Evaluation of Message Delay Correlation in Distributed Systems

Daniel Albeseder

da@ecs.tuwien.ac.at

Embedded Computing Systems Group

TU Wien, 20.05.2005

Overview

- Motivation for the Θ-Model
- Introduction of the Θ-Model
- Evaluation-Setup
- Results
- Conclusion

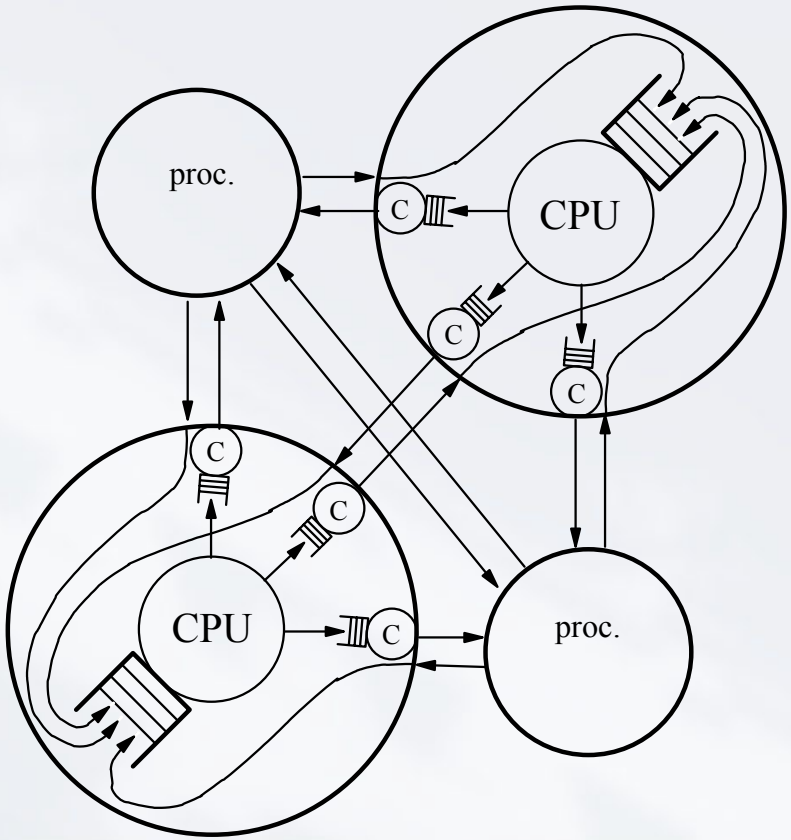
Timed Algorithms

- Most FT algorithms for distributed RTS have explicit time values (unit „seconds“) in their code / variables
- Toy example: Local real-time clock for timing out a crashed process

```
msg_pong  do_roundtrip(msg_ping, p) {
        send msg_ping to p
        TIMEOUT := C(t) + 2τ+   /* max. end-to-end delay τ+ (sec) */
        while C(t) < TIMEOUT do nothing
        if msg_pong did not arrive then msg_pong := NIL
        return msg_pong
}
```

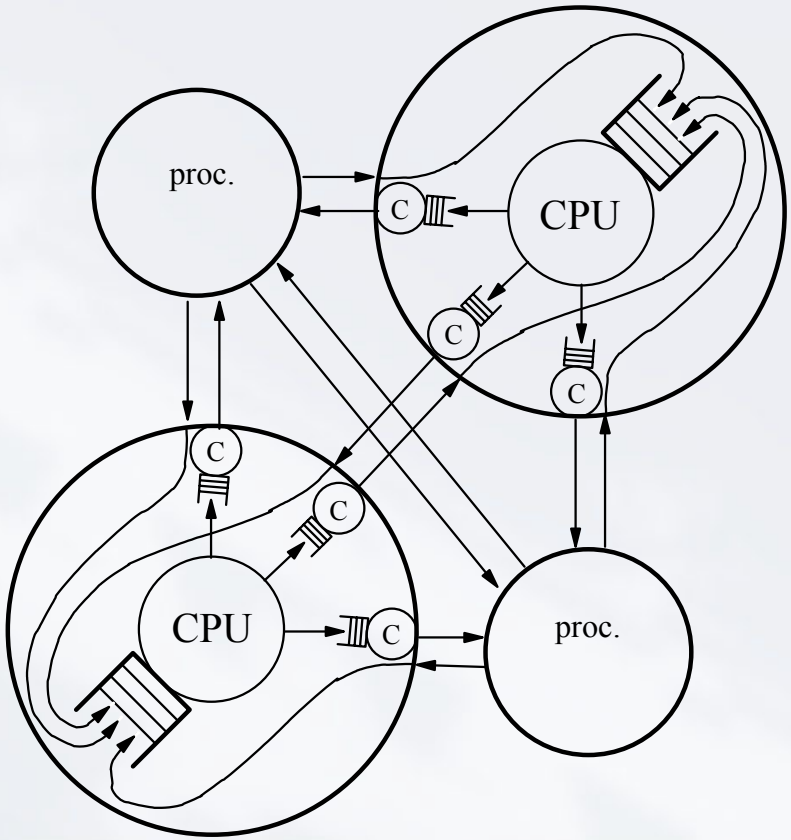
Why is determining τ^+ difficult ?

