Contiki NTP Client

Josef Lusticky

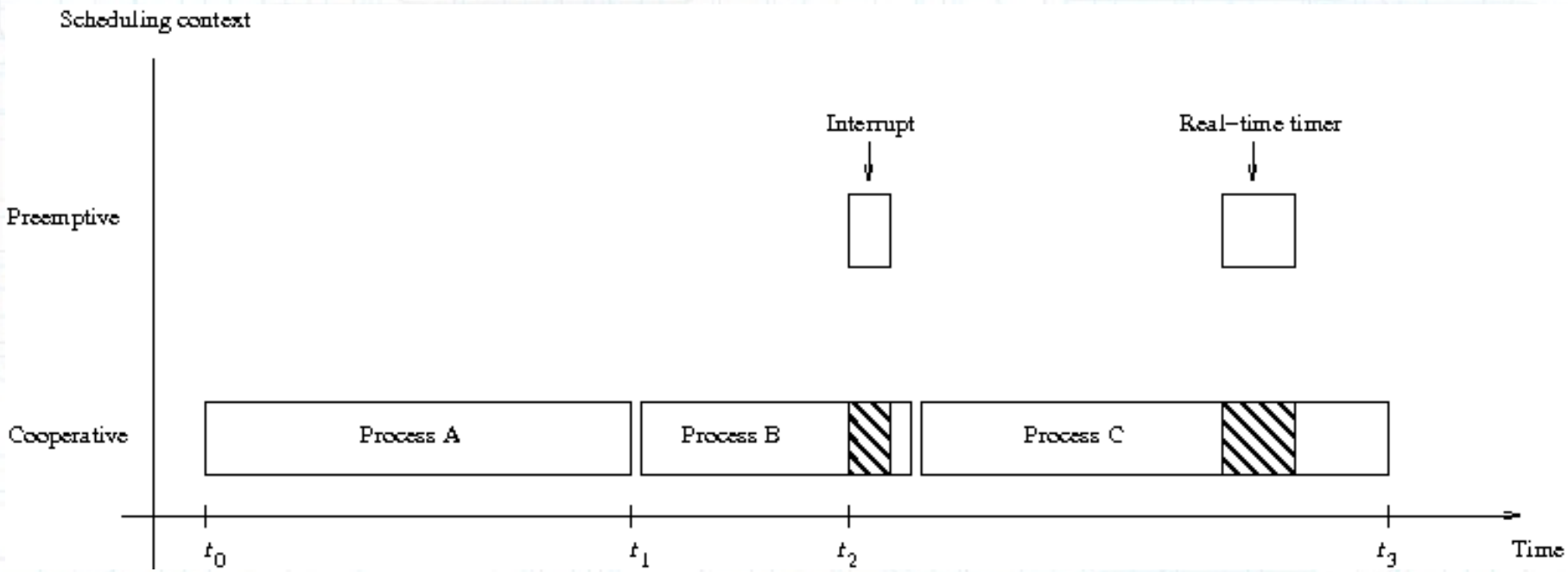
2012

Inhalt

- Contiki OS
- Network Time Protocol
- Analyse
- Design & Implementation
- Messungen

