Improving Time Measurement in TTM

Thomas Wana, RIPE NCC

Mark Santcroos, RIPE NCC

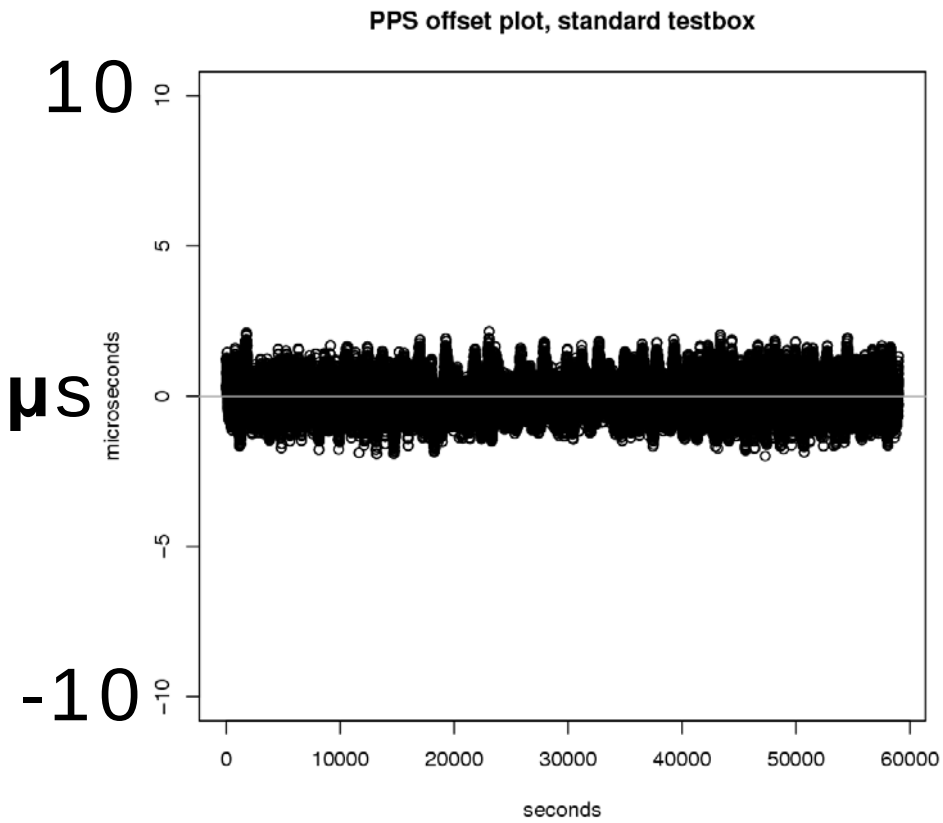
Agenda

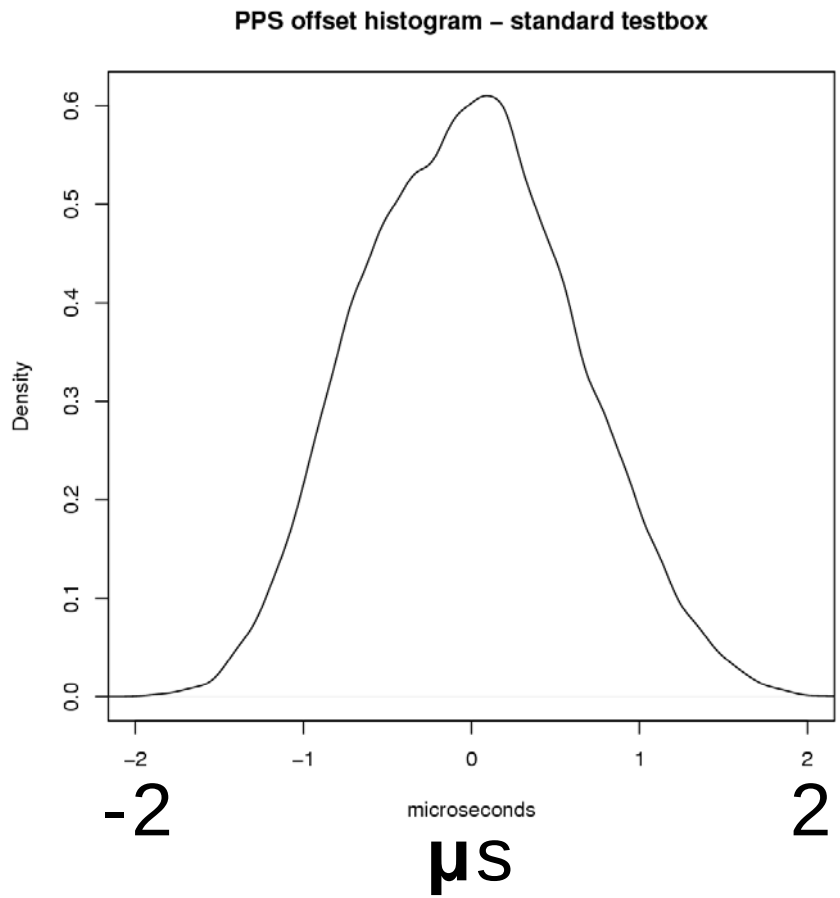
- TTM - Current performance
- NTP performance
- Kernel level timestamping
- Device polling
- DAG-TTM

Motivation

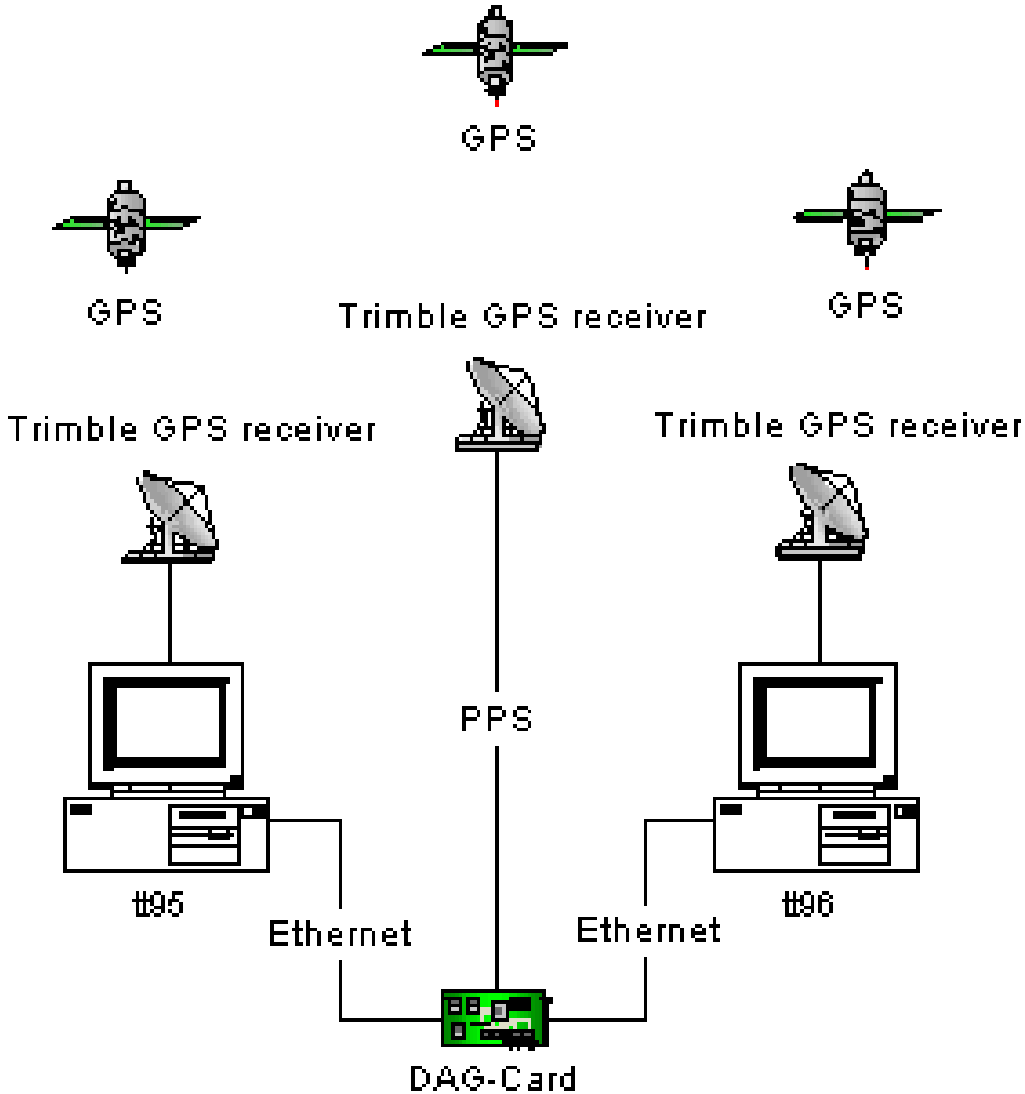
- TTM started in 2000
- New developments since then

