The Need for Accurate Time

- Coordination Amongst Distributed Systems
- Cellular Networks
- Factory Automation
- Electrical Power Grids
- High Frequency Trading
Introduction

Why Aren’t Computer Clocks Accurate?

- Commodity crystals can be ±30 seconds per year
- Can wander by tens of milliseconds over brief periods
- Environmental Effects
- Crystal Aging
Hardware Solutions

- Better Crystals
- Add in Cards
- All of these are expensive, $1000 per host
Network Solutions

- Ask a “better” clock what time it is
- daytime protocol, circa 1983
- NTP circa 1980 (The Gold Standard)
- Precision Time Protocol 2002/2008
Precision Time Protocol

- First defined as IEEE-1588/2002 (Version 1)
- Current version is IEEE-1588/2008 (Version 2)
- Heavy use of Multicast
- Works best on a LAN
Theory of Operation

- A Really Good Clock Exists
- Grandmaster Multicasts its time on the LAN
- Slaves work out network latency to Grandmaster
- Slaves adjust their local clocks based on their measurements
Terms

Grandmaster  The Really Good Clock

Steering  The process of manipulating the local clock

Slave  A client that wishes to steer its clock

Offset  Time difference between slave and master

Delay  Time a packet takes to go between the master and slave
Hardware Time-stamping

- Time-stamps should be recorded as close to the PHY as possible
- Recording a time-stamp in the kernel is better than in user space
- The kernel is really not quite good enough
Possible Time-stamping Locations

- PTPd
- Socket
- UDP
- IP
- Ethernet
- NIC
- gettimeofday()
- SO_BINTIME
- PCAP/BPF Time
- Hardware Timestamp
PTP Packets

- ANNOUNCE
- SYNC
- FOLLOWUP
- DELAY REQUEST
- DELAY RESPONSE
SYNC

- Main packet from master to slave
- Carries the time from the master (T1)
- Sent every 2 seconds (default)
- Slave timestamps these packets (T2)
FOLLOWUP

- Used when master does not have hardware time-stamping
- Follows the SYNC packet
- Carries the time at which the SYNC was transmitted
DELAY REQUEST

- Sent from the slave to the master
- First of a pair of packets used to derive network delay
- Slave records when it sent this packet (T3)
- Can now be sent via unicast or multicast
DELAY RESPONSE

- Master responds with this packet to a `DELAY REQUEST`.
- Carries the time at which the `DELAY REQUEST` arrived (T4).
- Used by the slave to determine network delay.
Network Time Diagram

Master

T1

SYNC [T1]

FOLLOWUP [T1]

T2

Slave

T3

DELAY REQUEST [T3]

T4

DELAY RESPONSE [T4]
Implementation

Features

- Can run as a master, serving hundreds of slaves
- Runs as a slave on the client
- Lightweight, does not consume many resources
- Uses BPF/PCAP timestamps for enhanced accuracy on FreeBSD
- Records performance data
- Records quality files (more later)
Supporting Scripts

- **ptplib** R library for logfiles
- **offset.R** Graphs offset and delay data from log files
- **compare.R** Compare two clients log files
- **stats.R** Raw statistics
Measuring Slave Synchronization

- All slaves record several measurements
  - Master to Slave Delay
  - Slave to Master Delay
  - Offset from Master
  - Quality
Inter-Slave Quality Measurement

- What often most concerns us is synchronization between slaves
- We need an external event that all slaves see equally
- It is necessary to line up measurements between multiple slaves
- Use the `SYNC` packet’s sequence id
- `SYNC` is multicast so all slaves see the same packet at the same time
  - More or less
- Record local time on each `SYNC`
- Post processing shows us how well synchronized two slaves are
Offset Measurements for Unsynchronized System
Offset Measurements for System Synchronized to NTP

![Graph showing offset measurements over time]

(ntpsync 2011-02-10 13:24:51 - 2011-02-10 23:08:06)
Offset Measurements for PTP Synchronized System
Quality Measurements for PTP Synchronized System
Sources of Timing Inaccuracy

- Network Jitter
- Store and Forward Routers
- Packet Asynchrony
- Competing Network Traffic
- Interrupt Moderation in NIC Drivers
Current Status

- Maintained at SourceForge
- BSD License
- Runs on FreeBSD, Linux and Mac OS
- FreeBSD Ports
- RedHat RPM
Future Work

- Support for low level timestamping
- netmap(4) support
- DPDK support
- Better support for embedded systems
- Move to github
- Always more testing to do