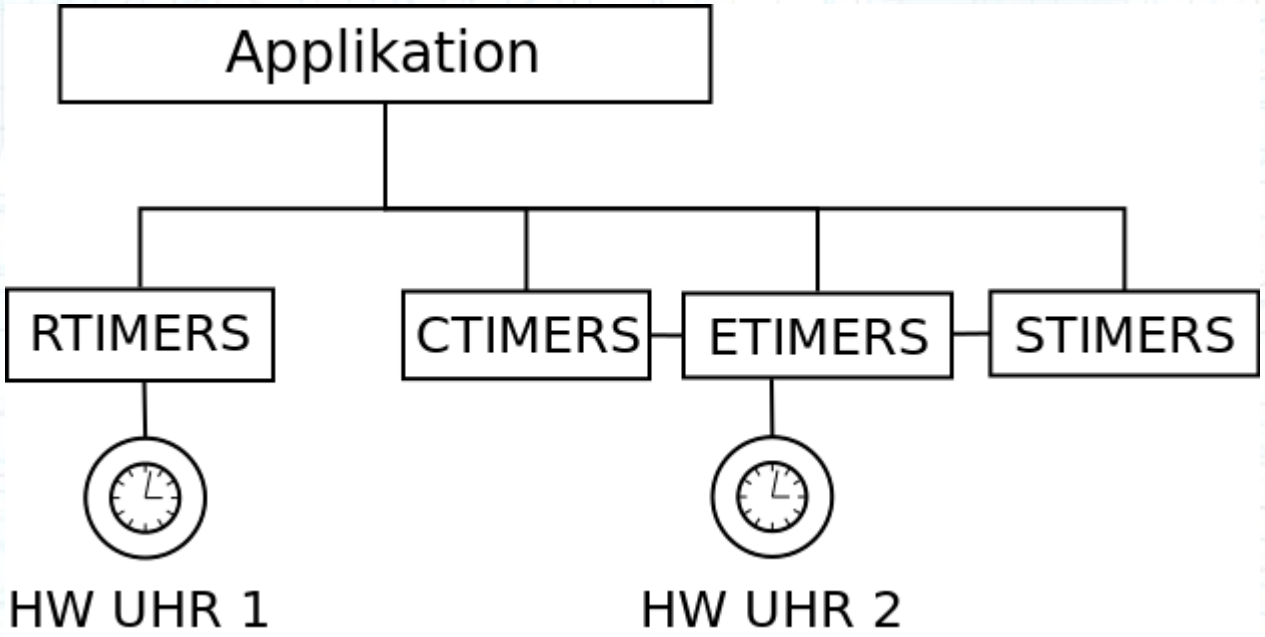
Contiki

- Adam Dunkels – SICS – Atmel,CISCO,...
- 1.0 version aus 2002, aktuell 2.5
- uIP stack – IP v6 und v4
- Kooperatives Multitasking vs. Rtimers, INT



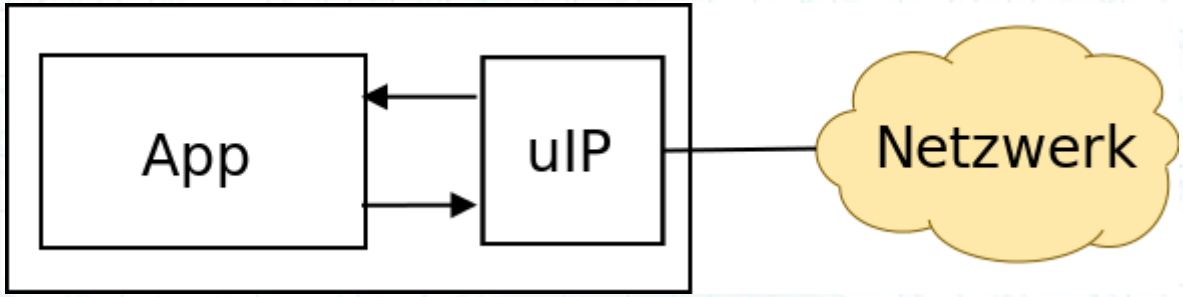
Contiki - Timers

- Real-Time timers (rtimers)
- Andere Timers (event, callback, sekunde)



