Choosing the Right Clock for the Right Job

Julien Ridoux (jrid@unimelb.edu.au)
Darryl Veitch (dveitch@unimelb.edu.au)

CUBIN

Centre for Ultra-Broadband Information Networks
THE UNIVERSITY OF MELBOURNE

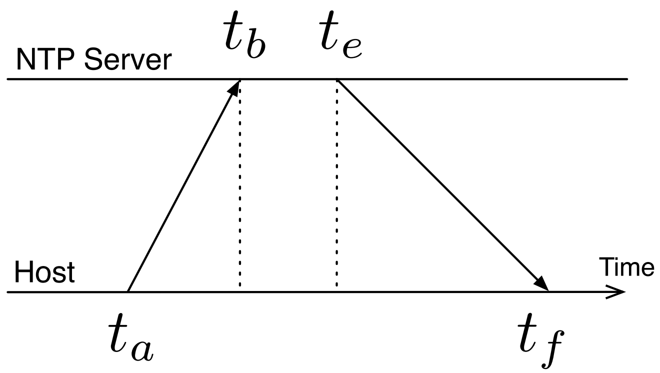
Introduction

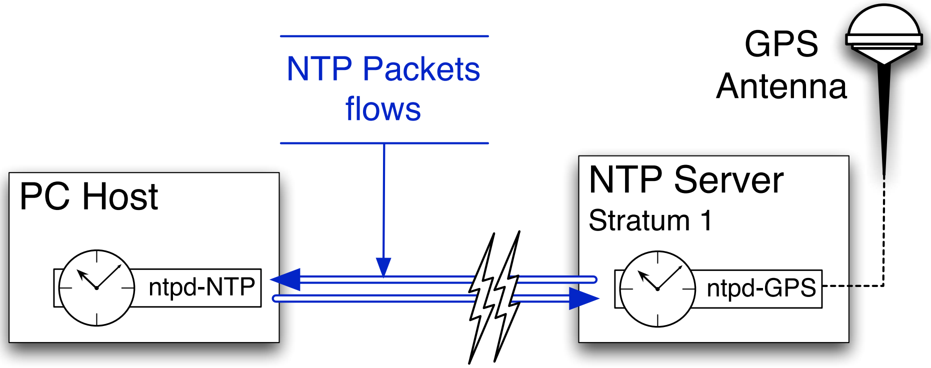
- **Synchronisation is critical to many applications**
  - Telecommunication Industry, Power Industry, Finance ...
- **Network monitoring / Traffic analysis**
  - Accuracy of packet timestamping was not good enough
- **RADclock project**
  - Robust Absolute and Difference Clock
  - Software clock
  - Alternative to ntpd

Synchronisation over the Network

- **Clocks are built upon oscillators (= hardware counters)**
  - HPET, ACPI, TSC
  - Counters are not perfect and drift (temperature variation, ageing...)
  - The job of the synchronisation algorithm is to track drift

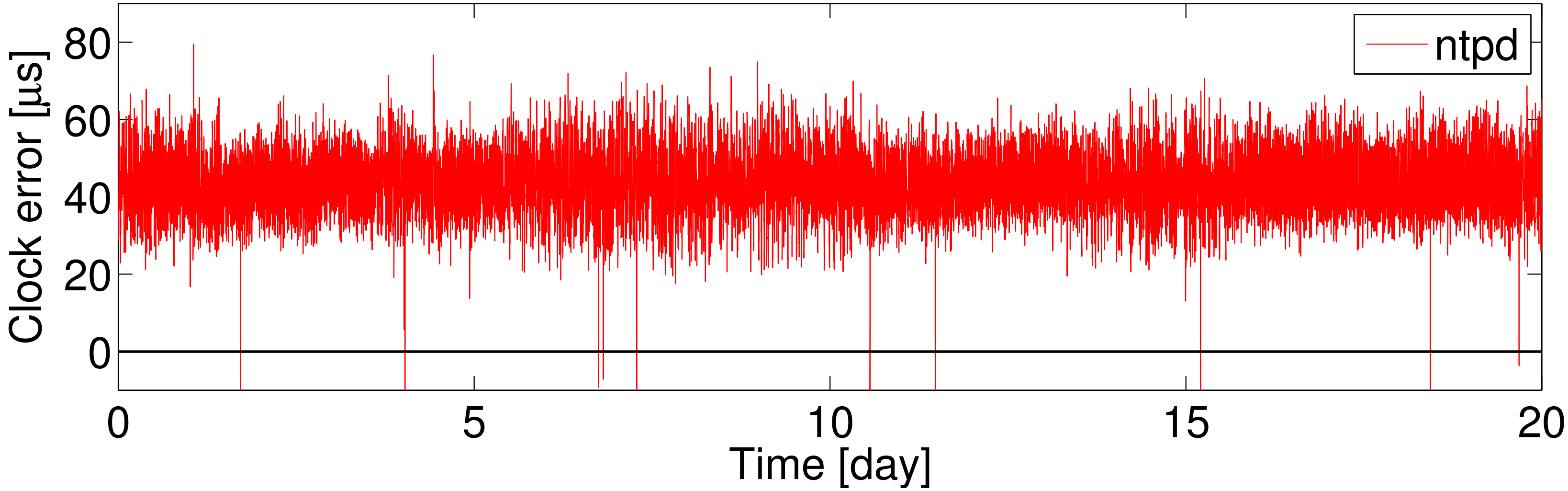
- **Synchronisation over the network**
  - Client send request to a reference clock: “what time is it?”
  - Algo input: 2 server timestamps, 2 client timestamps
  - ntpd has been the solution for the past 25 years +