- Queuing phenomenons:
  - Simultaneous messages from different peers (CPU)
  - Multiple processes (CPU)
  - Multiple messages (Link)

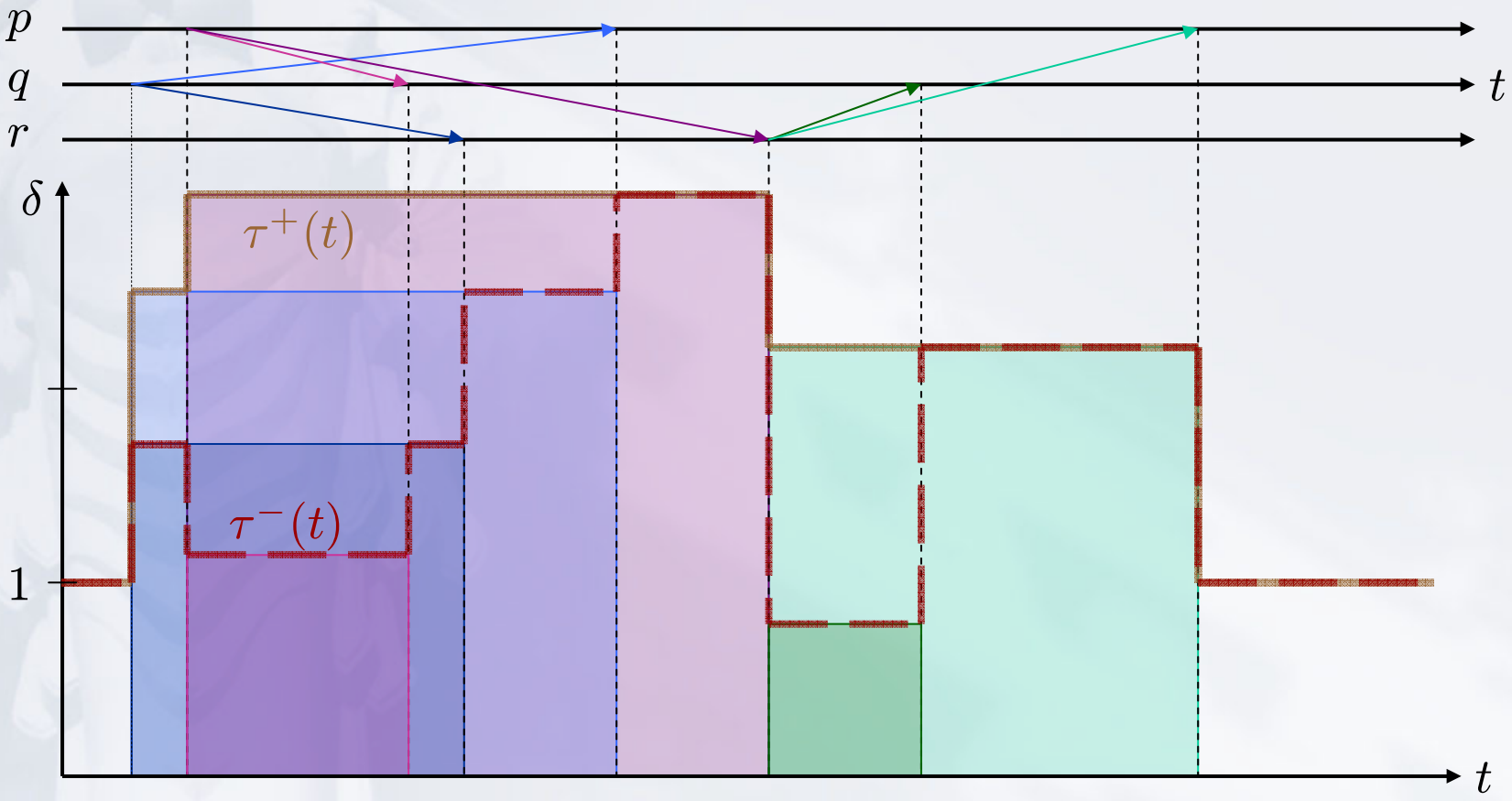


Why is determining τ^+ difficult ?