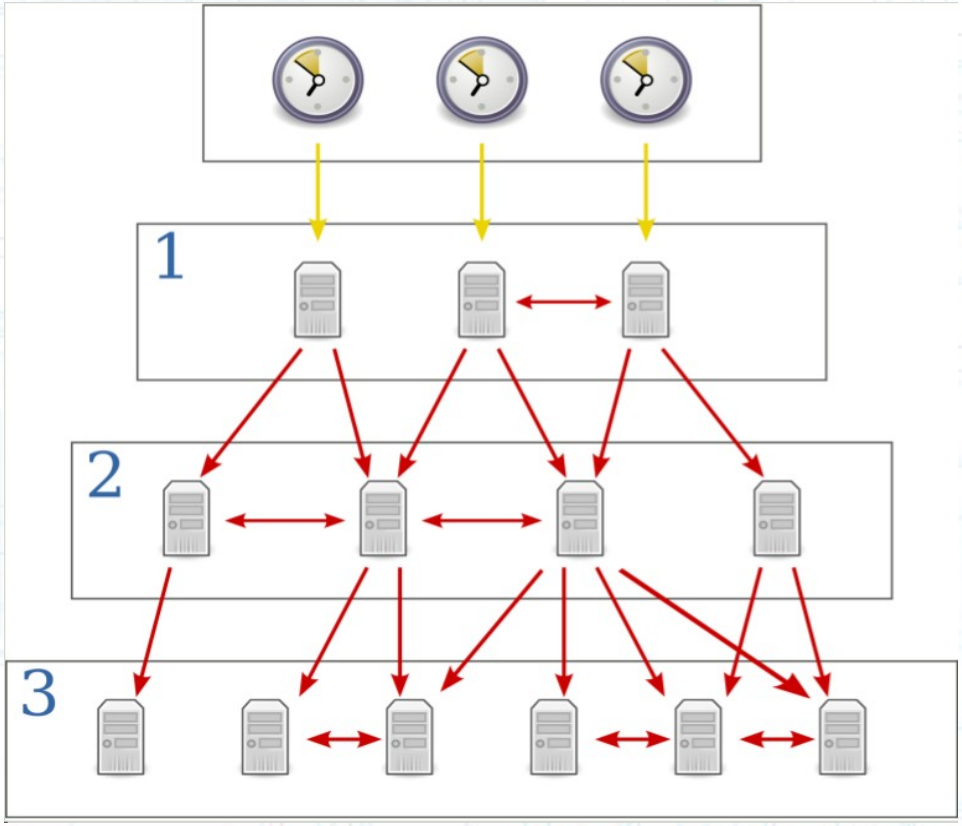
Contiki - uIP

- ~4KB code, ~200B ram - UDP,TCP,ICMP
- Single global buffer, Paketverlust
- API - events



NTP - Network Time Protocol

- David Mills – Uni Delaware 1982
- RFC aus 1985, NTP version 4 aus 2010
- Simple NTP (SNTP) – Teilmenge NTP

