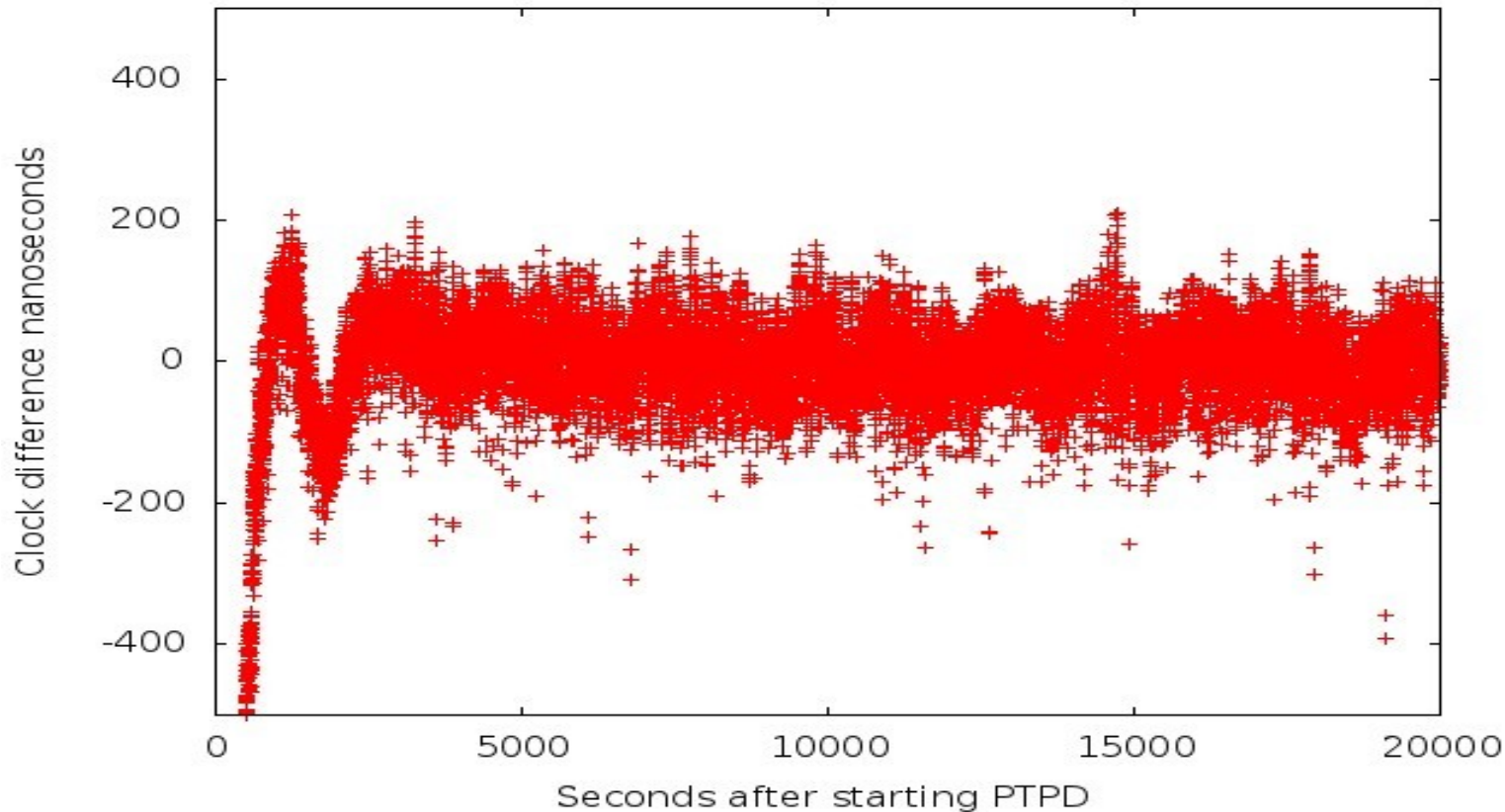
PTP Adapter Clock – LAN (CPU Load)

Stats

- Beginning 15 minutes after start
- Range : -58 to 63 nanoseconds
- Mean   : -0.052723
- StdDev: 17.305986