TTM performance – PPS offsets

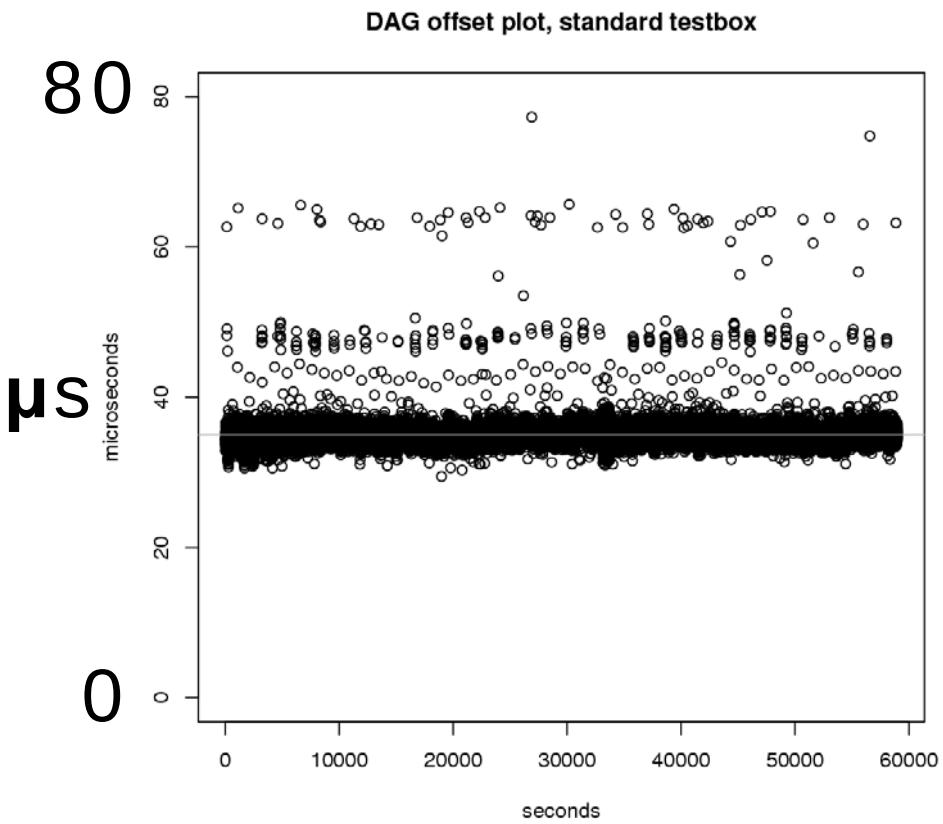


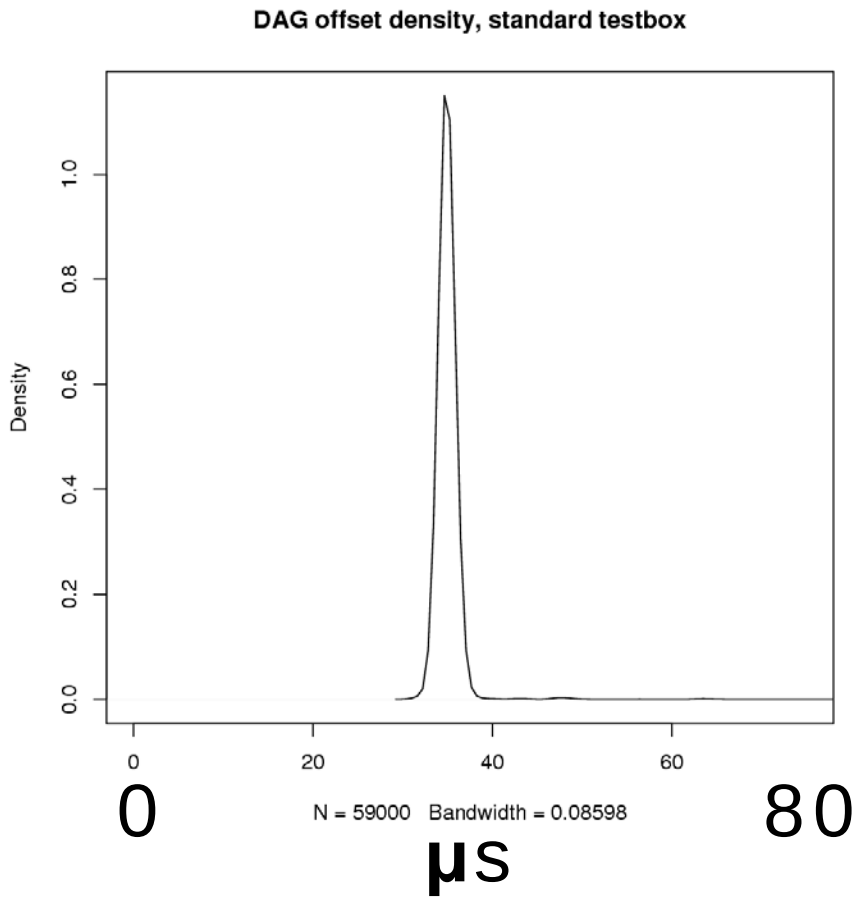


TTM performance – DAG setup

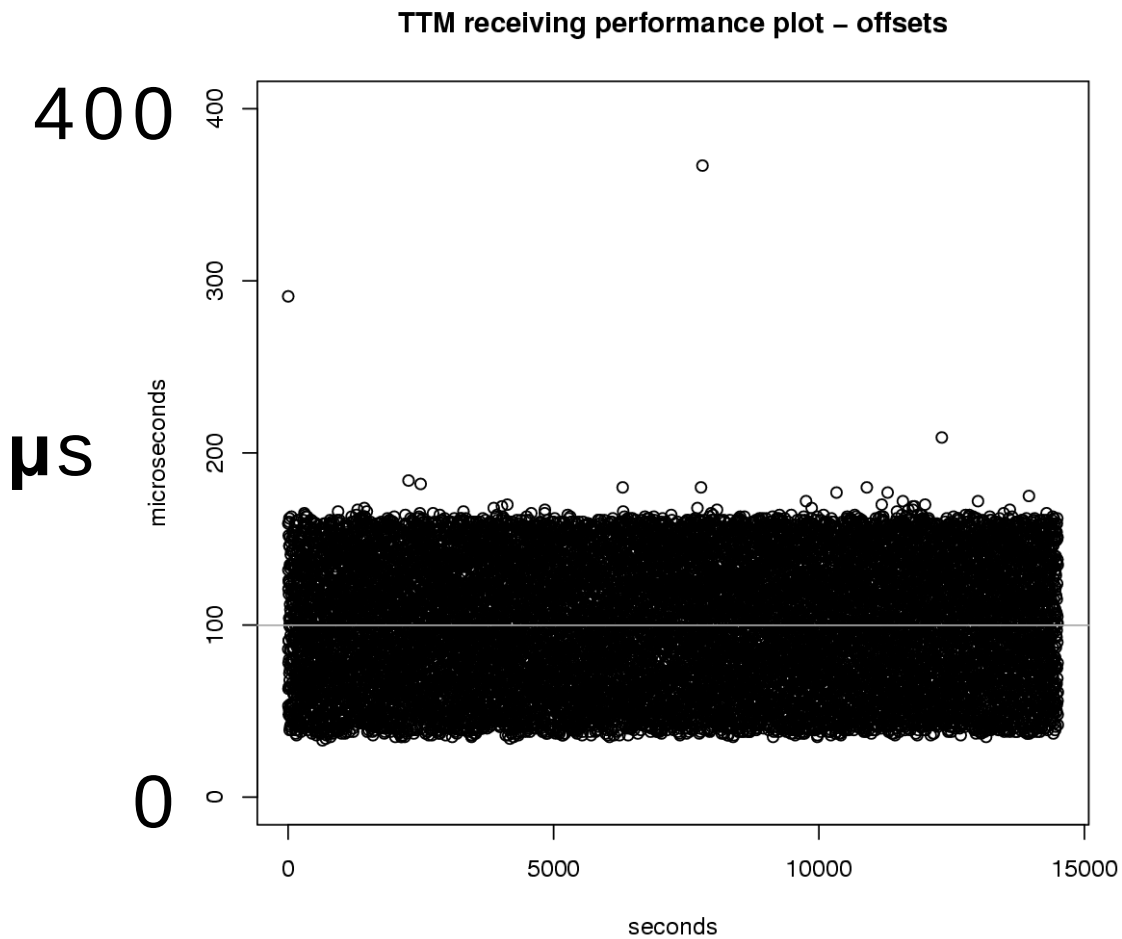


TTM performance – DAG offsets





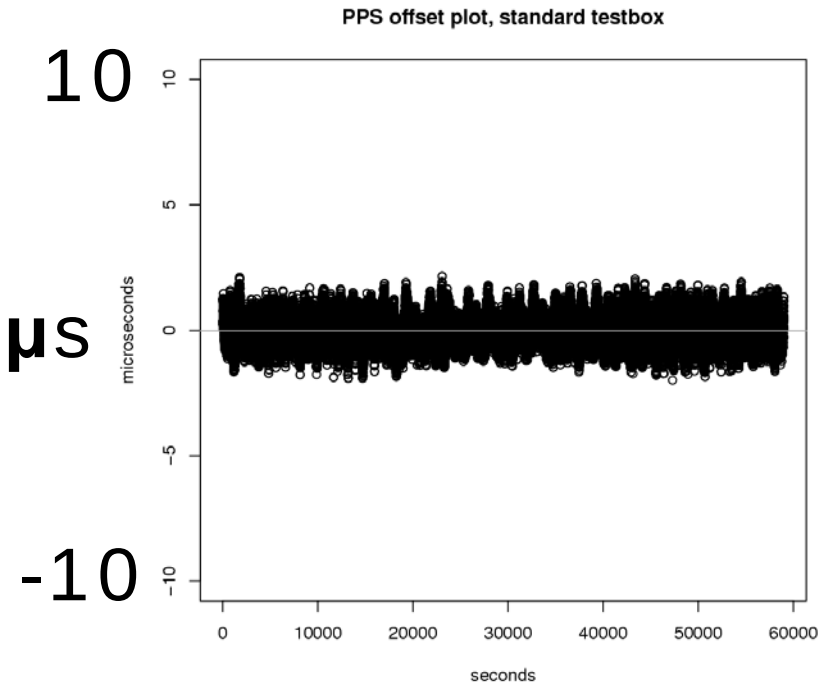
TTM performance – receiver

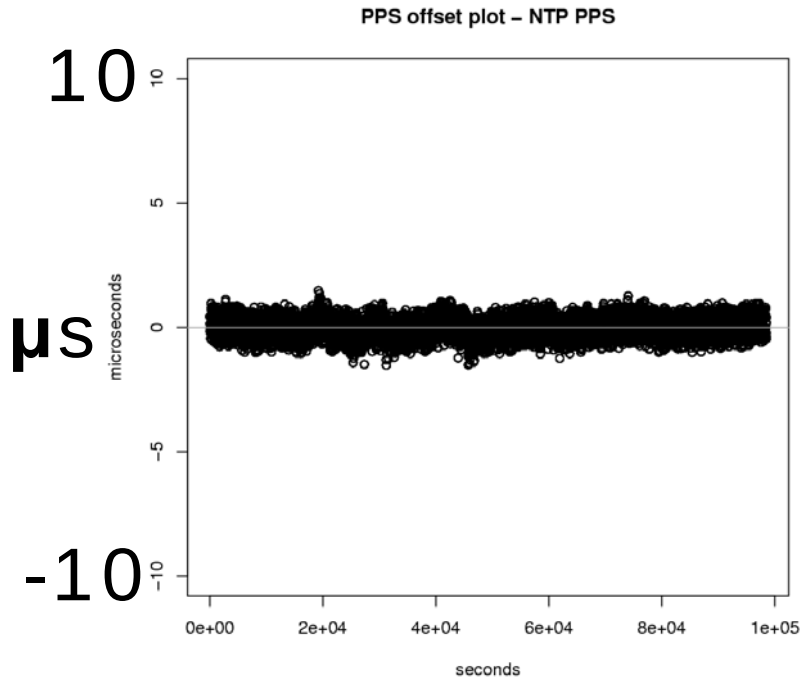


NTP performance

- TTM uses PPS_SYNC
- NTP provides a PPS (ATOM) driver