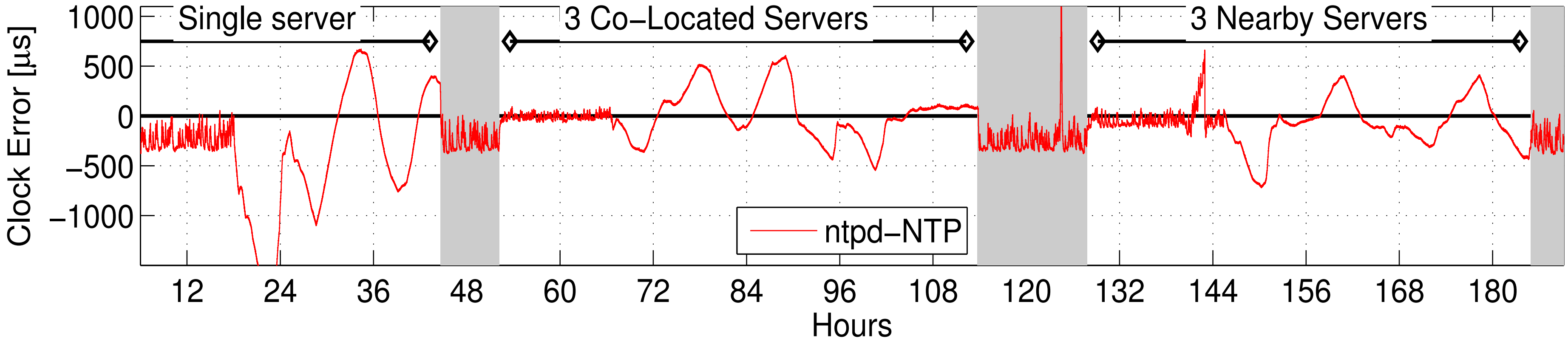
ntpd Performs Well

- **Lab environment experiment**
  - Good time server: Stratum-1, Atomic Clock locked to GPS receiver
  - Client is on the same LAN, barely any traffic
  - Constrained and small polling period: 16 sec



ntpd does NOT Perform Well

- **Same setup but changed configuration**
- **Follow ntpd's instructions**
  - Multiple servers
  - Relax constraint on polling period



ntpd Performs ... ?

- **No guarantee on ntpd's performance**

- **Feedback design**
  - Timestamps are input to ntpd clock correction algo
  - ntpd adjusts the system clock that produces timestamps
  - What if it get it wrong ... ?

- **Convergence ...**
  - takes time
  - may never reach acceptable level
  - is not guaranteed when faced with very variable network noise

An Alternative Exists

- **Feed-Forward approach**
  - Decouples timekeeping from timestamping
    - Timestamp events using “RAW” counter values
  - Previous clock adjustments do not influence current one

- **Advantages**
  - Robust clock-independent filtering
  - Can define several clocks
  - Simpler kernel support

Kernel Modifications

- **Modify Timecounters[1] abstraction**
  - New cumulative counter
  - 64 bit wide: does not wrap around
- **Timestamping function returns cumulative counter value**
  - RAW timestamps
- **Feed-Forward clock data to be maintained in the kernel**
  - Convert RAW timestamps to timeval / timespec
  - Pushed by the RADclock synchronisation daemon