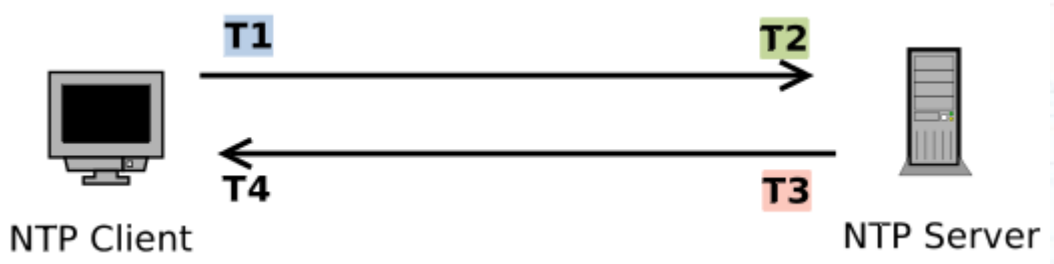


NTP - Kommunikation

- NTP Broadcast mode

$$\theta = t_3 - t_4$$

- NTP Unicast mode



$$\theta = \frac{1}{2}[(t_2 - t_1) - (t_4 - t_3)]$$

NTP – Timestamp

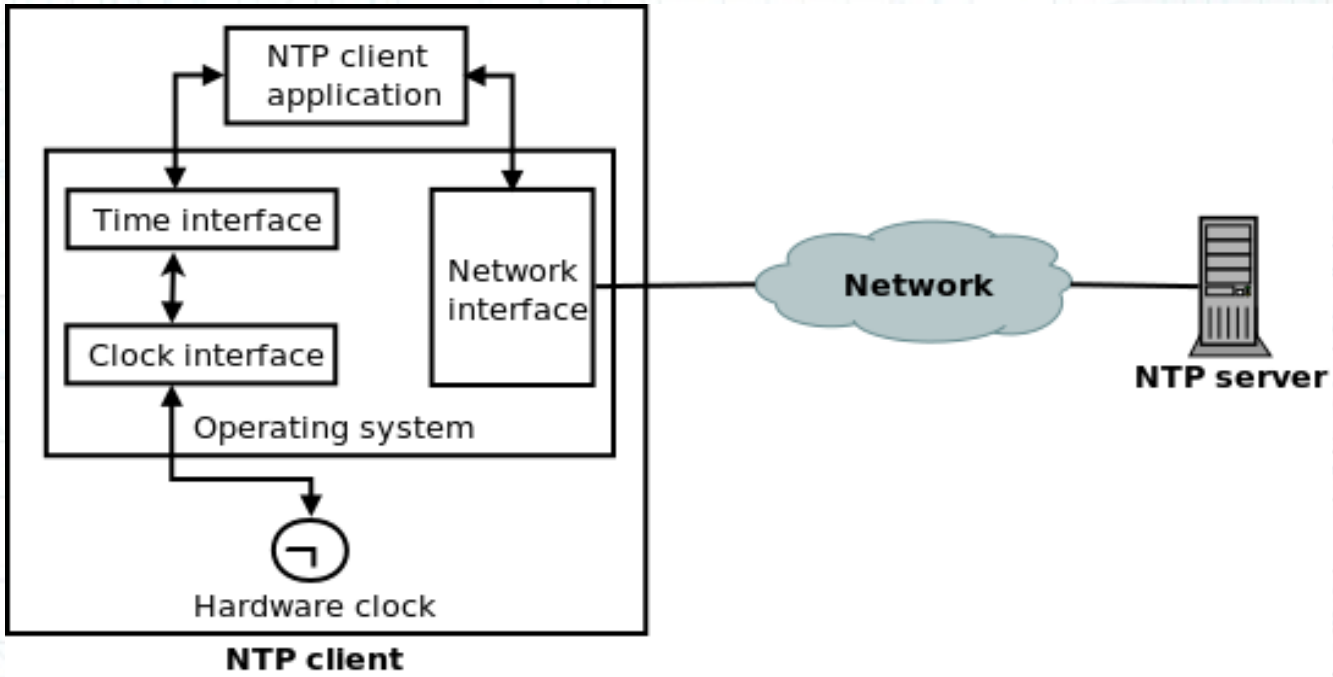
- 64 bits

SEKUNDEN SEIT 1900 BIS 2036	FRAKTION 2^{-32} AUFLÖSUNG 0.23ns

0 31 32 63

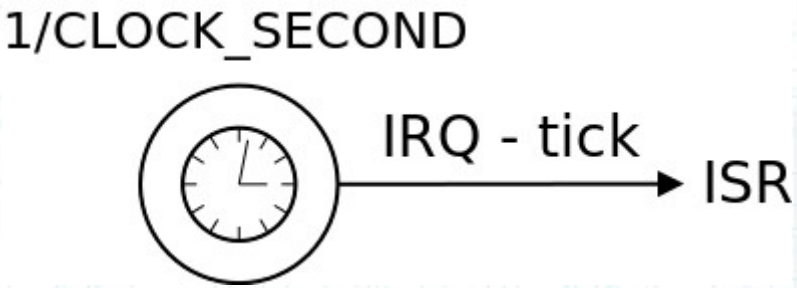
Analyse

- NTP klient - applikation



Contiki Clock Interface

- CLOCK_SECOND – INT/s
- clock_init()
- scount, seconds = 0
- ISR(hwclock)




```
{
scount++;          // timers
if ((scount % CLOCK_SECOND) == 0)
   seconds++;   // stimers
}
```

Contiki Time Interface

- seconds – uptime
- kein settime, gettime oder adjtime

- keine existierte Funktionalität ändern - timers
- höchste Präzision und Auflösung
- kleinstes Speicher Overhead

Design

- Time interface – settime, gettime, adjtime
- NTP klient
- AVR Raven (8-bit MCU)