NTP performance - ATOM

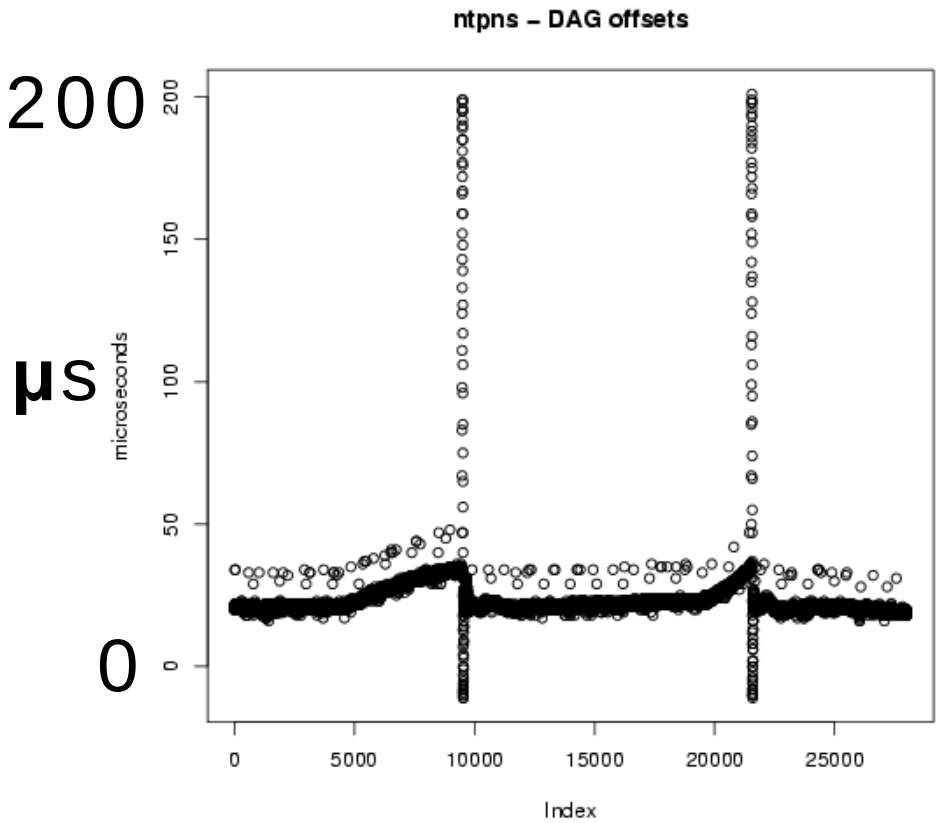




	Med.	*Mean*	*IQR*	*1-99*	*sdev*
PPS_SYNC µs	-0.017	-0.013	0.894	2.786	0.628
ATOM µs	0.007	-0.005	0.689	1.866	0.446

NTP performance - ntpns

- New NTPd by Poul-Henning Kamp for FreeBSD 5
- No monotonic timers for FreeBSD 4
- Not considered



NTP performance - Conclusions

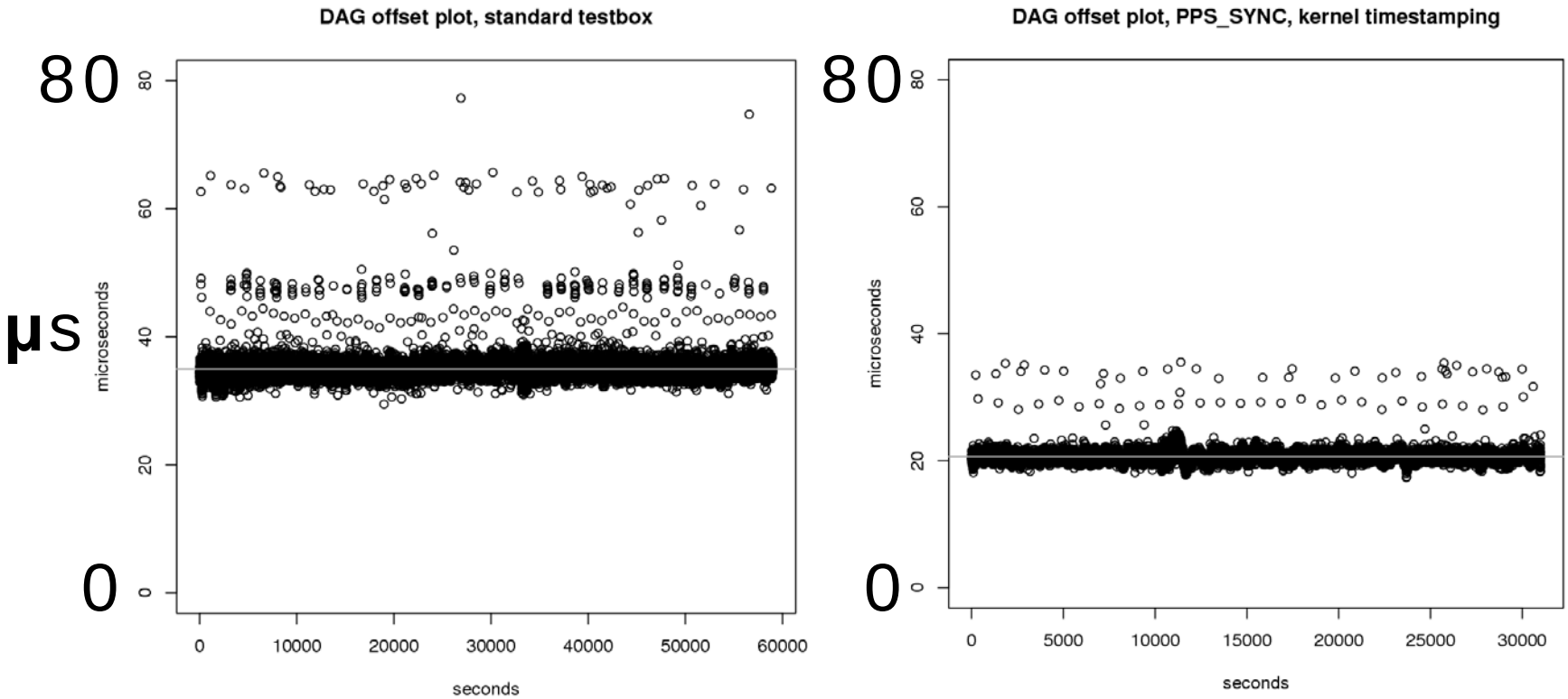
- Using ATOM instead of PPS_SYNC → slight improvements
- Easy to switch
- No stability concerns

Kernel level timestamping

- TTM sender: userland timestamping
- TTM receiver: already uses kts (BPF)
- Kernel patch: kernel timestamping
- ioctls:

```
setsockopt(sock, SOL_SOCKET, SO_RIPETSENABLE, 1);
setsockopt(sock, SOL_SOCKET, SO_RIPETSOFFSET, 40);
sendto(sock, buf, len, 0, &to, sizeof(to));
```