[1] - "Timecounters - Efficient and Precise Timekeeping in SMP kernels" (P-H. Kamp)

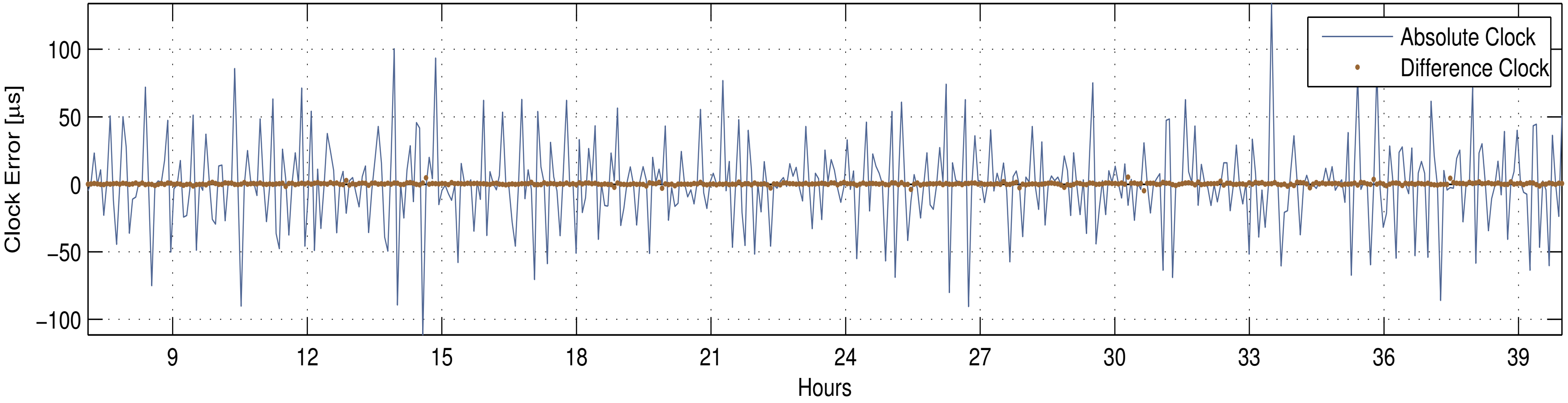
Clock Models in Practice

- **ntpd: actively adjusts clock rate to track drift**
  - $C_{ntpd}(t) = \text{Period}(t) * \mathbf{(}\text{counter}(t) - \text{counter}(t_{old})\mathbf{)} \;+\; C_{ntpd}(t_{old})$
  - Period changes on each update ⇒ no rate stability

- **RADclock: estimates clock rate and tracks drift**
  - 2 clocks can be defined: difference and absolute clock
  - $C_d(t_1,t_2) = \overline{\text{Period}} * \mathbf{(}\text{ffcounter}(t_2) - \text{ffcounter}(t_1)\mathbf{)}$
  - $C_a(t) = \overline{\text{Period}} * \text{ffcounter}(t) + \text{Offset}(t)$
  - $\overline{\text{Period}}$ is a long term average (barely changes) ⇒ rate stability
  - Offset tracks the drift ⇒ changes on every clock update

