Outline for Today

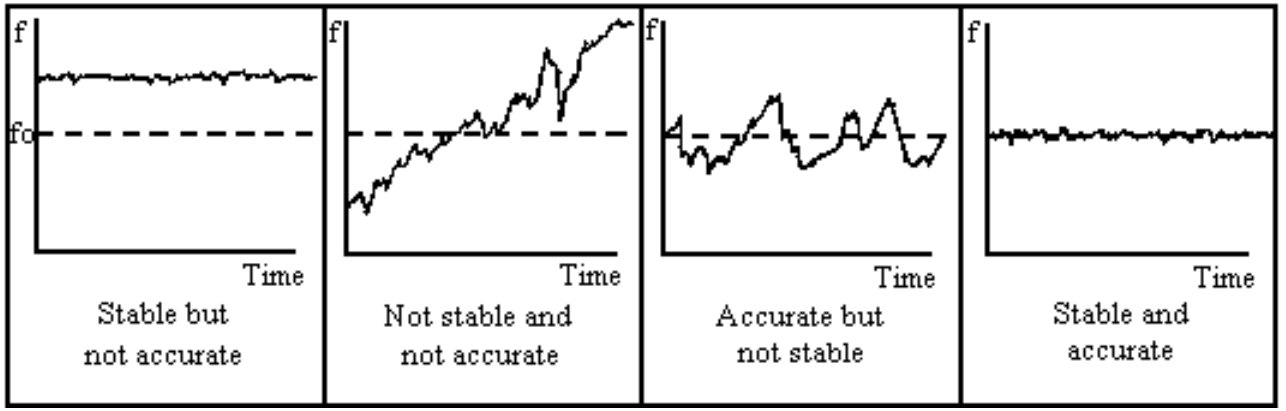
- Objectives:
  - Time and Timers
- Administrative details:
  - Talk on learning at 4 in 130 North Building
  - Questions?

Uses of Time

- Coordinating events
  - Synchronized clocks
- Measurements – durations of activities
  - Stability – ability to maintain constant frequency
    - Environmental factors (temperature) or age
    - Synchronization protocols that adjust clock
- Driving periodic events
  - Granularity (frequency)
- Scheduling dynamic events at a particular time in the future.
  - Accuracy
  - Relative or absolute time?