Kernel level TS – sending results

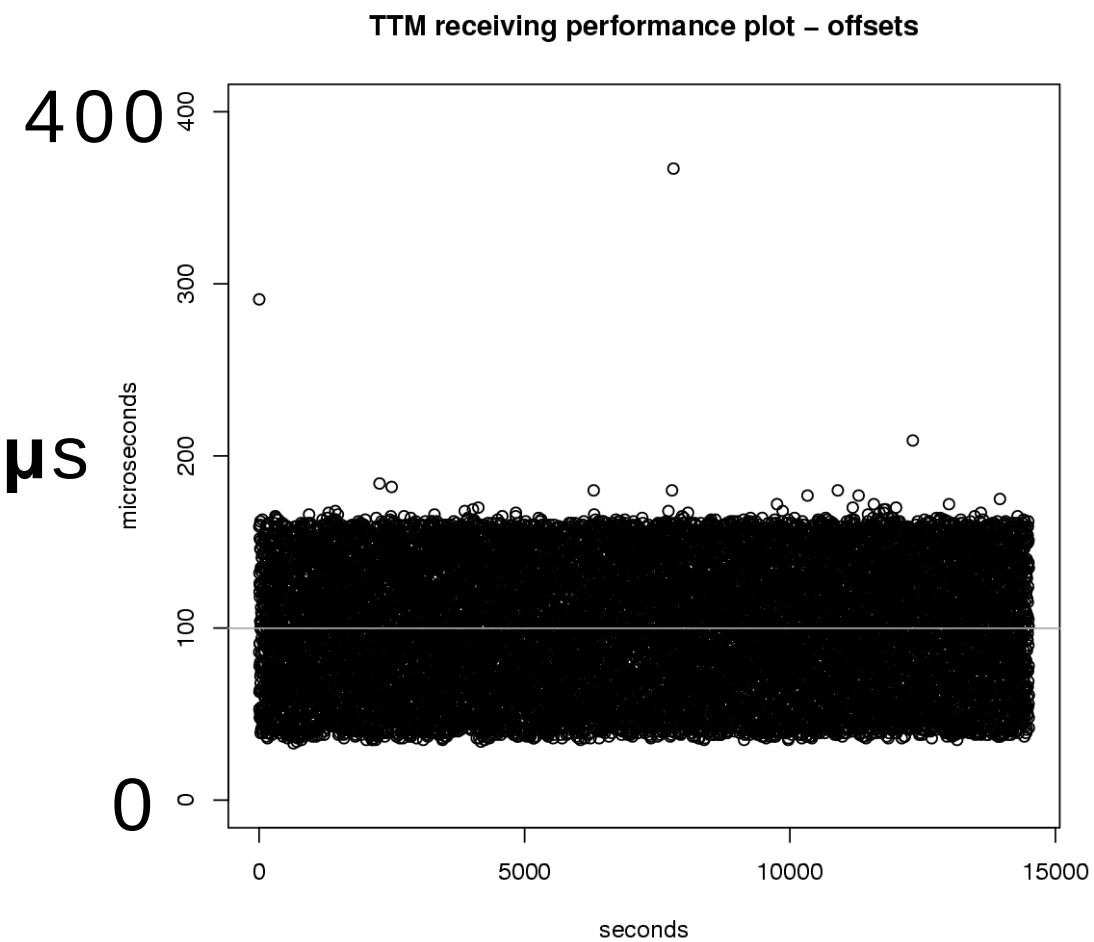


	Med.	*Mean*	*IQR*	*1-99*	*sdev*
Userland µs	33.65	33.75	0.96	3.92	1.64
Kernel µs	20.87	20.90	0.77	2.75	0.75

Kernel level TS - Conclusions

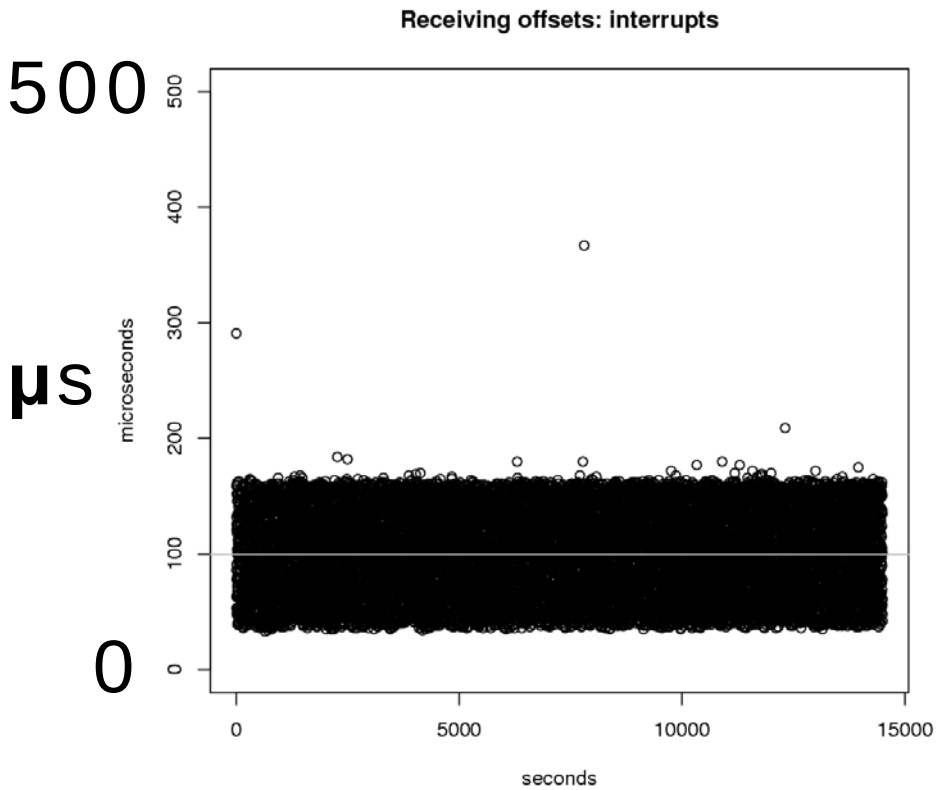
- Improves sending side: offset, variation
- Patch is small and simple
- TTM Software changes minimal
- Production ready

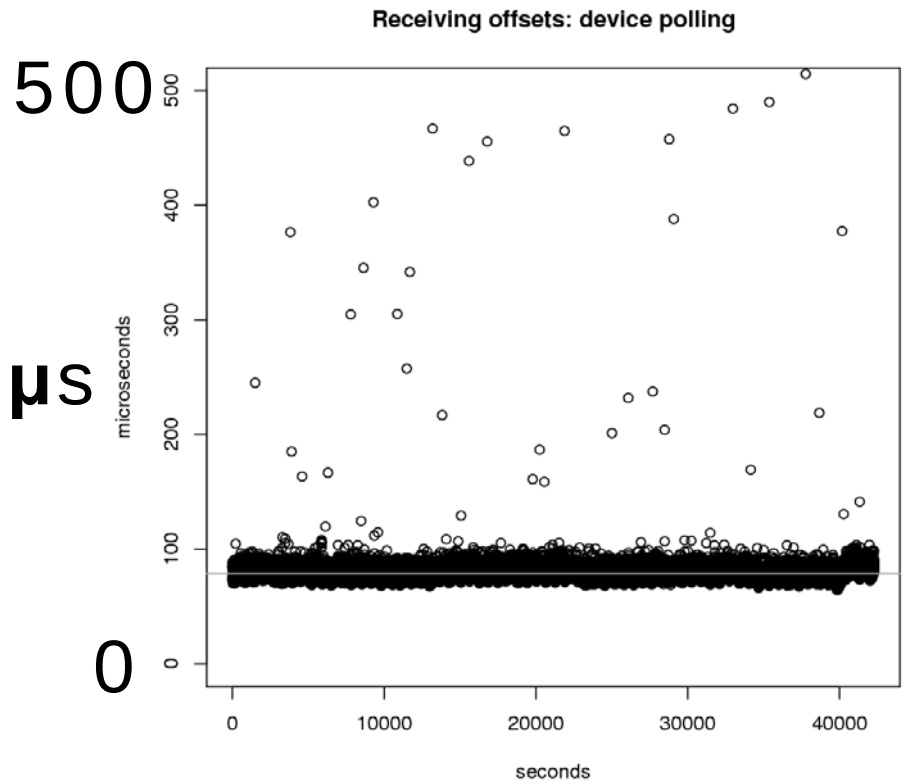
Device polling



- Slow network interrupts
- Device polling: no interrupts used
- Polling in the timer interrupt and idle loop

