An IEEE-1588 Compatible RADclock

Matthew Davis[1], Benjamin Villain[2], **Julien Ridoux**[1], Anne-Cécile Orgerie[3], Darryl Veitch[1]

[1] Electrical & Electronic Engineering Dpt, The University of Melbourne, Australia
{matt, julien}@synclab.org, dveitch@unimelb.edu.au

[2] Université de Nantes, IRCCyN UMR CNRS 6597, Nantes, France
benjamin.villain@gmail.com

[3] Ecole Normale Supérieure, Lyon, France
annececile.orgerie@ens-lyon.fr

This work was supported in part by a research grant from Symmetricom Inc.

Introduction

Present some new features and recent work on RADclock

- Robust Absolute and Difference clock
- Software clock that relies on a feed-forward paradigm

RADclock aims at being an "ideal" software clock, capable to:

- Use any synchronization protocol (IEEE 1588, NTP)
- Use software timestamps but be reliable and accurate
- Use NIC hardware timestamps when available
- Be robust to latency variability of network and OS
- Be robust to high system load (consistent performance)

Motivation

Previous work

- ISPCS 2008, RADclock (NTP) vs. ptpd (IEEE 1588) perfs
- Demonstrated impact of network delay on clock performance
- Advocated for a robust synchronization algorithm to filter noise

Purpose of this talk

- RADclock and ptpd have improved, work should be revisited
- Increasing hardware timestamping support on commodity NICs
- Compare to commercial solution (TimeKeeper by FSMLabs)

Comparing Contenders

	ntpd	ptpd	TimeKeeper	RADclock
Open Source	✓	✓	✗	✓
Linux	✓	✓	✓	✓
*BSD	✓	✓	✗	✓
Windows	✓	✗	✗	✗
NTP	✓	✗	✓	✓
IEEE 1588	✗	✓	✓	✓
S/W timestamps	✓	✓	✓	✓
H/W timestamps	✗	✗ *	✓	✓ *

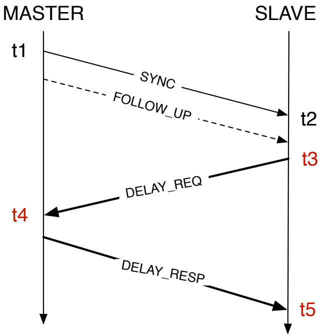
- Not possible to compare all solutions across all dimensions
- Present most interesting comparisons only (using IEEE 1588)

RADclock IEEE 1588 Support

Early Support / Proof of Concept

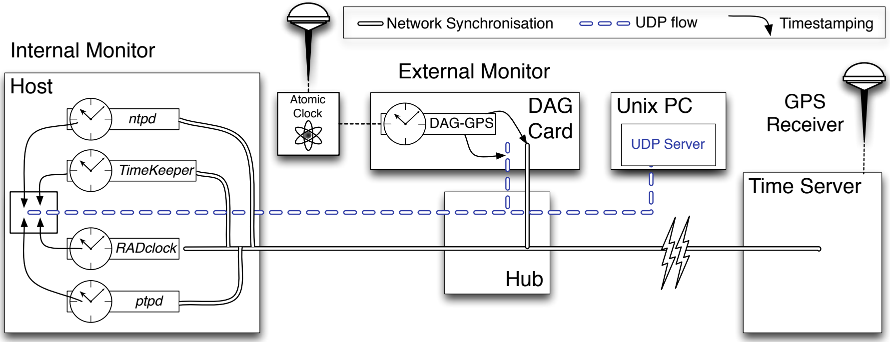
- Runs as slave only (reasonable for software clock)
- End-To-End only
- No support for Announce, Signalling or Management Message

Use 1588 Packets as Needed by Feed-Forward Algorithm



- DELAY_REQ and DELAY_RESP only
- Use bi-directional paradigm, for reliable RTT based filtering
- Ignore SYNC and FOLLOW_UP
- Much less input data than ptpd or TimeKeeper

Comparison Methodology



- DAG card provides the hardware time reference
- Compare two clocks at a time, each against DAG timestamps
- Clocks timestamp UDP test packets (almost) simultaneously
    - BPF/libpcap timestamps if we are conducting a software timestamping experiment
    - Hardware timestamps if available