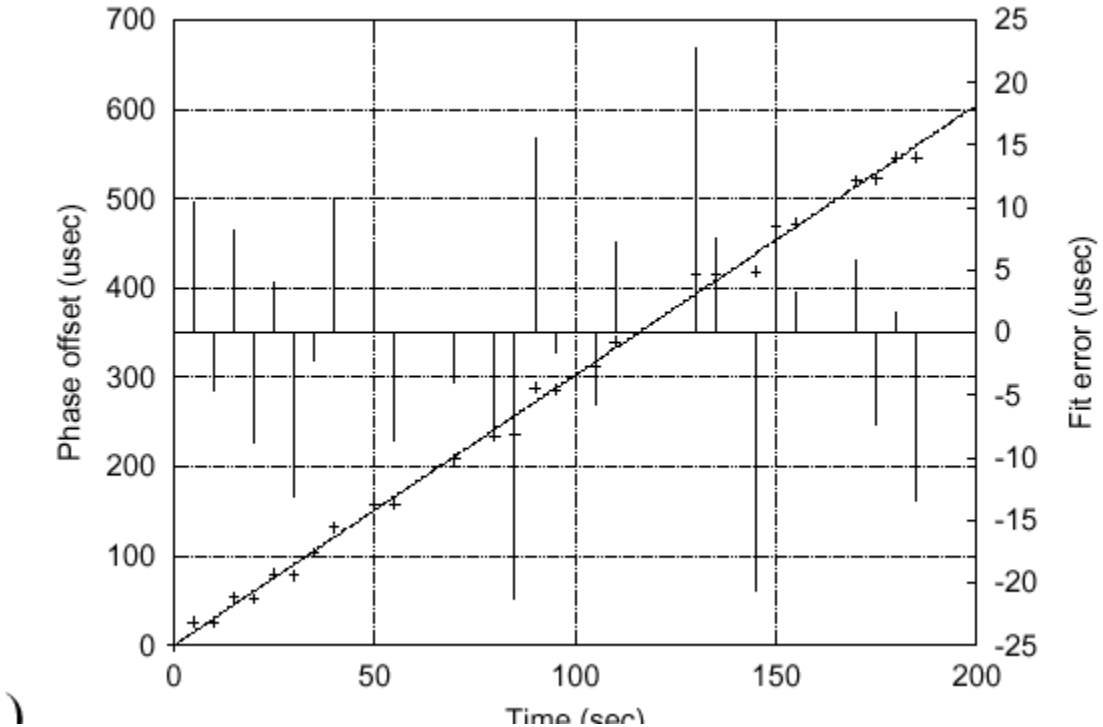
Time Definitions

- Clock stability – how well it maintains a constant frequency
  - Short term – temperature
  - Long term – aging of oscillator
- Clock accuracy – how well its frequency and time compare with standard



Time Definitions

- Offset – time difference between 2 clocks
- Skew – frequency difference between 2 clocks



Timer Basics (Linux)

- Real time clock (RTC) keeps track of time even when system is off – boot-time initialization
- System timer – provide periodic interrupts
  - Programmable interrupt timer running at tick rate of HZ frequency
    - Time update (jiffies, wall clock time), do accounting (resource usage), dispatch events that are due (dynamic timers), rescheduling
  - Jiffies – number of ticks since reboot
  - Time of day
    - xtime structure – contains seconds since Jan 1 1970; wall clock time based on that.
- Delaying execution by looping `udelay(us)` or sleeping `schedule_timeout(s*HZ)`