Device polling - results



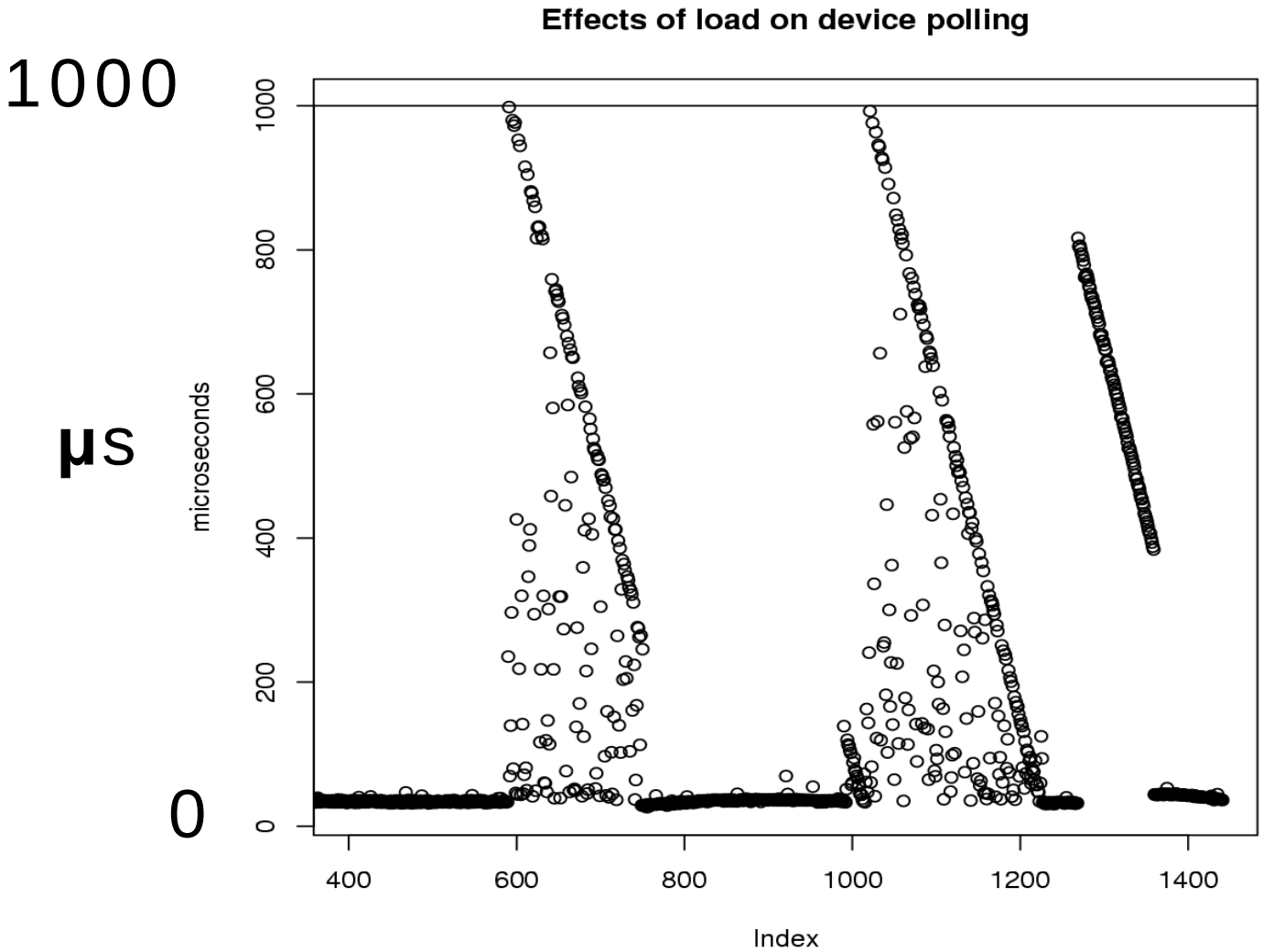


	Med.	*Mean*	*IQR*	*1-99*	*sdev*
Interrupts µs	100	99.9	62	124	36.33
Device polling µs	76.93	78.74	4.95	23.43	29.68

Device polling & system load

- Good results because of idle-loop polling
- What happens under load?