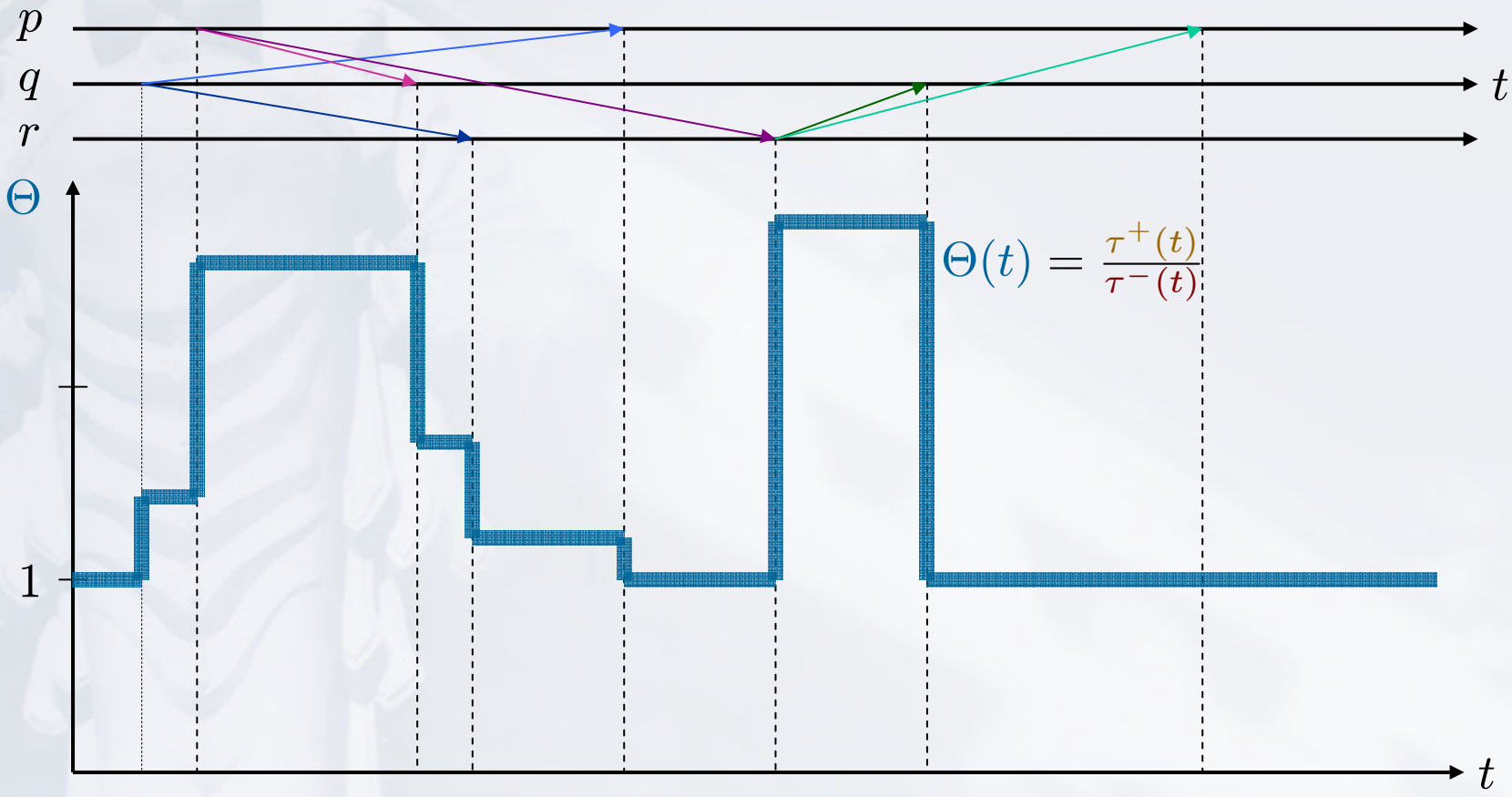
- End-to-end delays hence depend upon
  - message & computational complexity of algorithms
  - interaction („blocking factors“)
  - load conditions
  - scheduling disciplines