Dynamic Timers

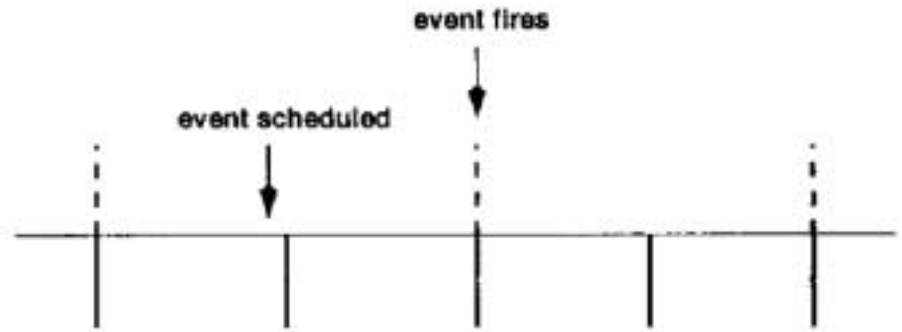
- Created and destroyed dynamically
- Handler is run when tick count is >= expiration time.
- ```
  init_timer(&mytimer);
  mytimer.expires = jiffies + delay;
  mytimer.data = 0; //arg passed to handler
  mytimer.function =myhandler;
  ```
- `add_timer(&mytimer);`
- Can change `mod_timer` or remove `del_timer_sync` timers
- Timers are stored in buckets depending on how far into the future they should expire.
- Run asynchronously with respect to other code – protect shared data appropriately.
  ```

Soft Timers

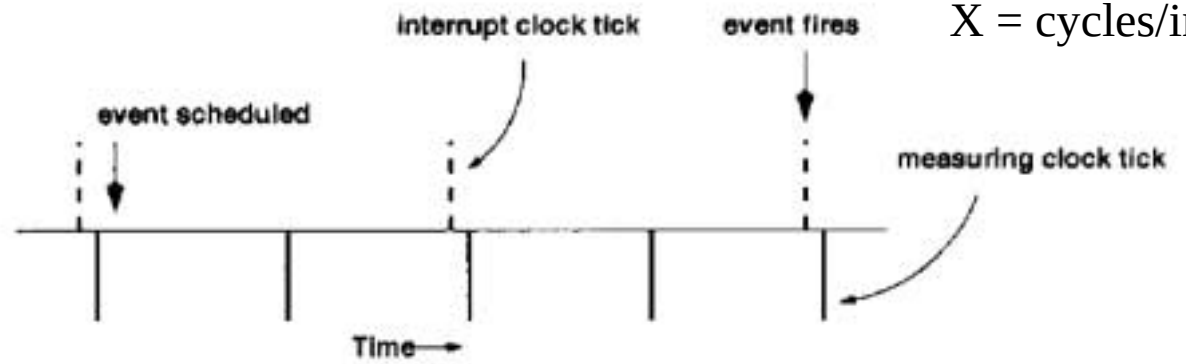
Aron & Druschel

- Goal: to provide usec granularity events with low overhead.
  - Do not want timer interrupts at that granularity