Device polling & system load

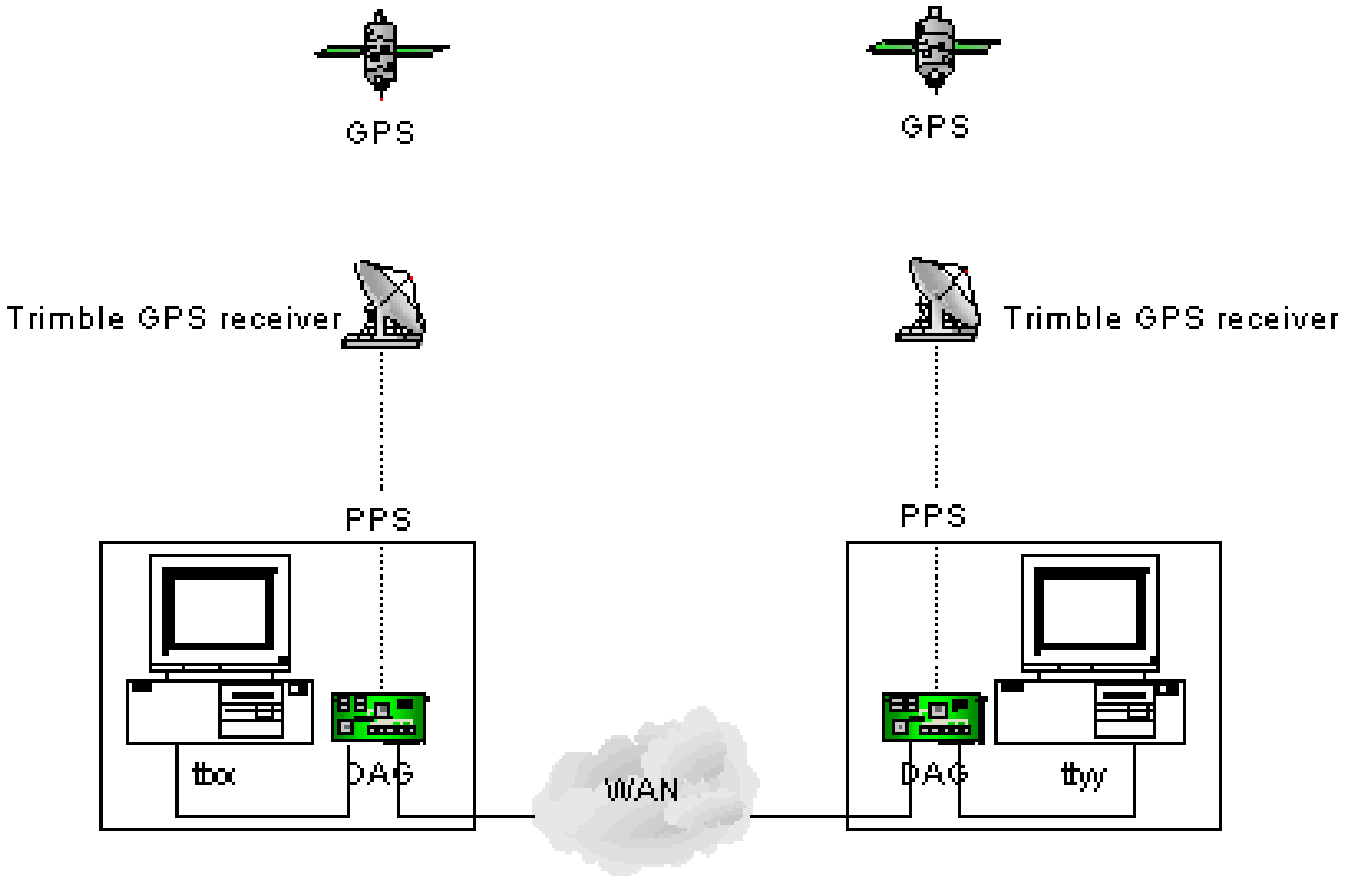


Device polling - conclusions

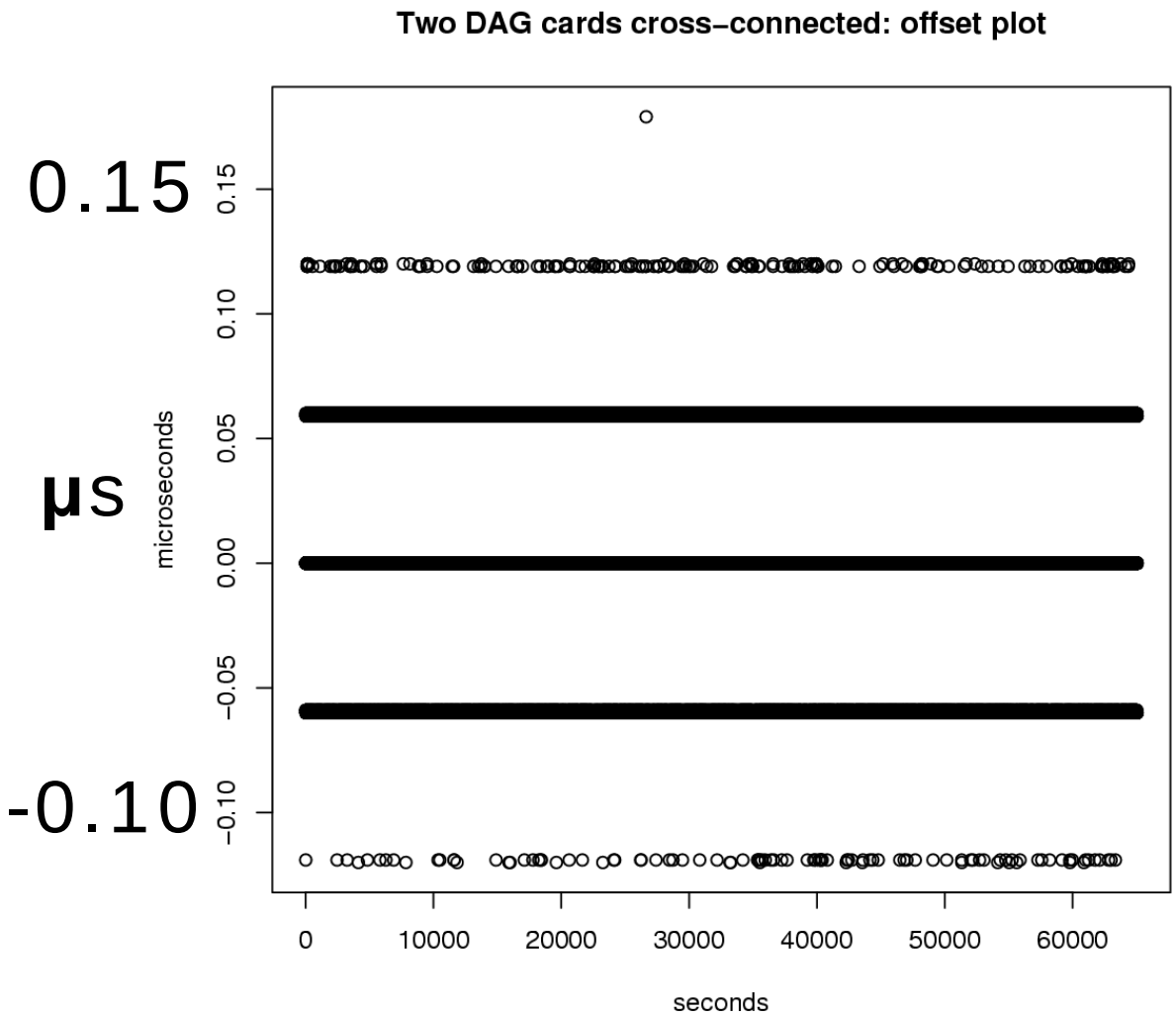
- Theoretically very good results
- Can’t be used in TTM because of load behaviour