Environment

- Adapter Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Min Net Activity

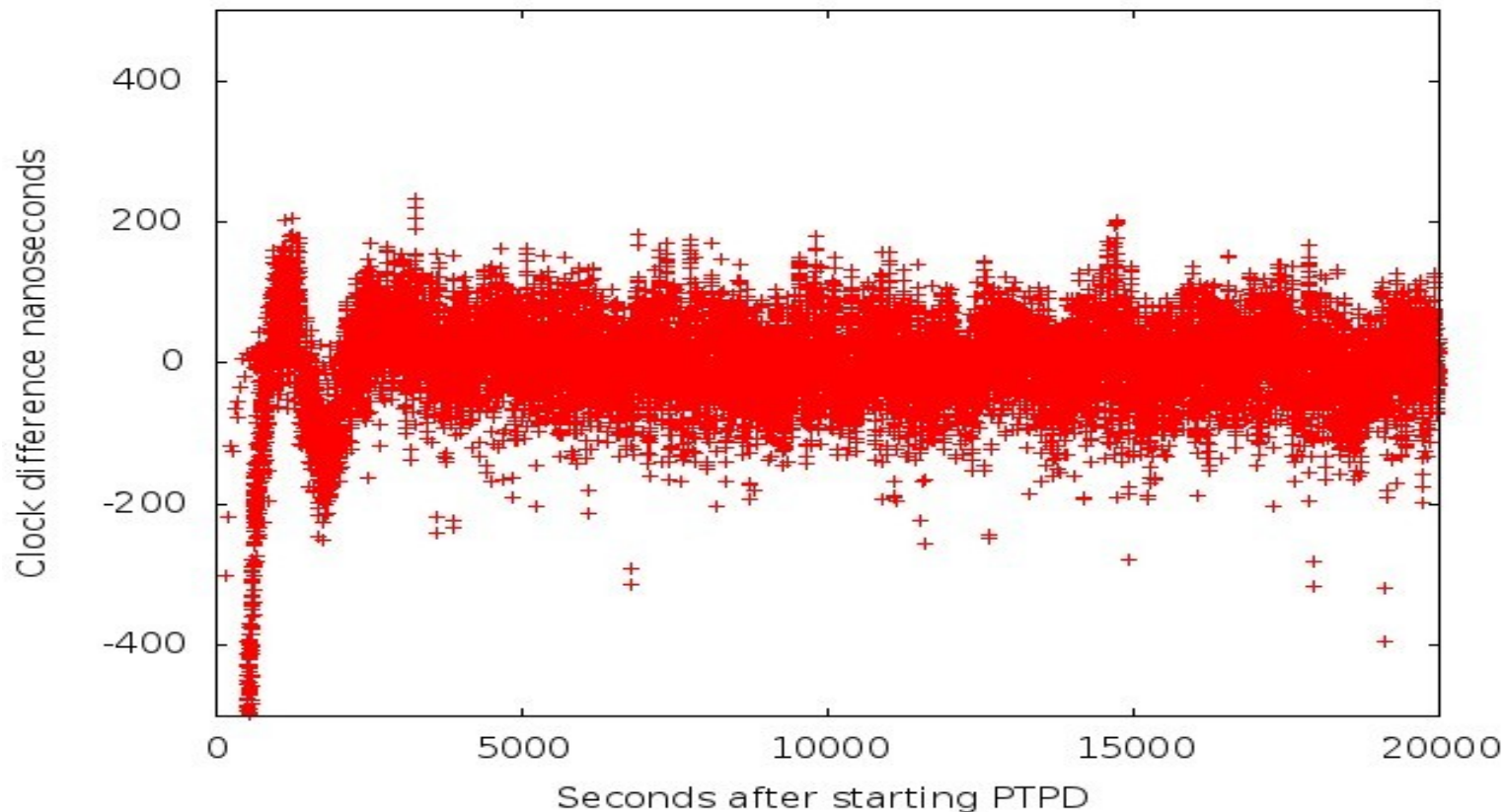# PTP System Clock (from Adapter) – LAN (CPU Load)

Stats

- Beginning 15 minutes after start
- Range : -391 to 211 nanoseconds
- Mean   : 3.450447
- StdDev: 55.813730