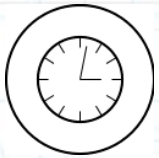
Time interface

- settime(unsigned long sec) {

  boottime = sec − seconds; } // stimers
- gettime(struct timespec *ts) {

  ts->sec = boottime + seconds;

  ts->nsec = scount // 1/CLOCK_SECOND

  + HW Uhr  ; }

- Auflösung auf AVR Raven 244 µsec

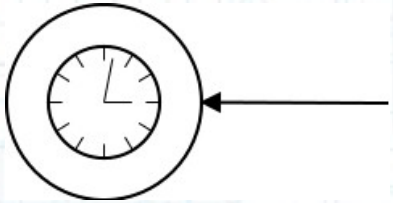
  (7,8 ms ohne Lesen der HW Uhr)

Design – Adjtime

- adjtime(struct timespec *ts) {

  <u>adjcompare</u> = ts->sec * 10^9 + ts->nsec; }

- ISR(hwclock) {

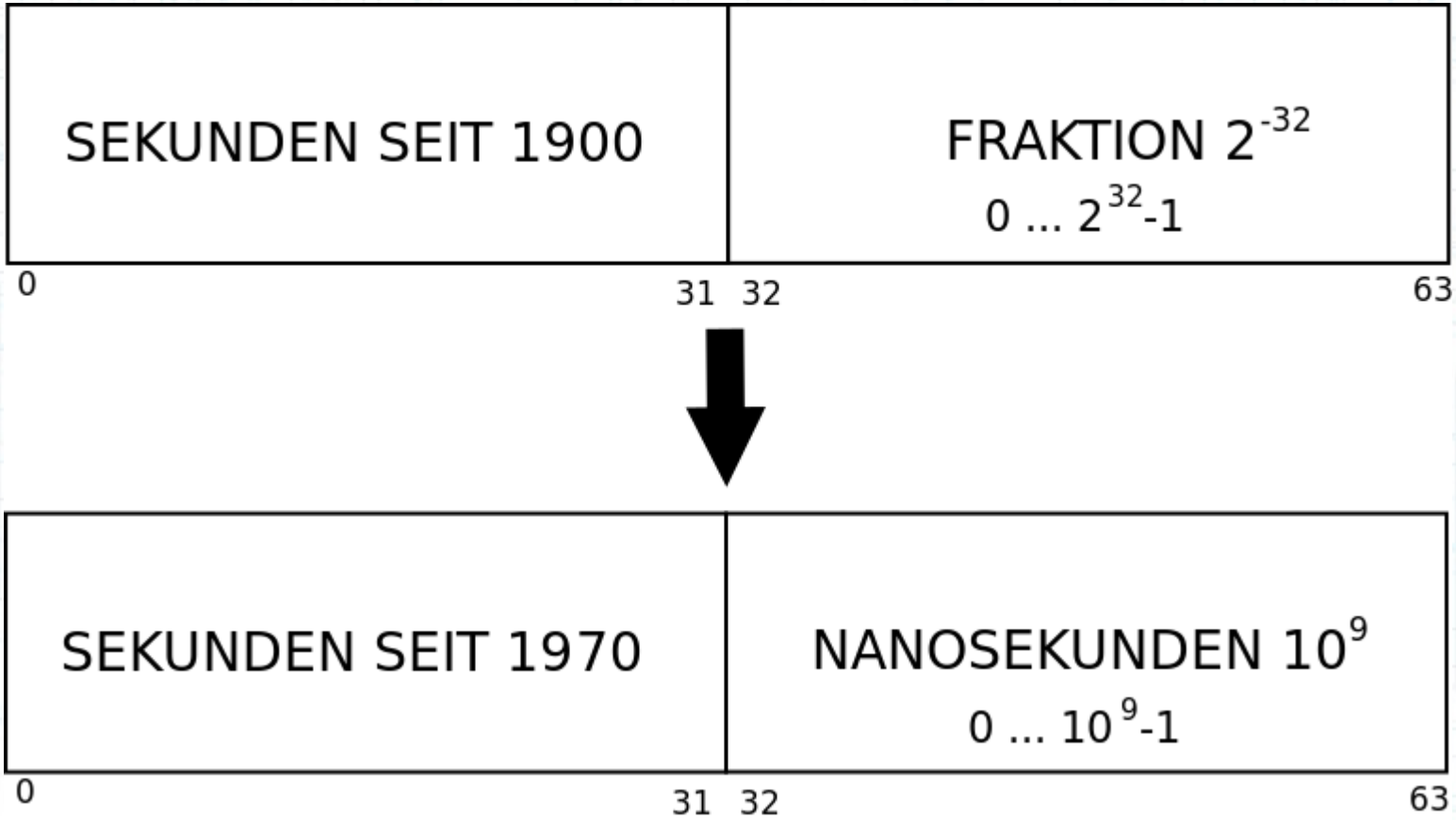
  

  <u>adjcompare</u>; } //timers beienfl.