DAG-TTM

- Hardware-timestamping solution



DAG TTM – direct connection

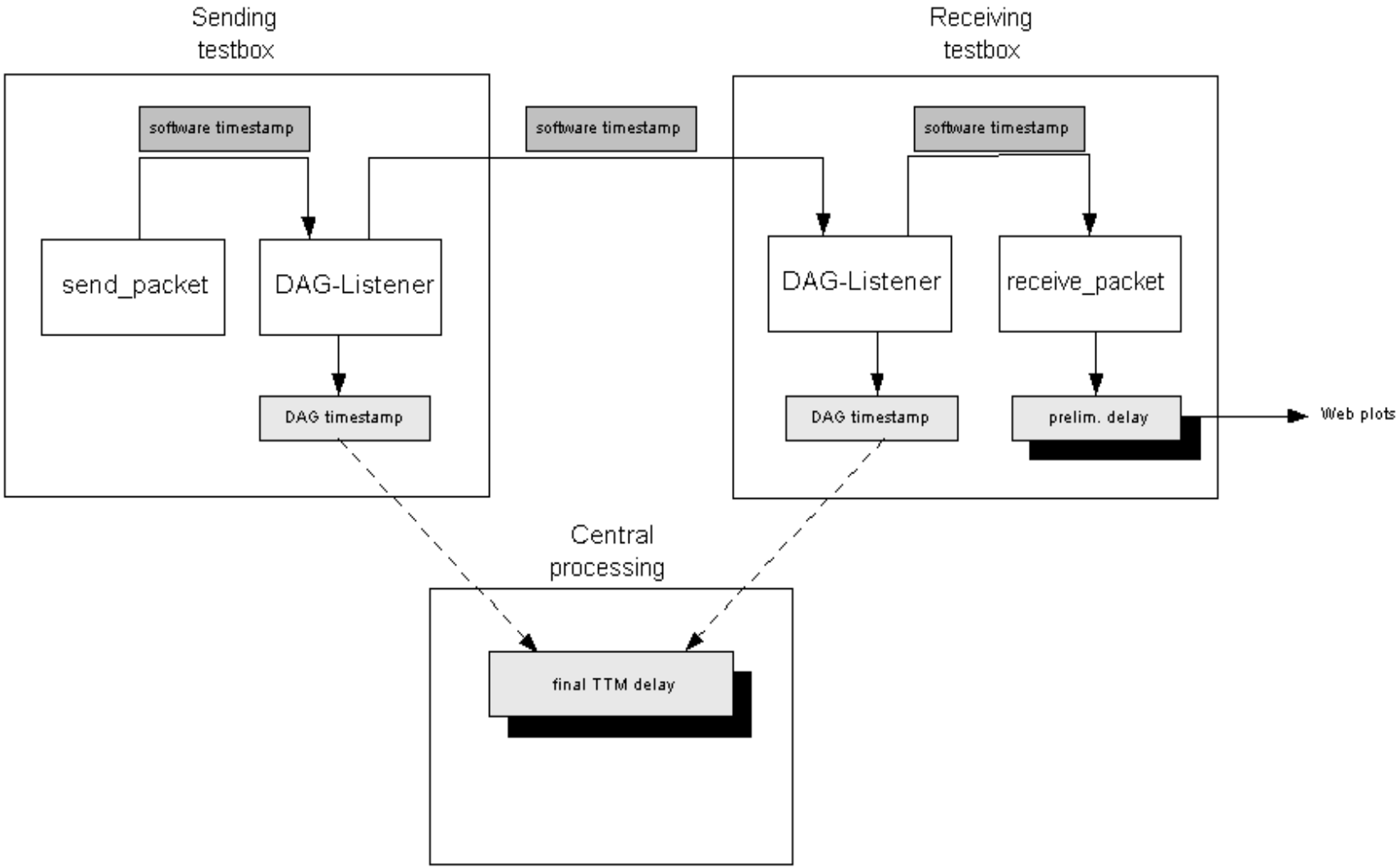


Med.	0.00
Mean	0.0038
IQR	0.00
1-99	0.12
stddev	0.036

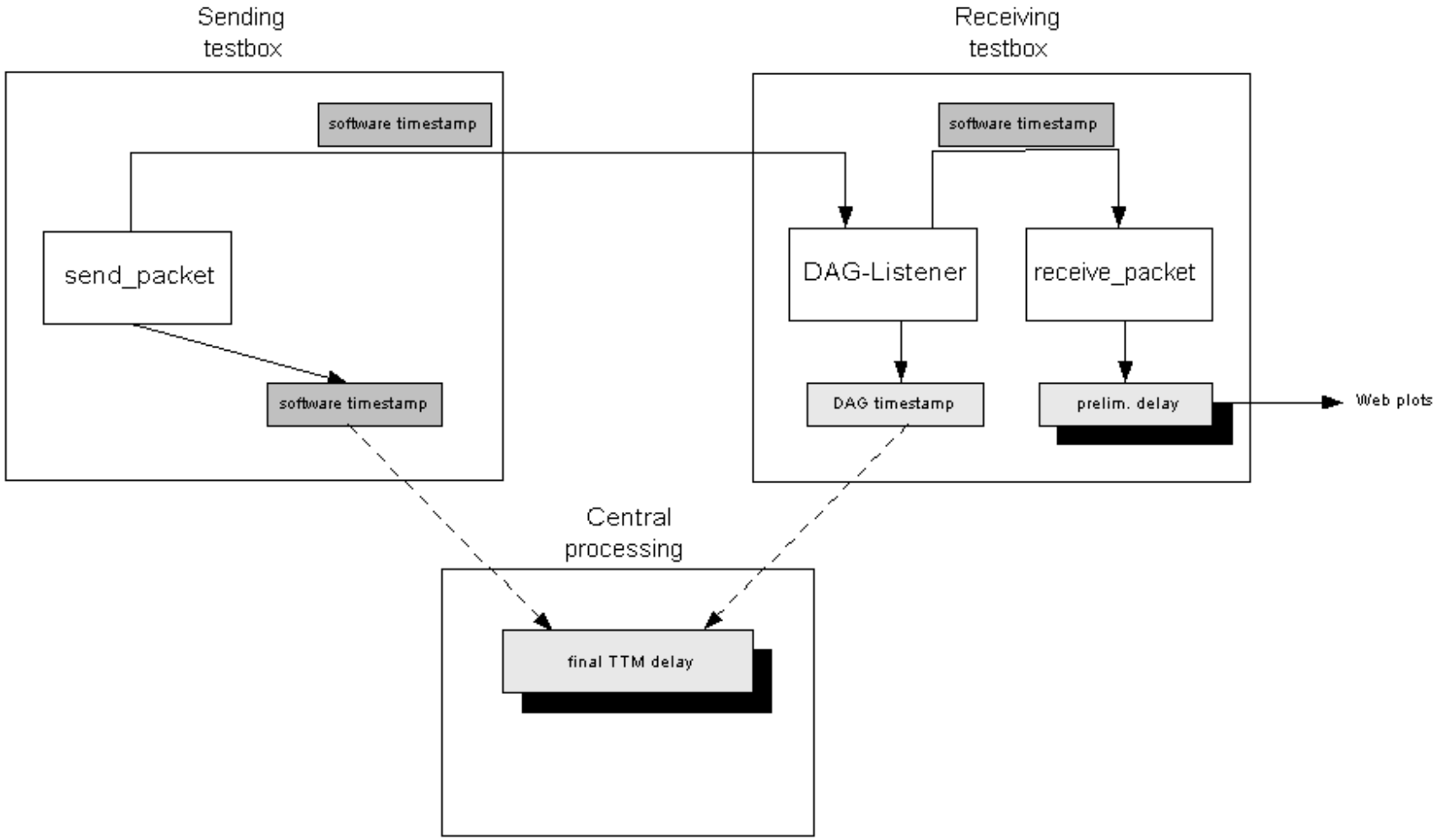
DAG TTM – Integration into TTM

- Backwards compatibility
- Having a DAG card in a testbox is *optional*, not *mandatory*!
- Biggest change: calculate final delays centrally instead of at the receiver
- Use preliminary delays for plots on the testboxes

DAG TTM – Integration into TTM



DAG TTM – Integration into TTM



DAG TTM - Conclusions

- Very good results: no host processing effects at all
- Needs some work
- Price tag

Summary

- NTP ATOM driver: good replacement
- Kernel level timestamping: improves sender
- Device polling: can’t be used in TTM
- DAG-TTM: very good results, needs some work, perhaps student project ☺

Questions?