Clock Input

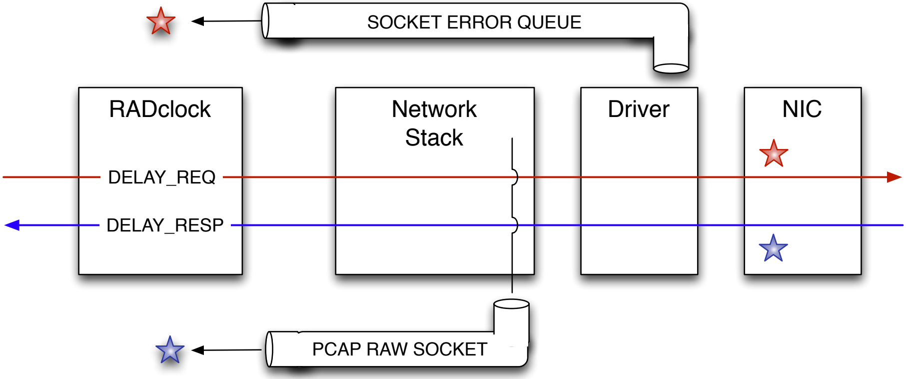
Clocks Share:

- Host load and latency
- Network conditions
- Time server quality

. . . but attain timestamps from different locations

- **ntpd SW**: Userland
- **ptpd SW**: SO_TIMESTAMP (socket layer)
- **TimeKeeper HW**: NIC timestamps converted by kernel
- **TimeKeeper SW**: ???
- **RADclock SW**: BPF/libpcap timestamps
- **RADclock HW**: NIC oscillator RAW counter

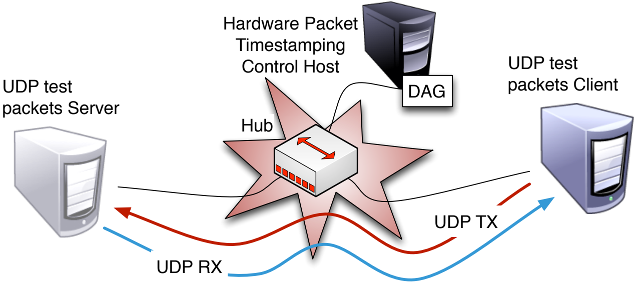
RADclock H/W Timestamping



We modified the Intel i350 NIC driver (Linux)

- Export raw i350 time counter from the NIC up to userland
- This hardware value can be read from userland via the Linux socket API

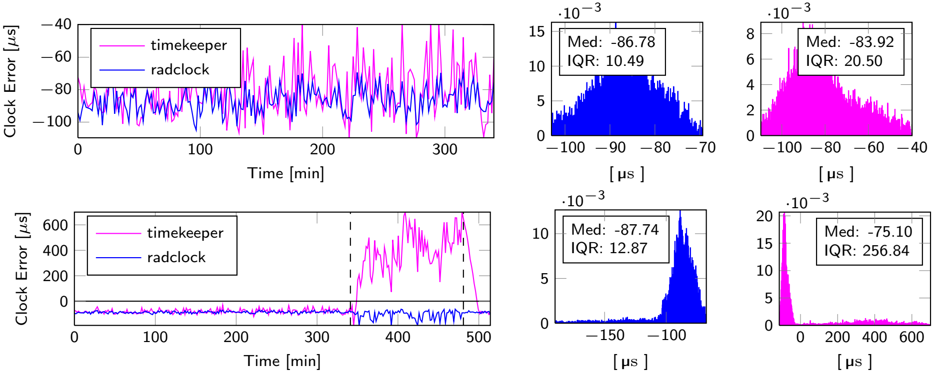
Experiments



- Clocks converged for 2+ hours then 2 hour stress period
- Stress period was a large data transfer across 100 Mbps hub
  - Transfer rate capped at 45Mbps
  - RTT of timing packets increases with large outliers

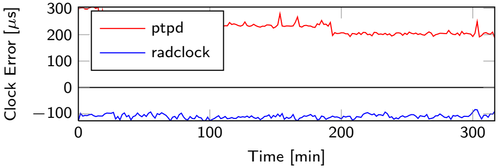
	NTP	PTP SW	PTP HW
90th prctile [μs]	480	340	240
90th pctile stress [ms]	10.7	10.8	0.40

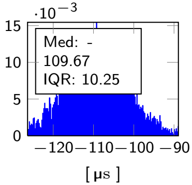
RADclock (NTP) vs. *timekeeper* (NTP) on Linux

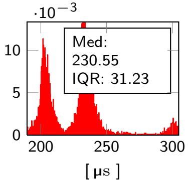


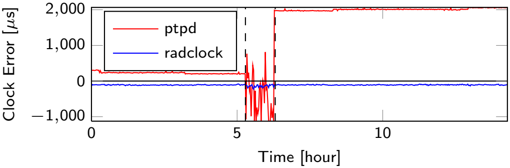
- *RADclock*'s median error is consistent with the network asymmetry seen by the DAG
- *timekeeper* affected by stress, but quickly recovers

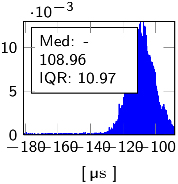
RADclock (NTP) vs. *ptpd* (1588) on Linux