Environment

- System Clock deltas from Adapter
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Min Net Activity
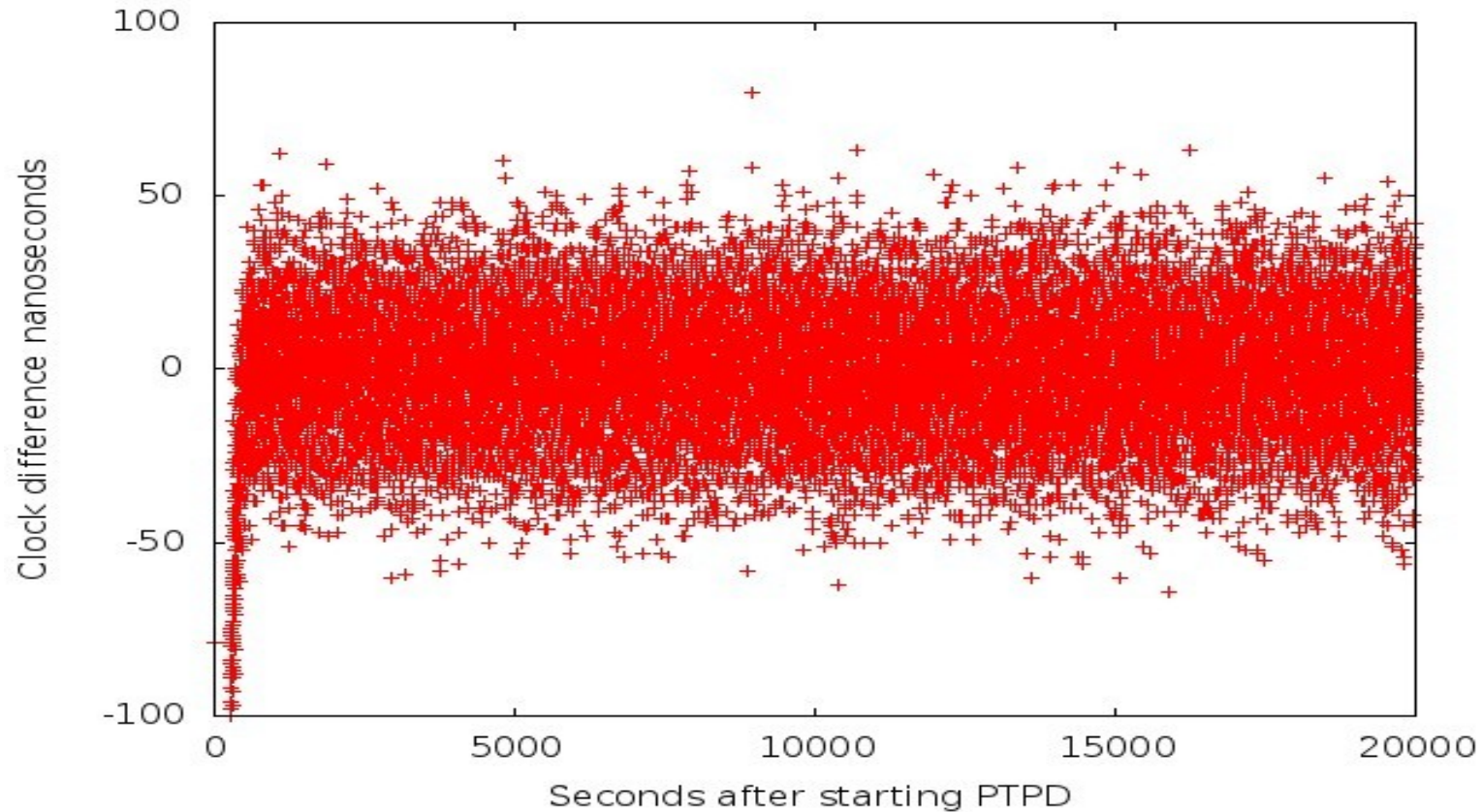
PTP System Clock (from Master) – LAN (CPU Load)

Stats

- Beginning 15 minutes after start
- Range : -395 to 232 nanoseconds
- Mean   : 3.259316
- StdDev: 57.199593