- Adj_{MIN} = Auflösung = 244 µsec
- Adj_{MAX} = 0,03 s/s

Design – NTP klient

- App. Rechnet Local clock offset
- Timestamp Koversion



Design - Konversion

- Std C: uint32 * uint32 = uint32
- FP: (double)ntp * 10^9 / 2^32; // fp: 3.5KB
- 64bit: (uint64)ntp * 10^9 >> 32; // 0.7KB

- ntp * (5^9*2^9) / 2^32 = ntp * 5^9 / 2^23
- Std. C: x * 2^e = x << e;

  x / 2^e = x >> e;
- Mathe: 5*x = 4*x + x;

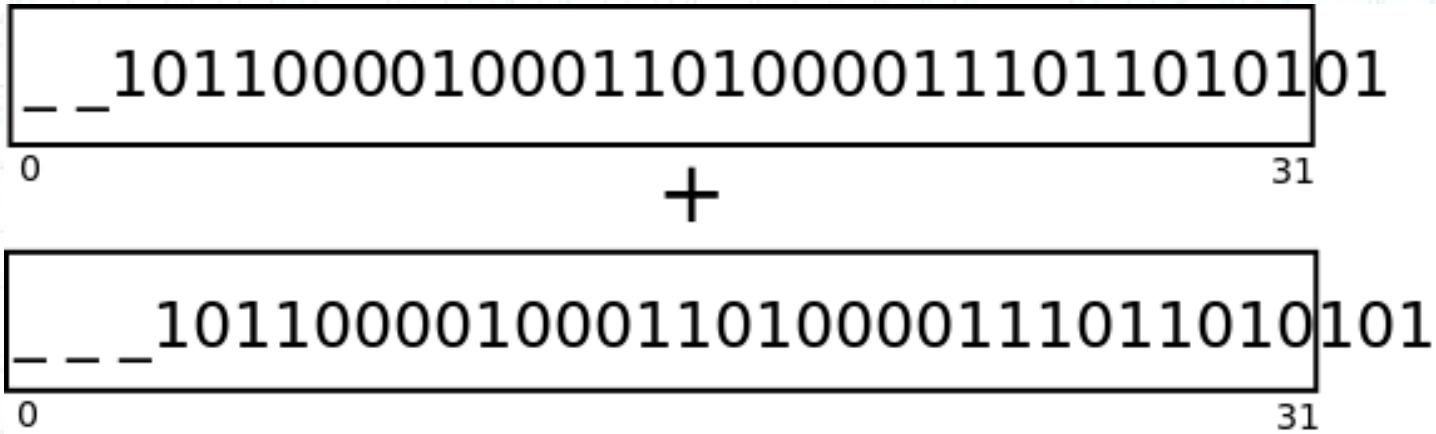
Implementation – Konversion

- nsec = (5*ntp) / 8 = ntp / 2 + ntp / 8

  ntp >> 1 + ntp >> 3;

- nsec = 3*ntp + ntp/8 = (25*ntp) / 8

  ntp << 1 + ntp + ntp >> 3;