The Θ-Model



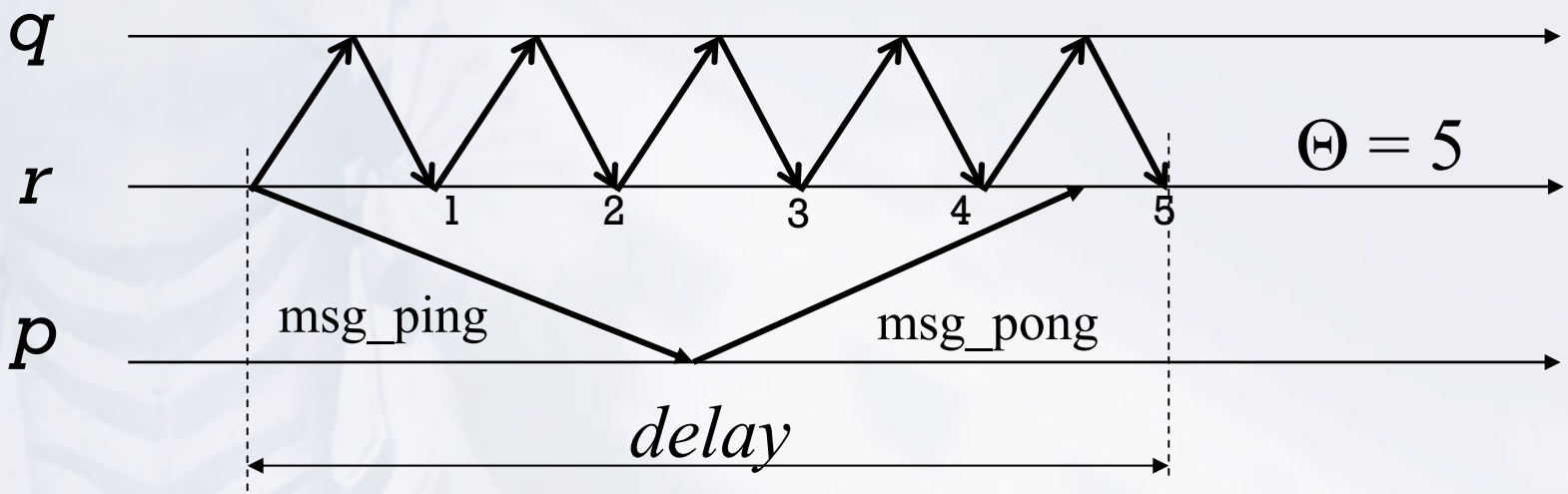
The Θ-Model



Time-free implementation of do_roundtrip(.)

```
send msg_ping to p
for i=1 to Θ do          /* Θ is dimensionless ! */
begin                    /* do additional roundtrips for waiting */
   send delay_ping(i) to process q
   wait for delay_pong(i) from process q
end
if msg_pong did not arrive then msg_pong := NIL
return msg_pong
```

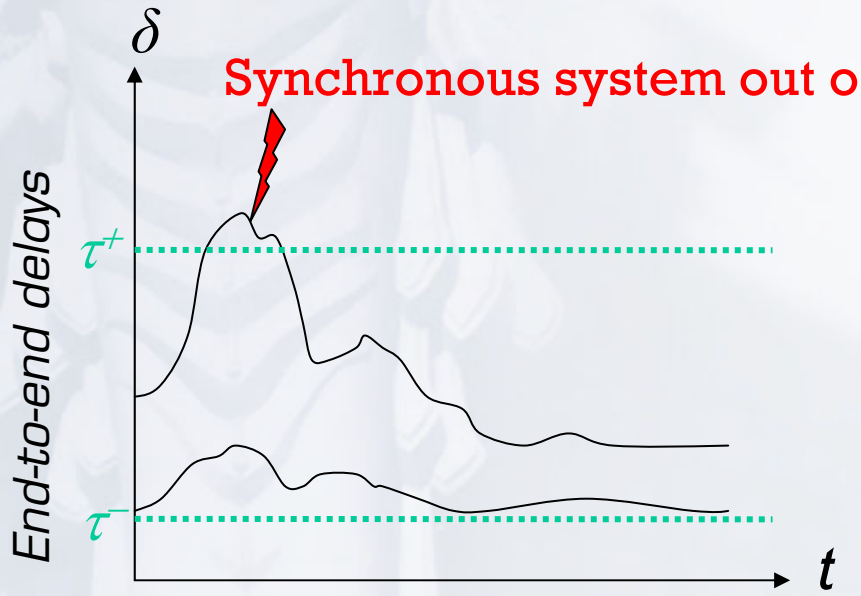
Time-free implementation of do_roundtrip(.)



- Timing behavior solely emerges from the underlying system [*delay* adapts automatically to actual speed]

Correlation → Coverage Expansion

- Given some bound τ^+ and τ^- assumed during system design, compute $\Theta = \tau^+ / \tau^-$
- Unanticipated overload: $\tau^+(t) > \tau^+$



if $\tau^+(t) \leq \Theta\tau^-(t)$
however,
Θ-system still OK

Goals

- Is there a correlation between maximum and minimun end-to-end delay?
- Is the minimum end-to-end delay increasing for higher load settings?
- How big is Θ in a real system?