Environment

- System Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Min Net Activity

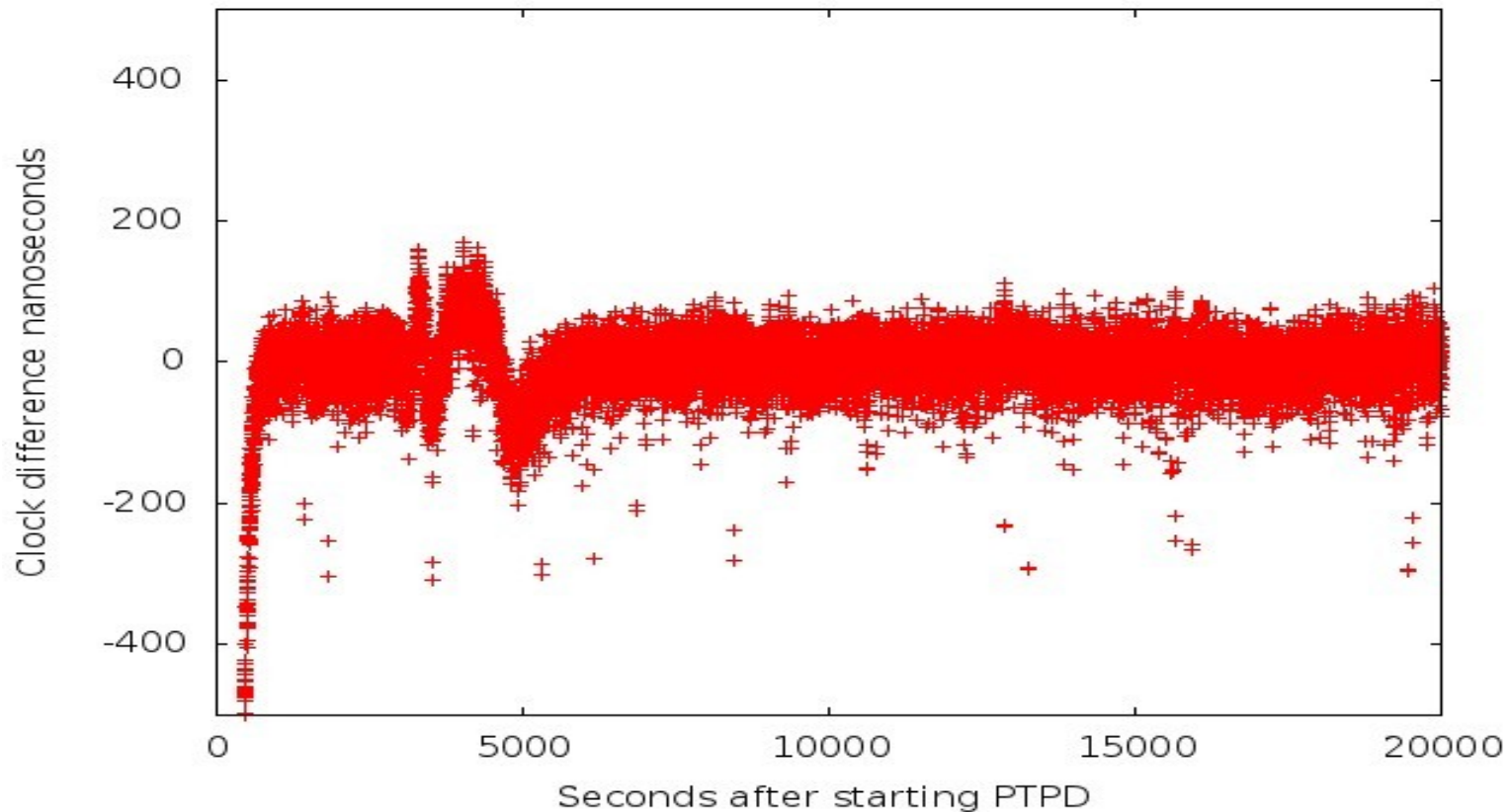# PTP Adapter Clock – LAN (CPU & Net Load)

Stats

- Beginning 15 minutes after start
- Range : -64 to 80 nanoseconds
- Mean   : -0.031430
- StdDev: 17.836507