RADclock: Difference Clock

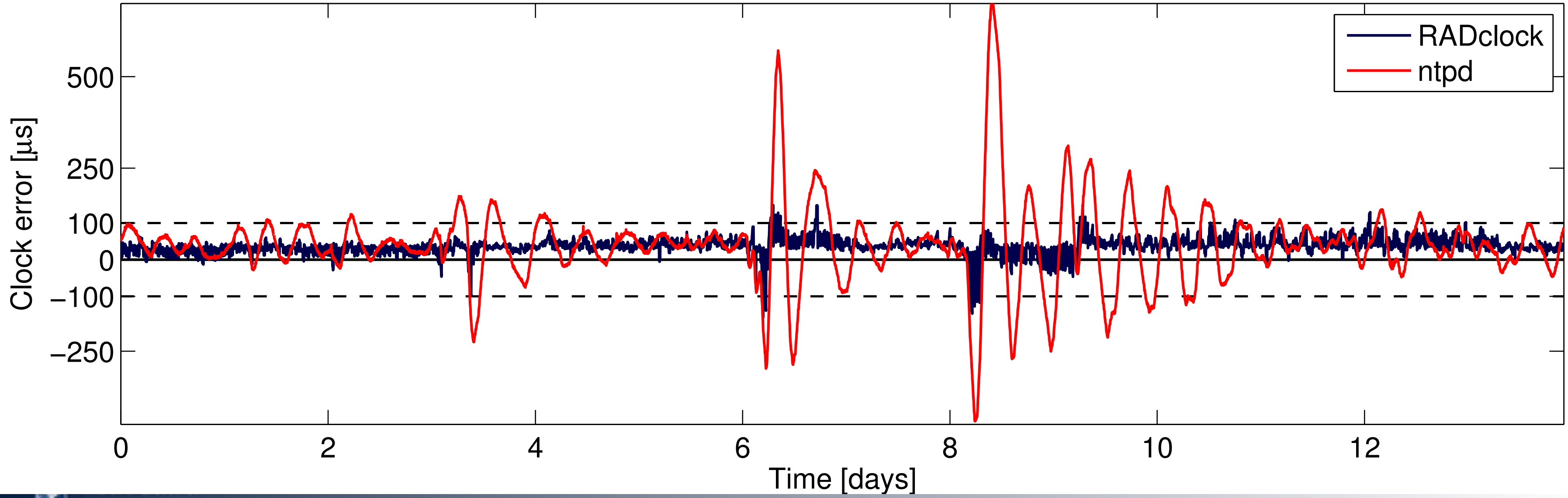
- **Difference clock**
  - $C_d(t_1,t_2) = \overline{\text{Period}} * \textbf{(}\text{ffcounter}(t_2) - \text{ffcounter}(t_1)\textbf{)}$
  - Can be defined since RADclock ensures stable rate
- **Use the difference clock to measure (small) time intervals!**
  - In Kernel Pulse-Per Second Timestamping
  - Timestamp a 1 sec interval with Absolute and Difference clock



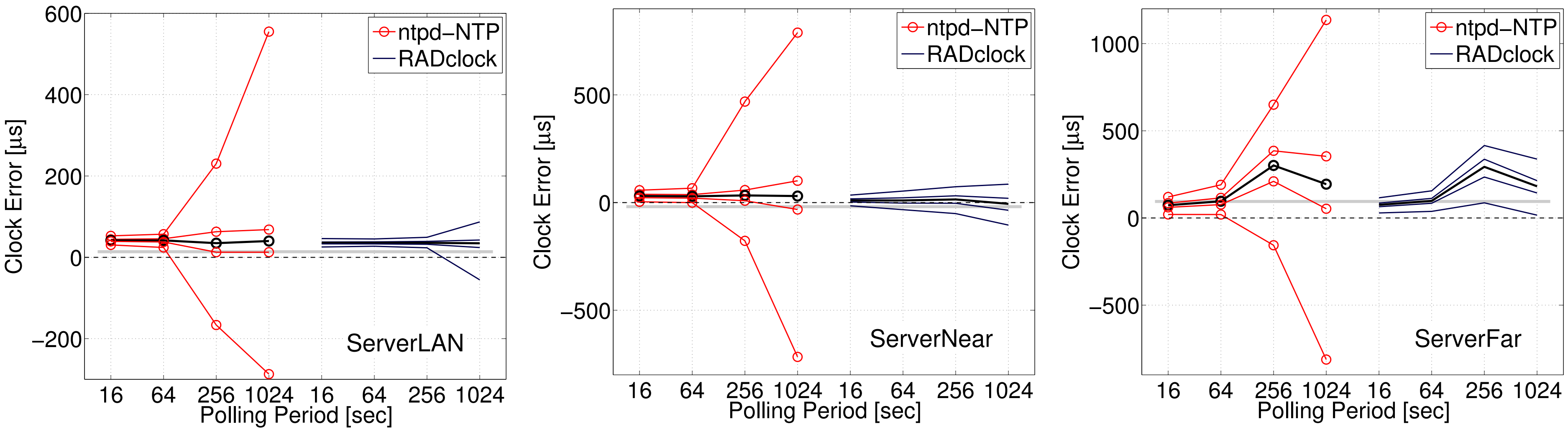
RADclock: Absolute Clock

- **Absolute Clock**
  - $C_a(t) = \overline{\text{Period}} * \text{ffcounter}(t) + \text{Offset}(t)$

- **Robust RTT filtering based on the Difference Clock**
  - Stratum-1 on a LAN
  - Polling period 1024 sec, no cross traffic
  - RADclock and ntpd share the same flow of NTP packets



RADclock: Absolute Clock



- **Polling Period and Server Distance**
  - Each dataset is over 1 month long

- **RADclock outperforms ntpd all the time**
  - Better performance
  - More robust