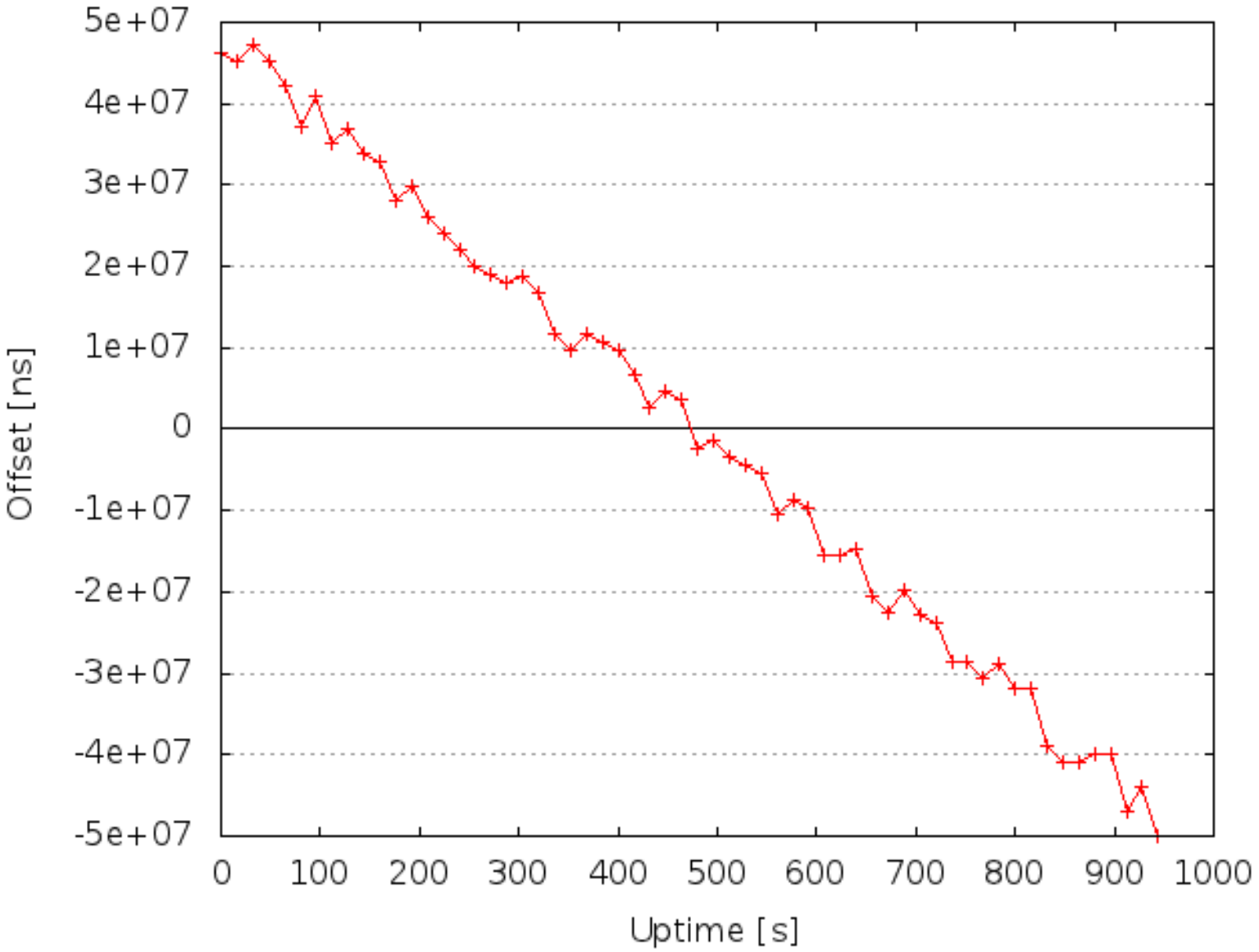


- grösste Differenz 5ns

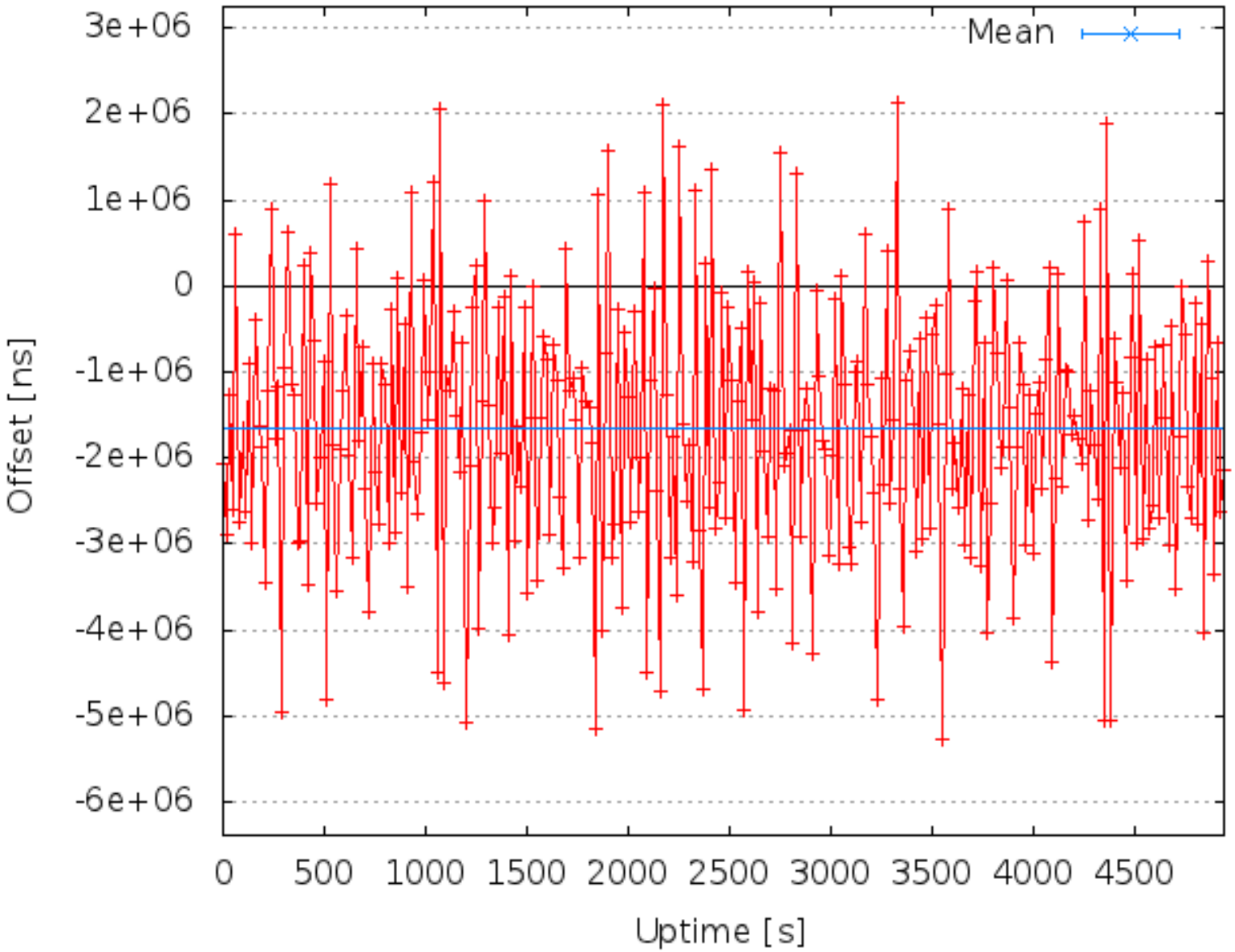
Implementation – Klient

```
send_packet();
set_event_timer(16s);
for (;;) {
   WAIT_EVENT();   // kein Blockierung
   if (event == tcpip)  process_packet();
   else if (event == etimer) {
        send_packet();
        set_event_timer(16s); }
}
```

Messungen – ohne NTP



Messungen – mit NTP



Contiki NTP Client

Josef Lusticky

2012