Evaluation

Clock Synchronization Algorithm

```
VAR k : integer := 1;

send (init,1) to all [once];                /* initialization */

if received (init,j) from at least f+1 distinct processes with j >= k
    k:= j;                                  /* catch-up with new round */
    send (init,k) to all [once];
fi

if received (init,k) from at least n-f distinct processes
    delay(D);
    k := k + 1;                             /* advance to next round */
    send (init,k) to all [once];
fi
```

Significant Messages

- Not all message are significant for Θ-assumption
- Actually only the messages which contribute to a round-switch (RS) are relevant
- Therefore, we define:

$\tau_r^-(t)$ ... n-f fastest message contributing to a RS

$\Theta = \max_t \frac{\tau^+(t)}{\tau_r^-(t)}$ ... Significant uncertainty ratio

Evaluation Setup

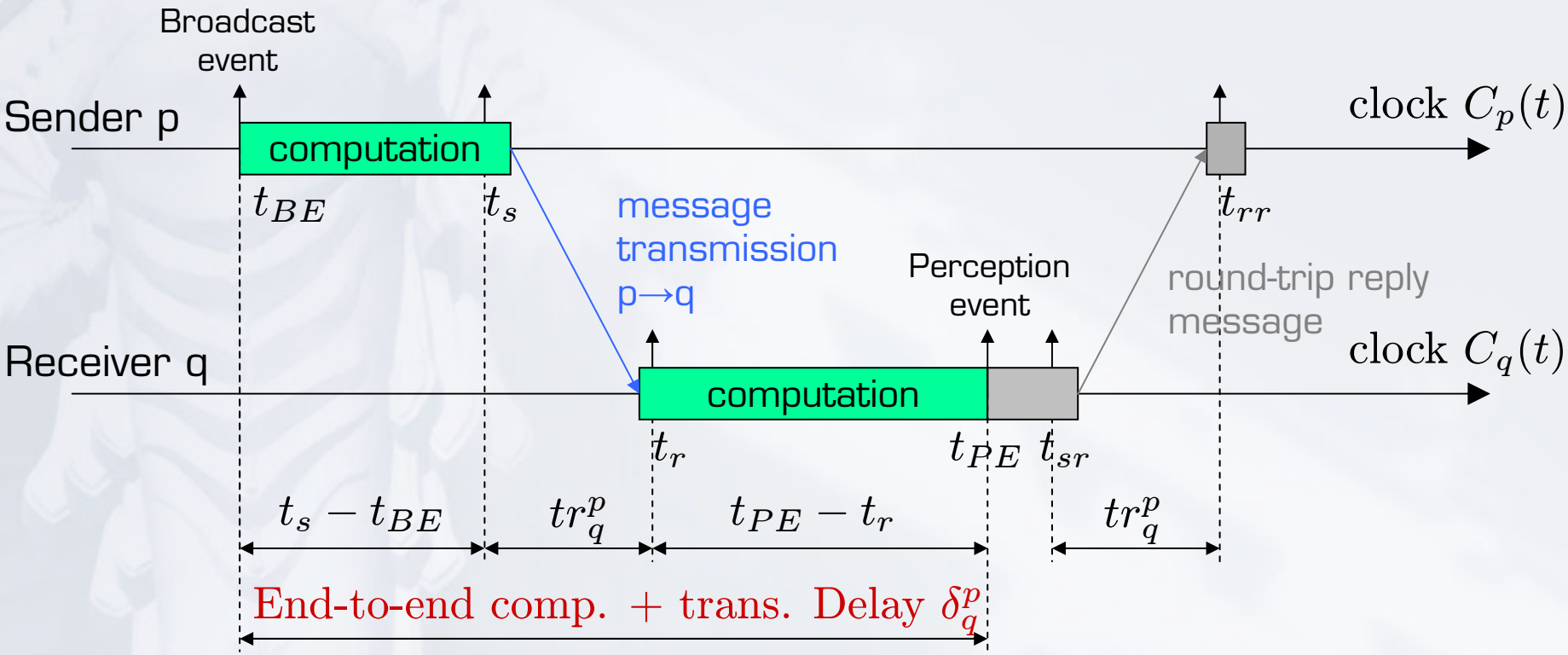
- Pentium4 workstations (2,4GHz FSB533)
- Fully switched Fast-Ethernet over two Cisco Catalyst 2950 switches (connected over fiber Gigabit-Ethernet backbone)
- Red Hat Linux 7.2 with 2.4.20 kernel, patched with High-Resolution-Timers and Kernel-Preemption

Measurements

- Number of processors n=4 and n=7
- Failure settings f=1 and f=2 (without failure injection)
- Interround-Delays D=1ms, D=100µs and omitted
- Various load cases:
  - Constant load,
  - increasing load and
  - load jumps