Environment

- Adapter Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Heavy Net Activity

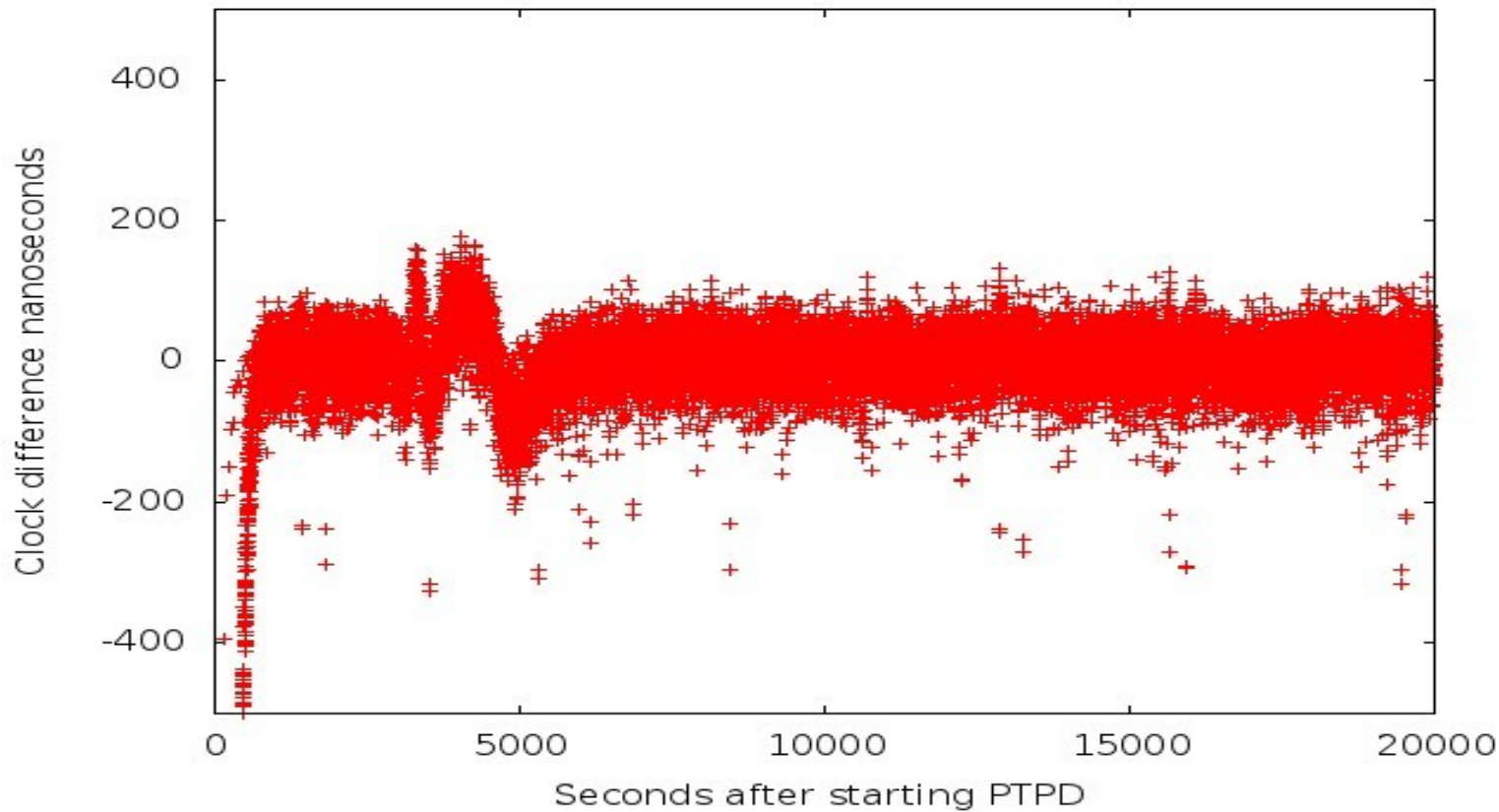# PTP System Clock (from Adapter) – LAN (CPU & NET Load)

Stats

- Beginning 15 minutes after start
- Range : -308 to 169 nanoseconds
- Mean   : 0.631878
- StdDev: 39.611248

Environment

- System Clock deltas from Adapter
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Heavy Net Activity