- Approach: To leverage trigger points when execution has already been interrupted – amortize context switch and cache pollution already incurred by other causes.
  - End of syscall processing, end of exception handler, end of executing interrupt handler, during CPU idle loop
  - Bounded overrun if a trigger point doesn't happen – backup hardware interrupt set

Accuracy



Example of minimum Event Time (just larger than T=1)



Example of maximum Event Time (just smaller than T+X+1=4)

X = cycles/interrupt

Fig. 1. Lower and upper bounds for event scheduling.

Overhead

Table I. Per-Event Timer Costs with Null Event Handler

	Alpha-500	8253/PII-300	8253/PIII-500	APIC/PIII-500	Soft Timers
Overhead (μsec)	8.64	4.45	4.36	0.8	$\approx$0

	APIC/PIII-500	Soft Timers
Overhead (μsec)	5.1	3.5
Icache-misses (x10^6)	153.2	149.7
Dcache-misses (x10^6)	551.4	377.9
ITLB-misses (x10^6)	18.25	17.00

Timer costs with synthetic event handler
scheduled every 10usec

Synthetic event handler touches 50 cache lines, 2 instr cache lines

Trigger Occurrence

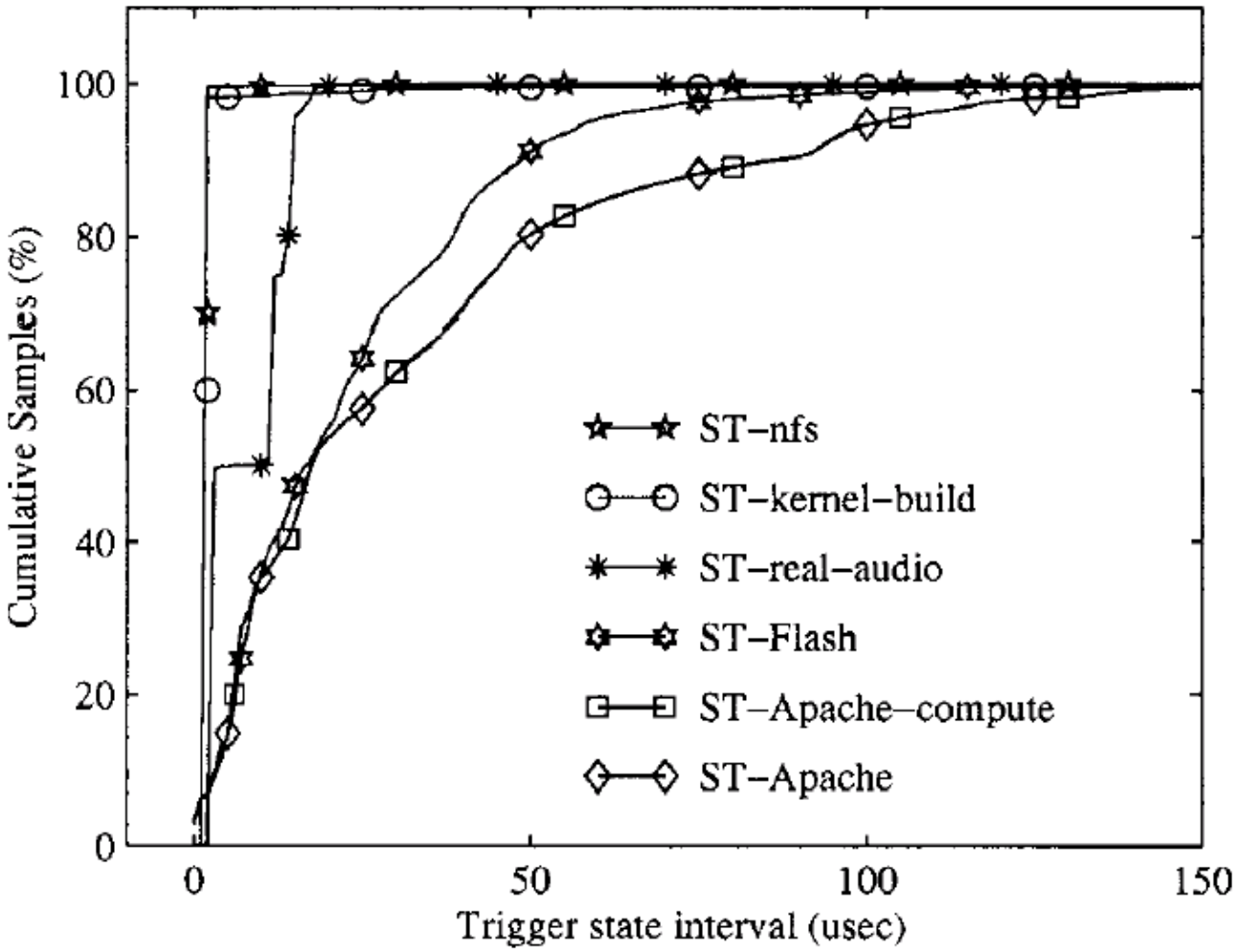


Fig. 2. Trigger state interval (CDF), 300MHz PII.

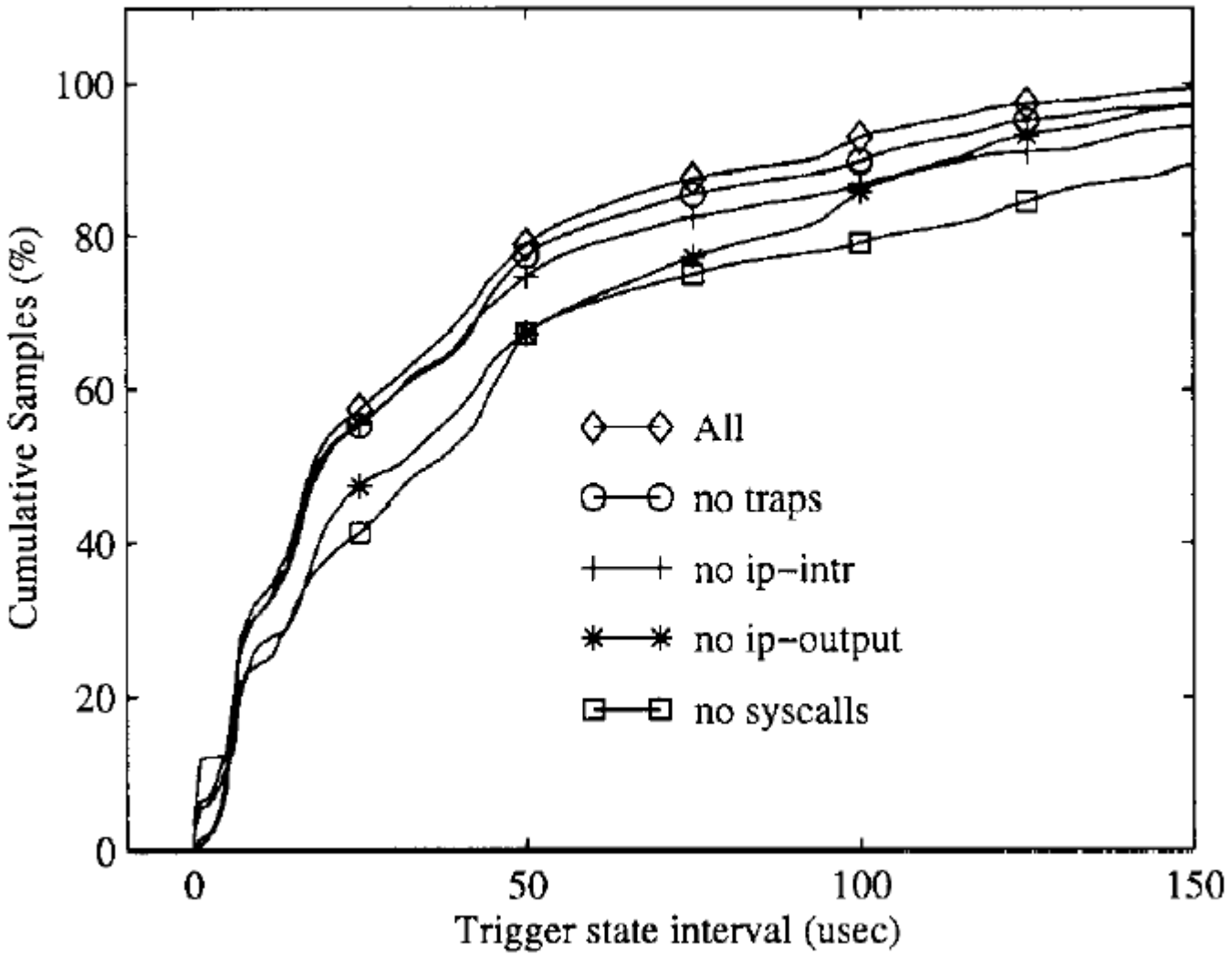
Trigger Sources

Table V. Trigger State Sources

Source	Fraction of samples (%)
syscalls	47.7
ip-output	28
ip-intr	16.4
tcpip-others	5.4
traps	2.5

Impact of Trigger Sources

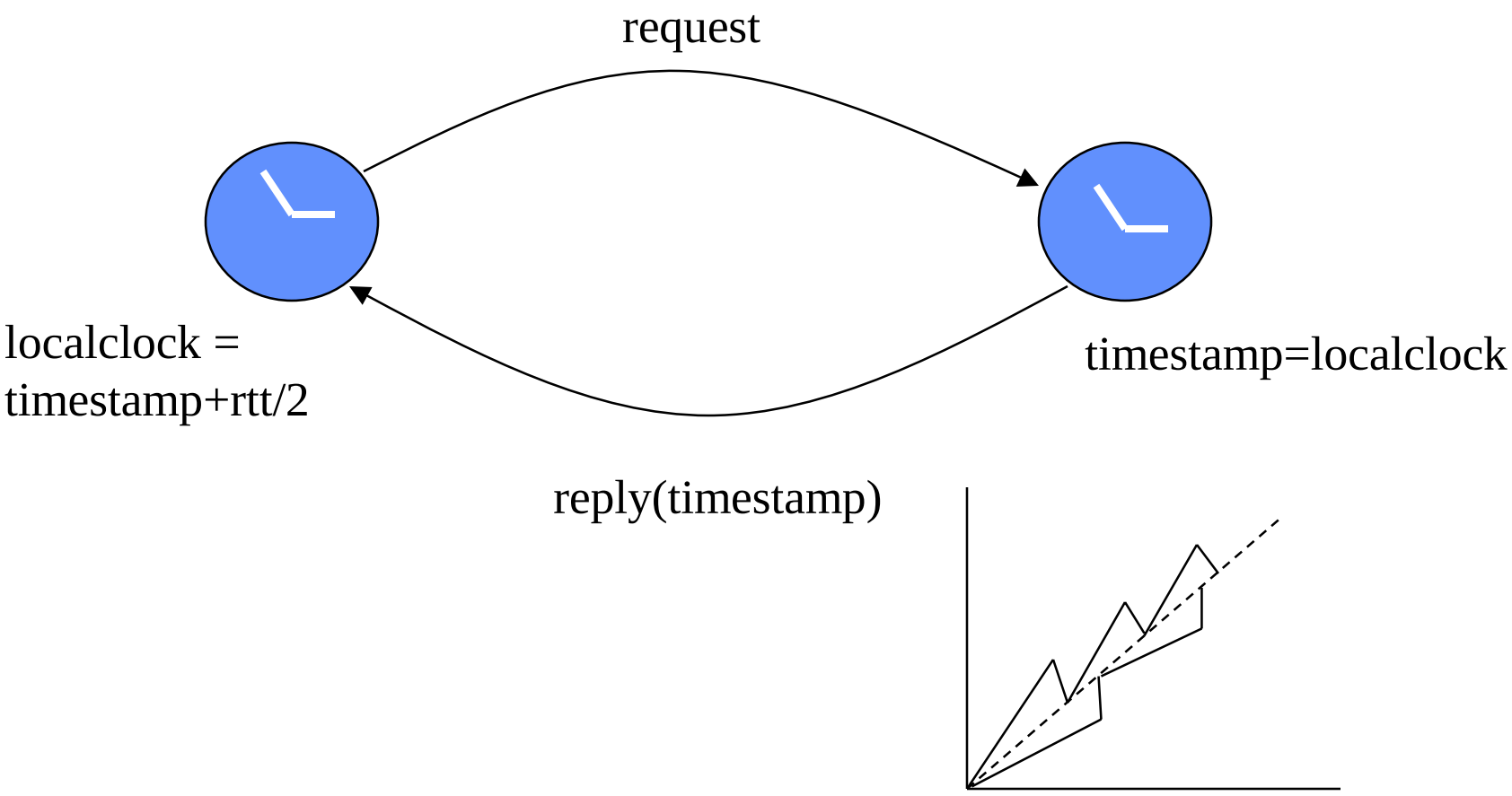
(ST-Apache)



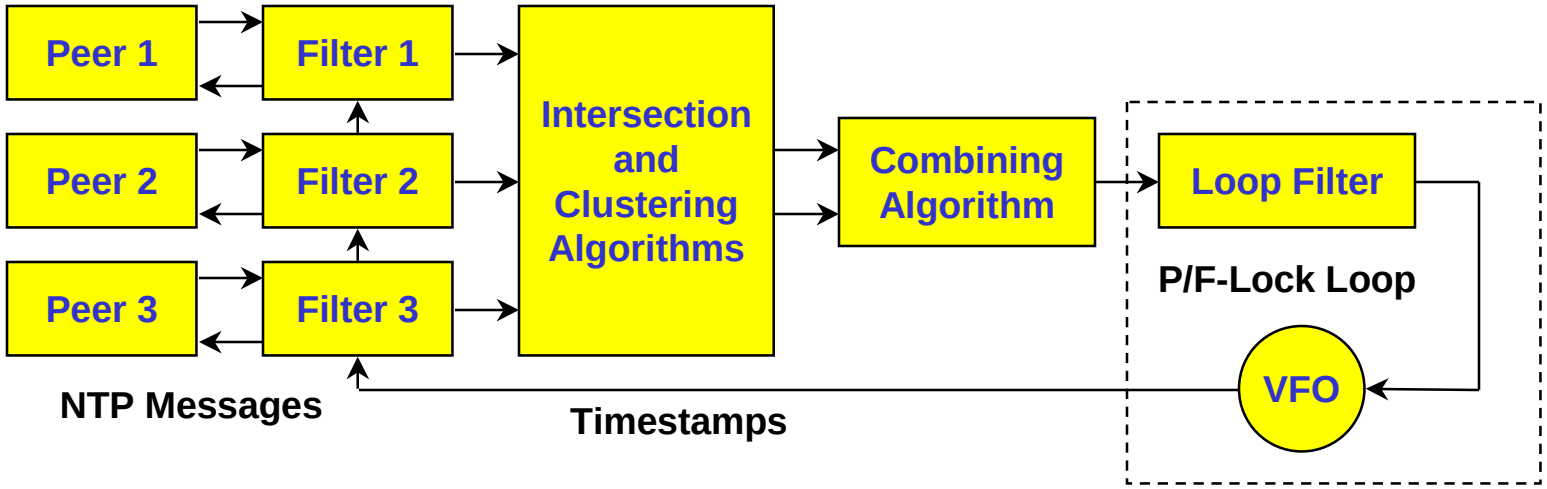
Target Applications

- Rate-based clocking in the networking system
  - Schedule transmissions according to desired rate
  - If achieved rate falls below target, schedule to allow maximal allowable burst
- Polling network interfaces

Naive Clock Synchronization



How NTP works



- Multiple synchronization peers provide redundancy and diversity
- Clock filters select best from a window of eight clock offset samples
- Intersection and clustering algorithms pick best subset of servers believed to be accurate and fault-free
- Combining algorithm computes weighted average of offsets for best accuracy
- Phase/frequency-lock feedback loop disciplines local clock time and frequency to maximize accuracy and stability