Fast Timestamping

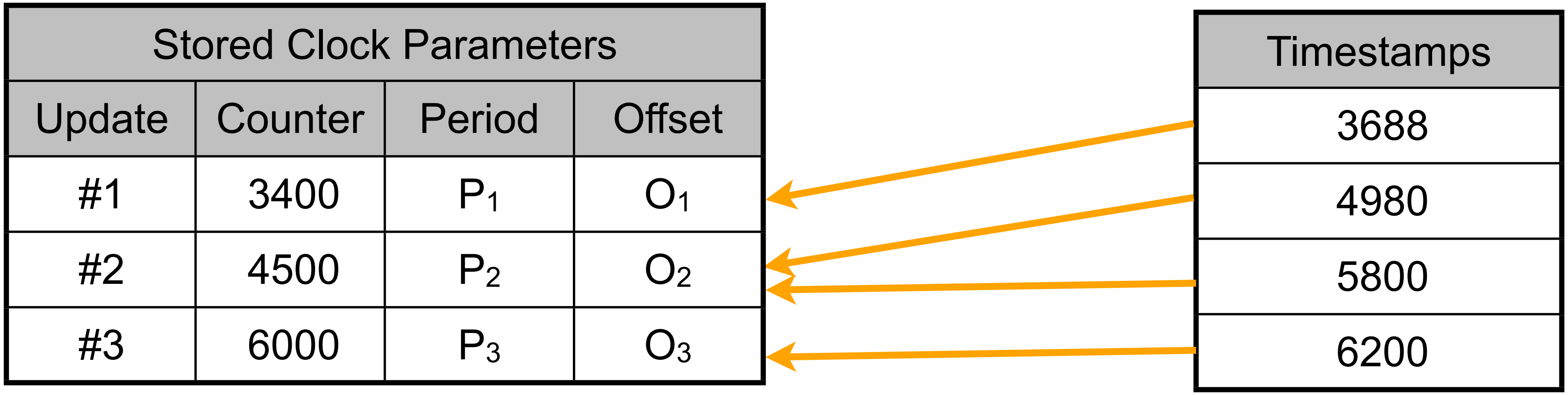
- **It is not only about performance / robustness**

- **Timestamping and Timekeeping are decoupled**
  - Counter values do not have to be converted to time right away
  - Clock parameters are updated on every NTP packet only!

- **Fast timestamping and delayed conversion**
  - Store RAW counter values only
  - Retrieve clock parameters when you are less busy
  - Convert counter values to time

Replaying Time

- **Going further: timestamps created in post-processing**



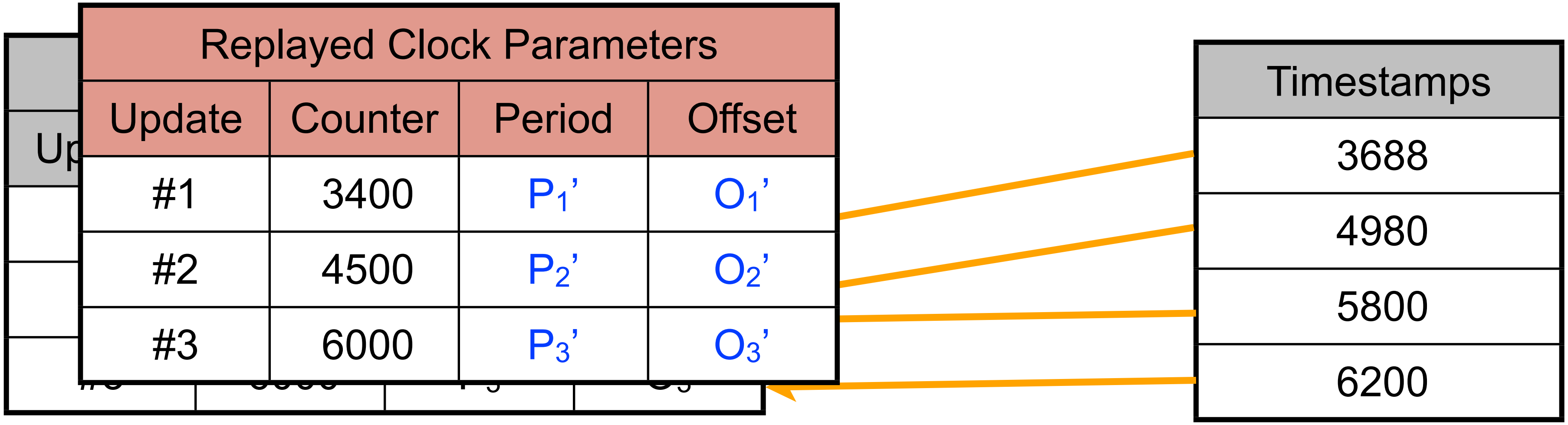
Stored Clock Parameters			
Update	Counter	Period	Offset
#1	3400	P_1	O_1
#2	4500	P_2	O_2
#3	6000	P_3	O_3