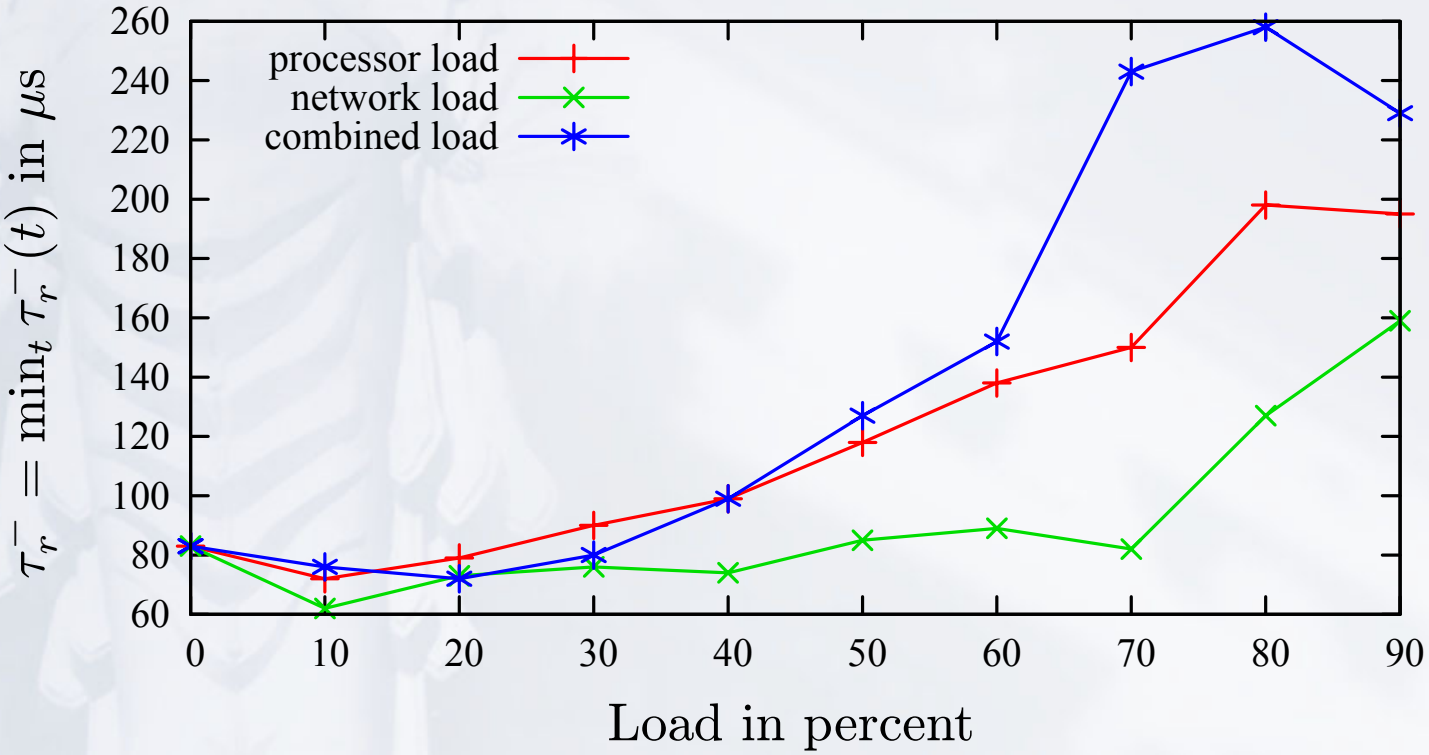
} 1. Processor load
2. Network load
3. Combined load