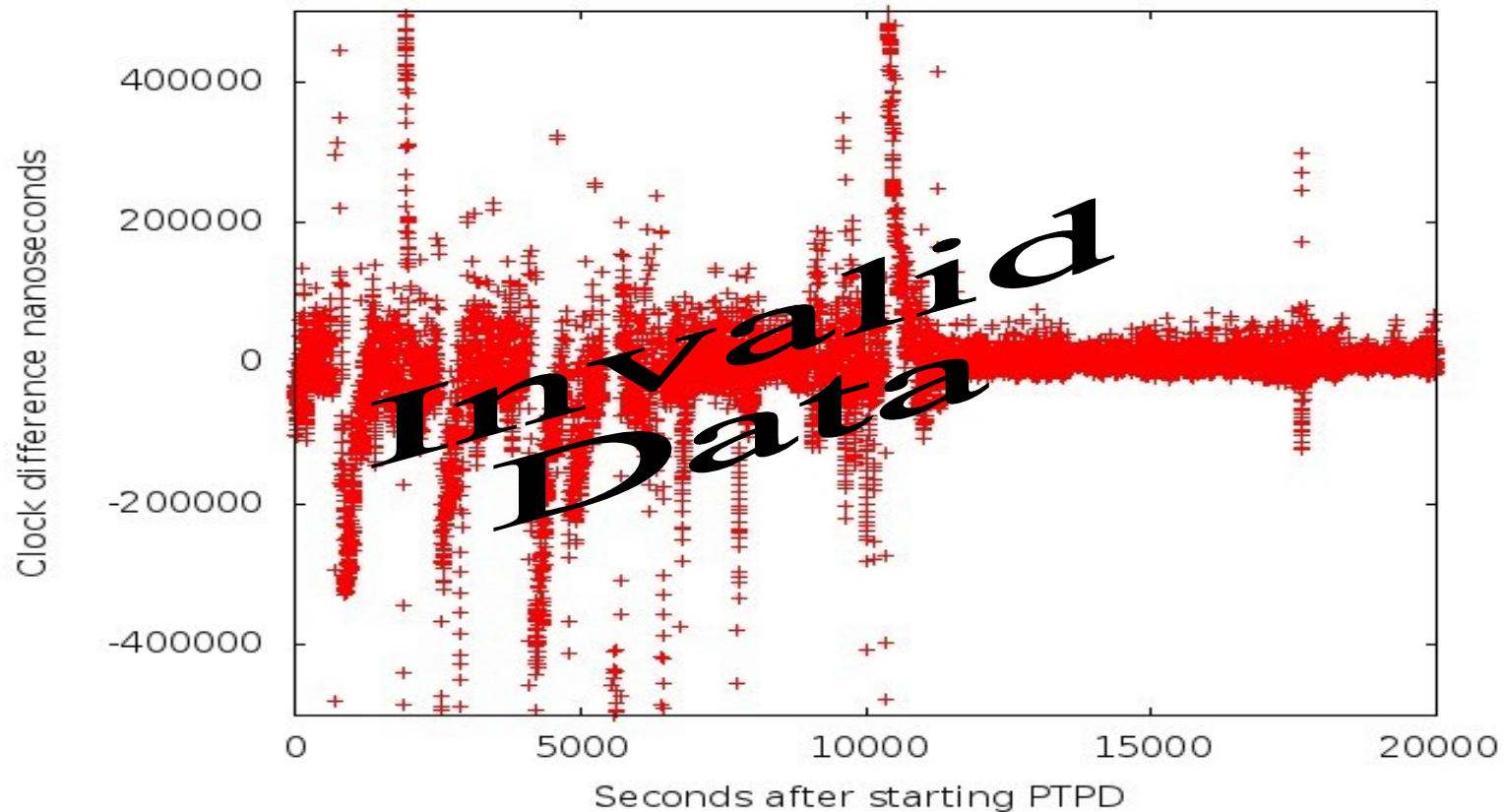# PTP System Clock (from Master) – LAN (CPU & NET Load)

Stats

- Beginning 15 minutes after start
- Range : -326 to 178 nanoseconds
- Merge : 0.586328
- StdDev: 42.471485