Timestamps
3688
4980
5800
6200

Replaying Time

- **Going further: timestamps created in post-processing**

- **You can even replay (and improve) the clock time!**
  - Make a better job in post-processing
  - You have access to all NTP packets



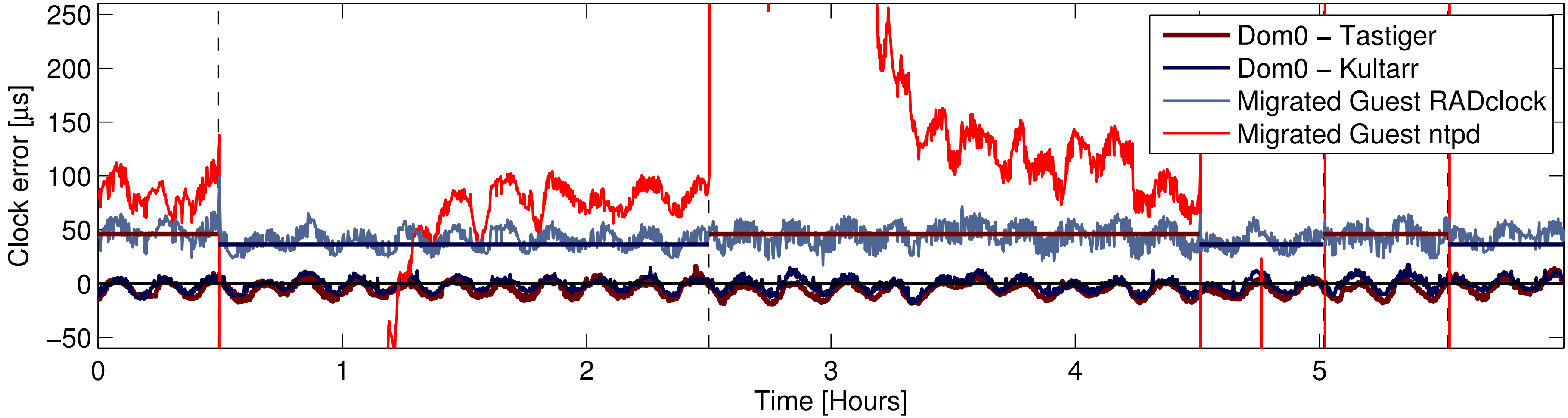
Replayed Clock Parameters			
Update	Counter	Period	Offset
#1	3400	P_1'	O_1'
#2	4500	P_2'	O_2'
#3	6000	P_3'	O_3'

Timestamps
3688
4980
5800
6200

Virtualization

- **Feed-Forward requires a simpler kernel support**
  - Push clock parameters (period and offset) to kernel

- **Application to clock dependent synchronisation in Xen**
  - Dom0 runs the sync' daemon, write clock params to XenStore
  - DomU reads parameters from XenStore
  - Live migration works: simply read from "migrated XenStore"



Summary

- **Feed-Forward approach has many advantages**
  - the difference clock can be defined
  - the absolute clock is a more robust clock
  - time can be replayed
  - a simpler kernel support
  - it enable clock dependent mode for virtualization
- **Feedback and Feed-Forward can co-exist**
  - Feedback is still very good for local synchronisation (GPS, etc)

Looking forward

- **What we have now:**
  - RADclock daemon is an implementation of Feed-Forward clock
  - http://www.cubinlab.ee.unimelb.edu.au/radclock/
  - FreeBSD kernel support
    - Developed since FreeBSD 5.3
    - Still very RADclock / prototype oriented

- **What is coming next?**
  - FreeBSD Foundation project
    - Develop generic support for Feed-Forward clock
    - Give users the choice to use Feed-Forward or Feedback
    - Give users the choice to use an Absolute or Difference Clock
    - Develop a fully functional system clock
  - Get you guys to try it !