Round-Trip Measurement



Results

Minimal End-To-End Delay

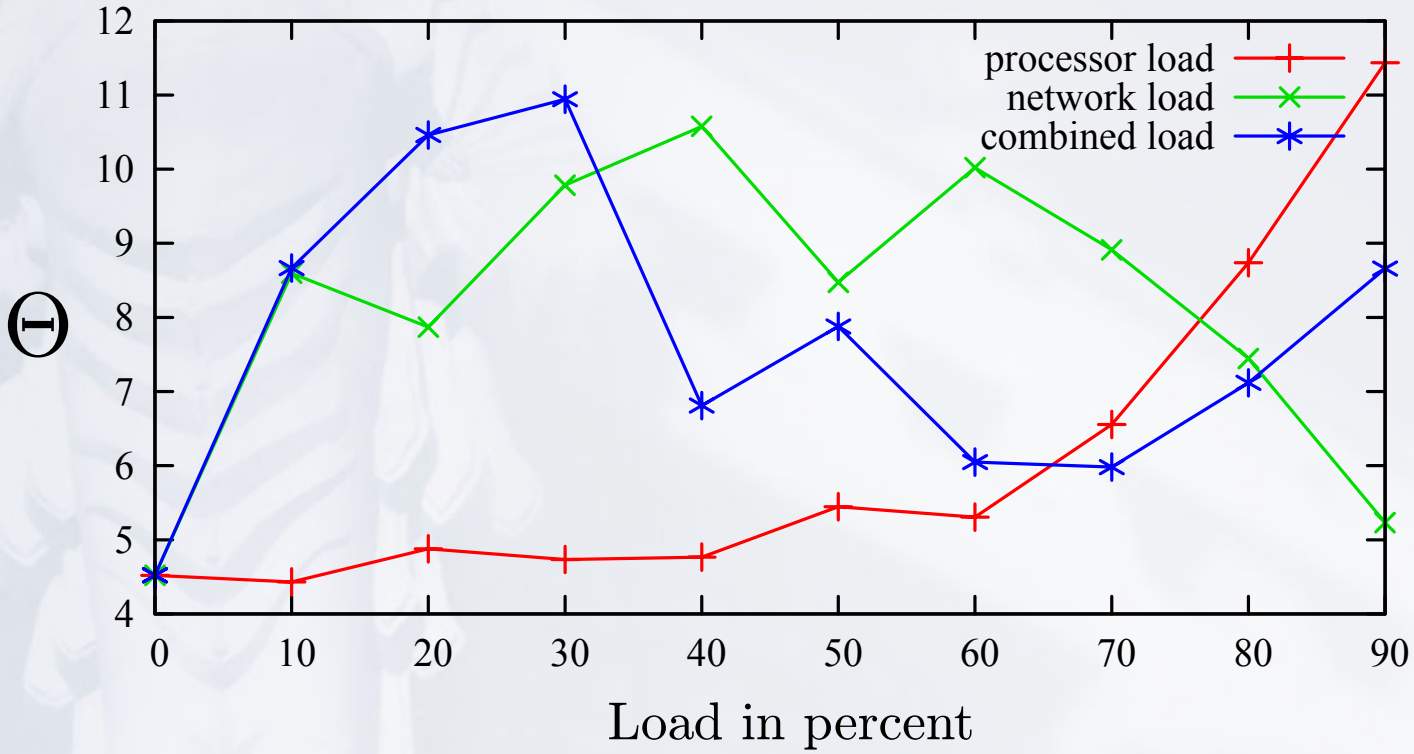


$n = 7$

$f = 2$

$D = 1\text{ms}$

Uncertainty Ratio Θ

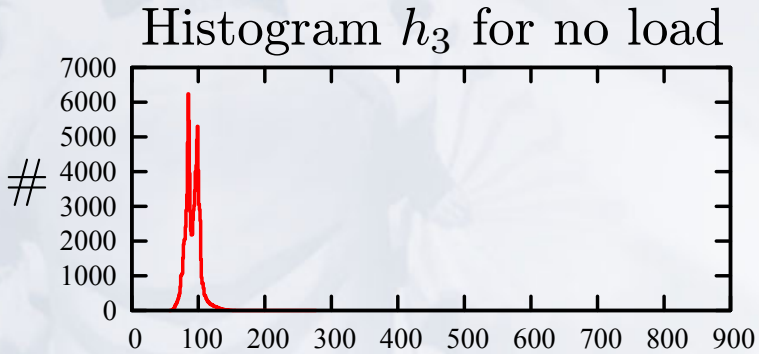


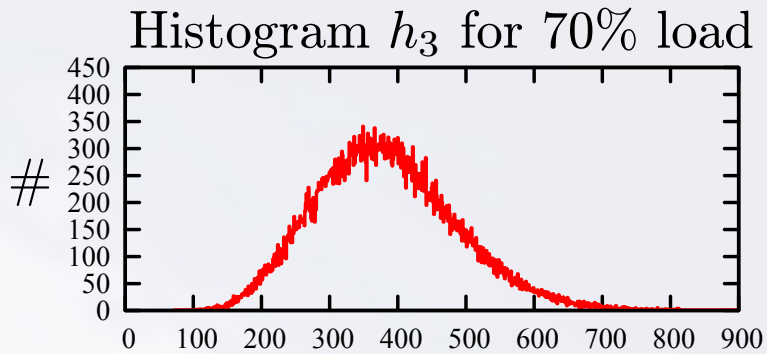
$n = 7$

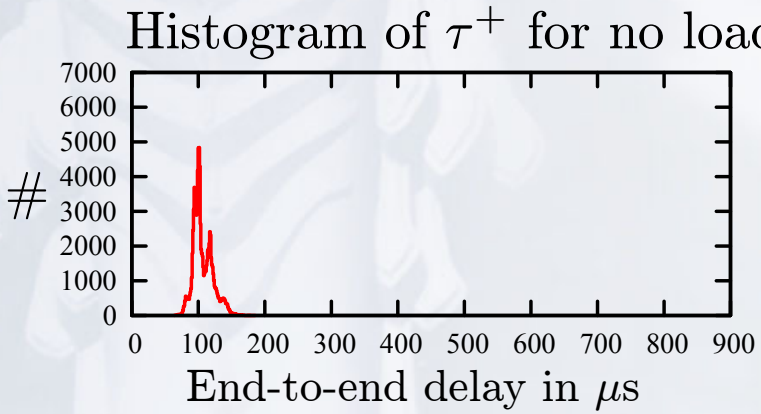
$f = 2$

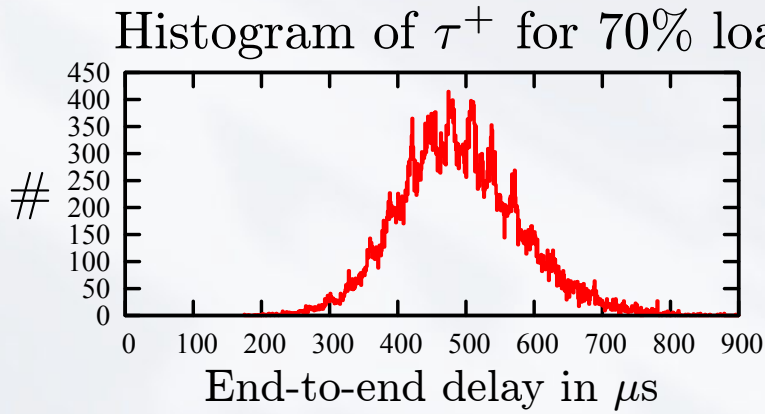
$D = 1\text{m}7\text{s}$

Histograms

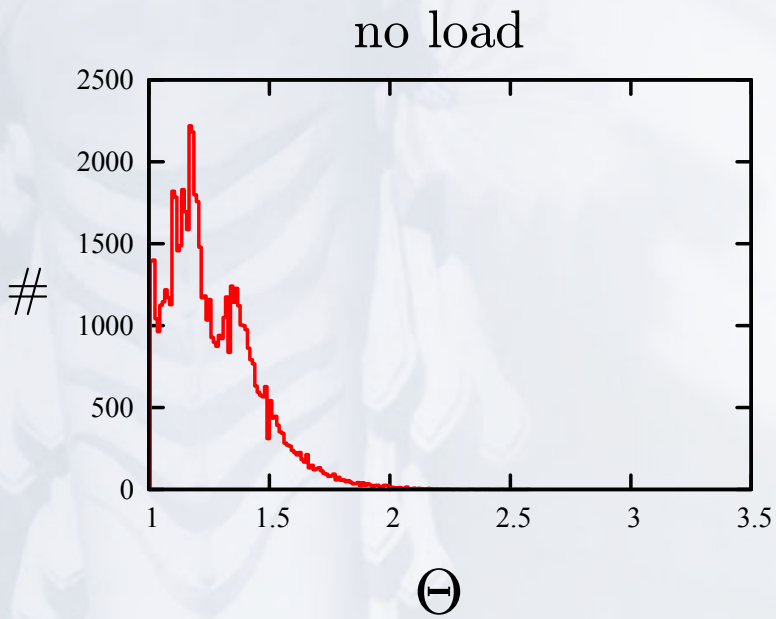

Histogram h_3 for no load
#
0
1000
2000
3000
4000
5000
6000
7000
0
100
200
300
400
500
600
700
800
900

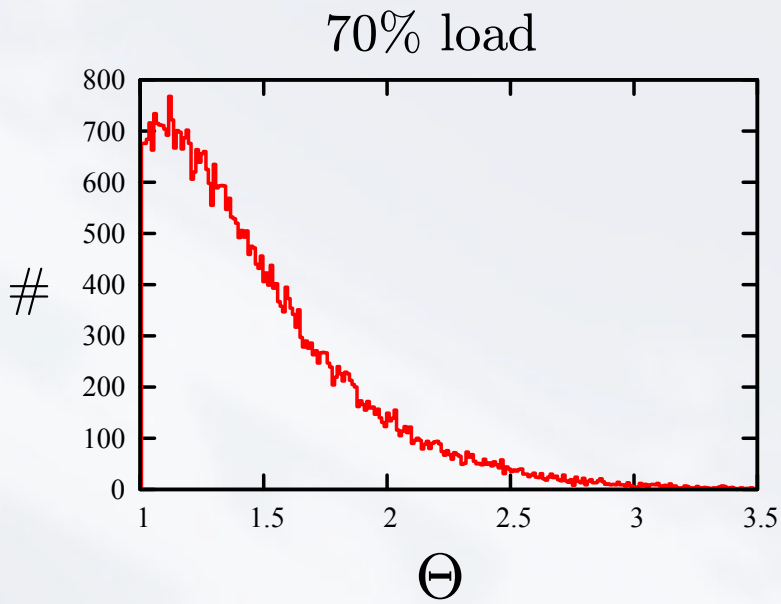

Histogram h_3 for 70% load
#
0
50
100
150
200
250
300
350
400
450
0
100
200
300
400
500
600
700
800
900


Histogram of τ^+ for no load
#
0
1000
2000
3000
4000
5000
6000
7000
0
100
200
300
400
500
600
700
800
900
End-to-end delay in μs


Histogram of τ^+ for 70% load
#
0
50
100
150
200
250
300
350
400
450
0
100
200
300
400
500
600
700
800
900
End-to-end delay in μs

Histograms of Θ>1





Conclusion

- Experimental evaluation of a clock synchronization algorithm
- Common OS (Linux) with enhancements used
- Θ-Model is valid for our assumptions
- This evaluation is only the beginning of a more comprehensive study

Thank you!