Environment

- System Clock deltas from Master
- Simple 10Gb LAN for NTP traffic
- Sync interval 1 sec
- Heavy CPU Activity, Heavy Net Activity
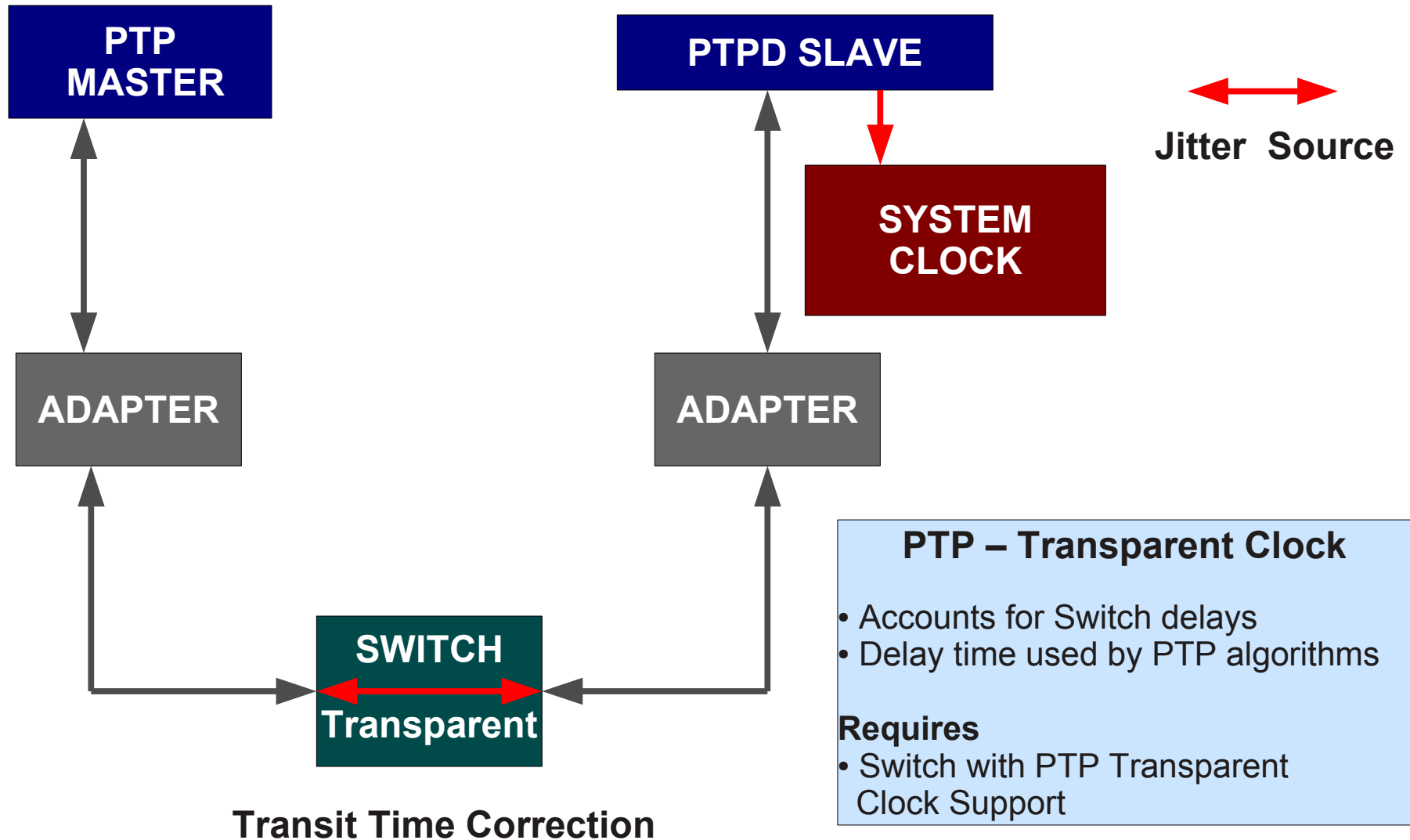
Same Test – Bad Network Cable

Stats

• Range -7,320,339 to 82,032,290 ns
• Packets delayed up to 5,444,813 ns

Environment

• Bad Network Cable
• Do not try at home :)

PTP – Transparent Clock/Switch Support
Addressing Jitter in the LAN

PTP MASTER

PTPD SLAVE

↔ **Jitter Source**

SYSTEM CLOCK

ADAPTER

ADAPTER

SWITCH Transparent

Transit Time Correction

PTP – Transparent Clock

- Accounts for Switch delays
- Delay time used by PTP algorithms

Requires
- Switch with PTP Transparent Clock Support

PTP – Boundary Clock/Switch Support.   Bridging Multiple LANs

PTP MASTER

PTP SLAVE **PTP SLAVE** **PTP SLAVE**

SWITCH (Transparent)   Slave   **SWITCH (Boundary)**   Master   **SWITCH (Transparent)**