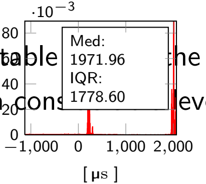






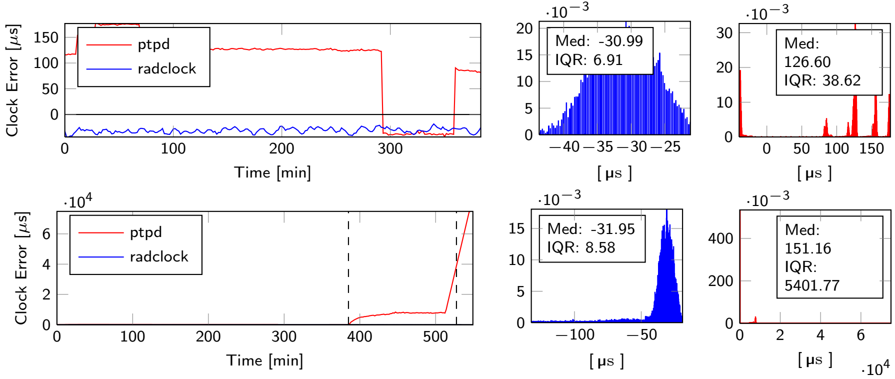






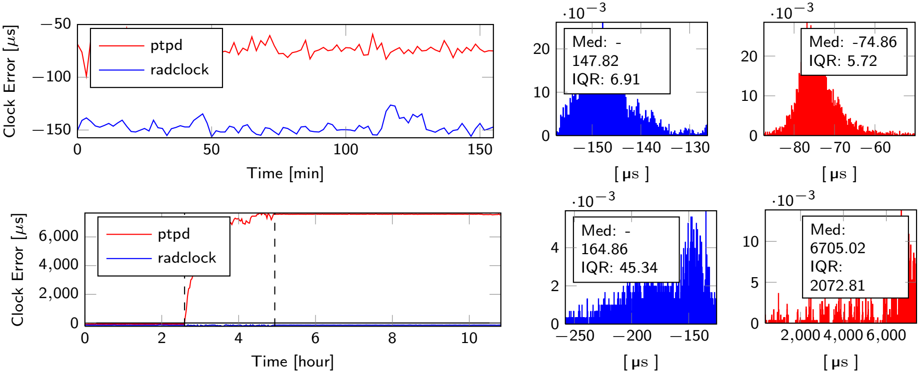
- *RADclock* remains stable … he network stress
- *ptpd* demonstrates a cons… level shift ($2\,\mathrm{ms}$) during the stress period

RADclock (NTP) vs. *ptpd* (1588) on BSD



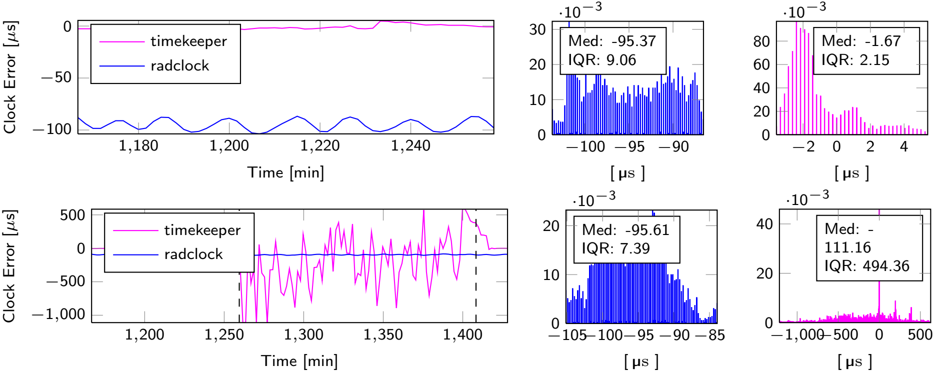
- *ptpd* keeps accumulating error after stress period
- *RADclock* hardly affected by stress

RADclock (1588) vs. *ptpd* (1588) on Linux



- *RADclock* comparable to *ptpd* during non-stressed periods
- Both affected by stress, *RADclock* discards input (implementation problem)

RADclock (1588, hardware) vs. *timekeeper* (1588, hardware) on Linux



- *timekeeper* outperforms *RADclock* during non-stress period
- *timekeeper* affected by stress (IQR from 2 µs to 495 µs)
- Oscillations of *RADclock* due to server-room air conditioning and feed-forward algorithm sliding windows

- We investigated other client software solutions and compared them against *RADclock*
- *RADclock* shows high stability during periods of stress

`http://www.synclab.org/radclock/`