PTP SLAVE **PTP SLAVE** **PTP SLAVE**
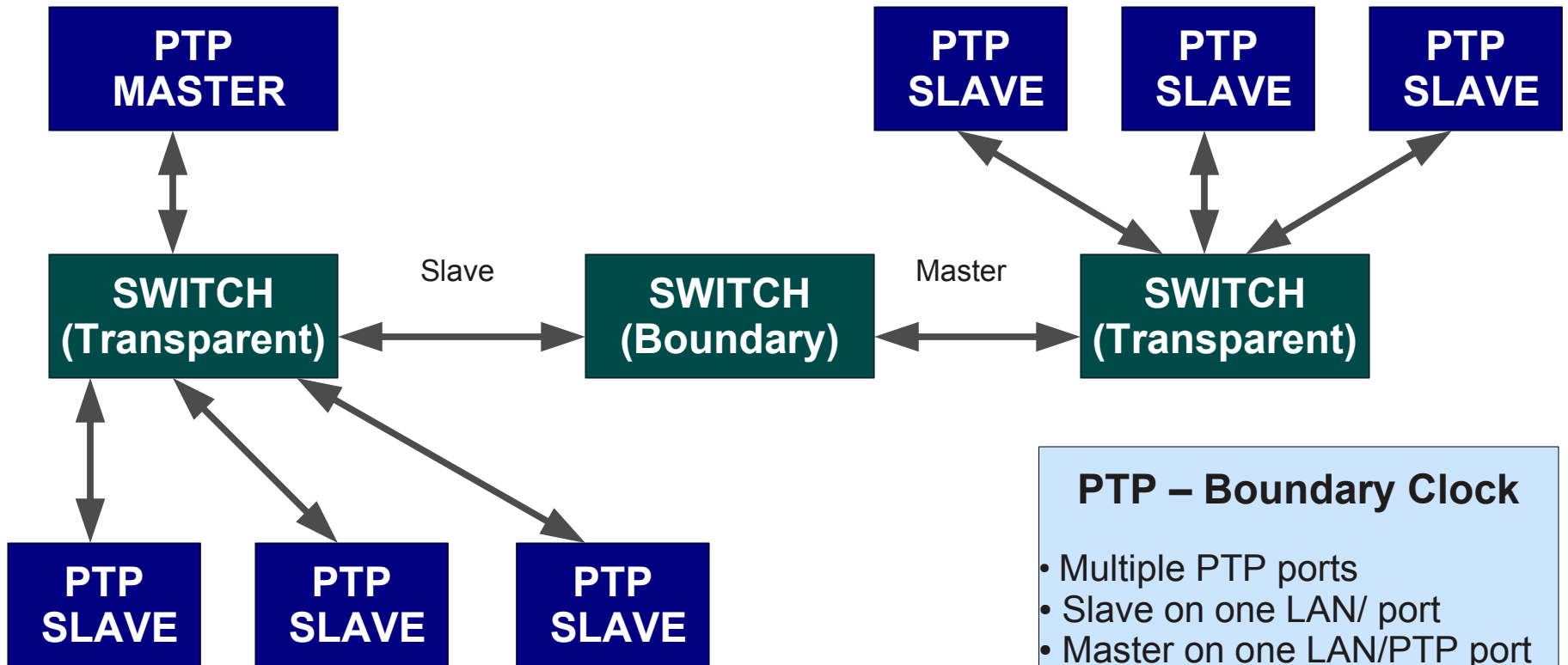
PTP – Boundary Clock

• Multiple PTP ports
• Slave on one LAN/ port
• Master on one LAN/PTP port
• Ideal topology minimizes
  Boundary Clock Traversals

Requires
• Switch with PTP Boundary
  Clock Support

Acknowledgments

- Solarflare Corporation
    - PSFN5322F 10GbE Adapters with PTP Support

- Symmetricom Corporation
    - SyncServer S350 Time Server with PTP support

Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.

- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.

- UNIX is a registered trademark of The Open Group in the United States and other countries

- Linux is a registered trademark of Linus Torvalds.

- Other company, product, and service names may be trademarks or service marks of others.