On Energy Consumption and Drifting Clocks

and why models and algorithms are needed to assess their interplay

Holger Hermanns

Dependable Systems and Software
Saarland University, Germany

VASY
INRIA Rhône-Alpes, France

Claim (my version of the MLQA mission)

- There is an enormous spectrum of intriguing research questions out there … in embedded system design

- Many of them
  - are crying for mathematically well-founded investigations.
  - are very tough to investigate with current technology.

- Often, there is no other way.

- So, let's invest in better modelling and analysis techniques for them.

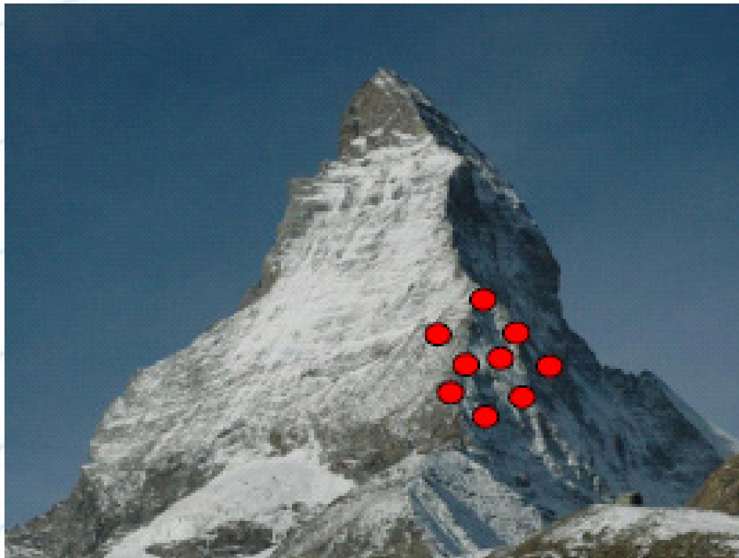
How the practical guys look at the problems

Remember the time when "compilers" didn't yet have error/warning messages?

Jan Beutel, ETHZ

PermaSense project @ Matterhorn

and many others

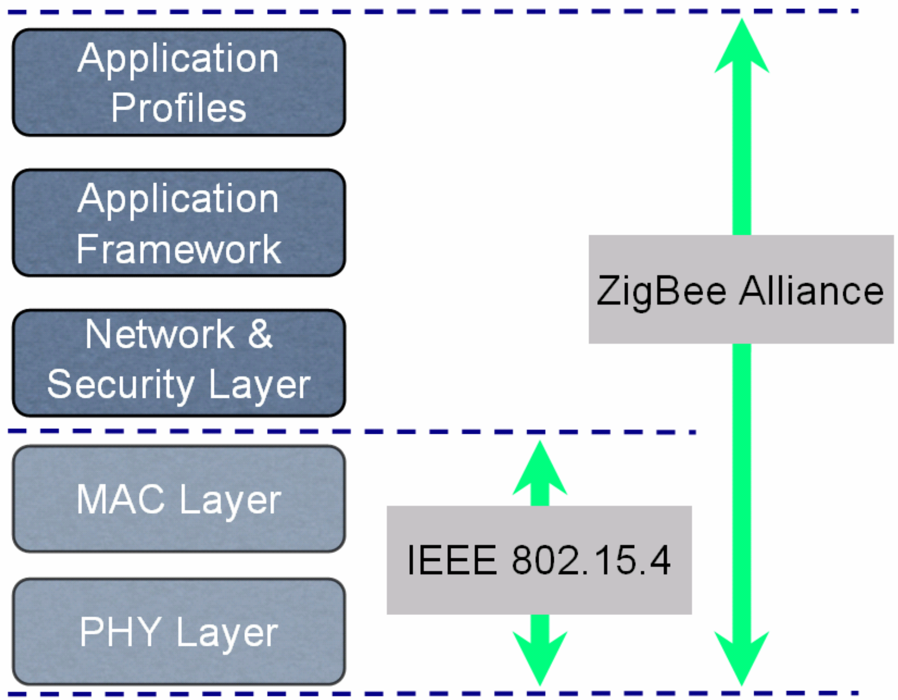


Example

Example: ZigBee

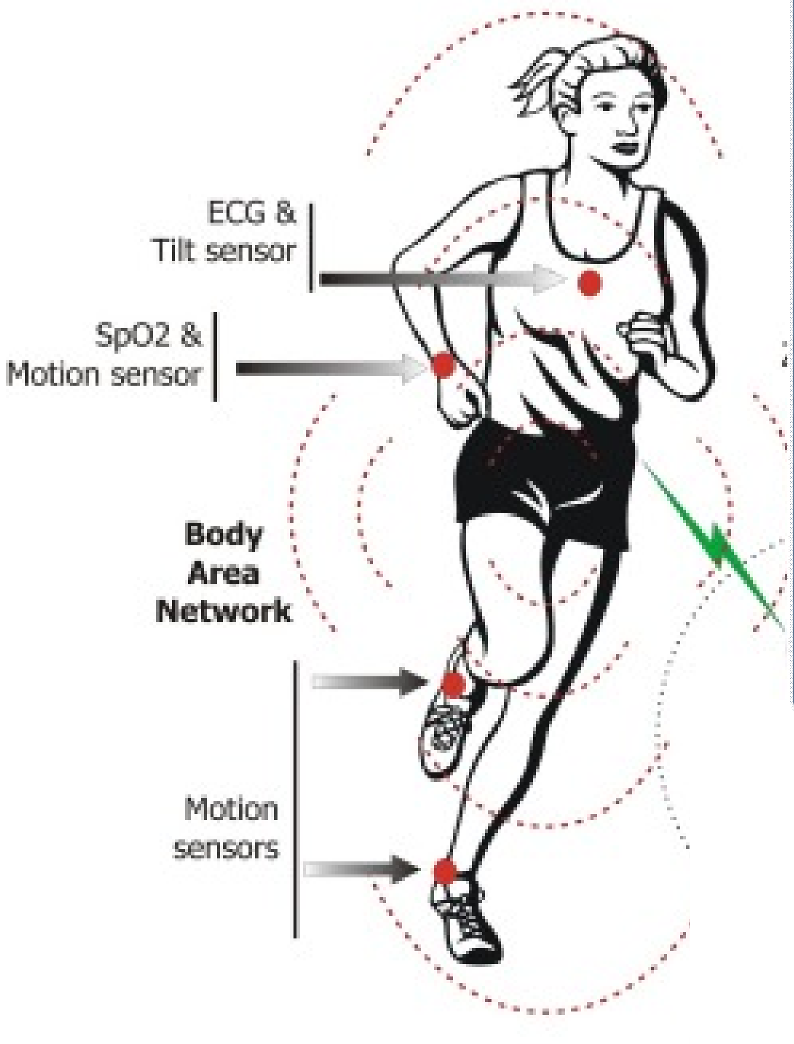
Sensors and Actors

- Low-Rate Wireless Personal Area Network
- 250, 40, or 20 kbit/s data rate



Holger Hermanns

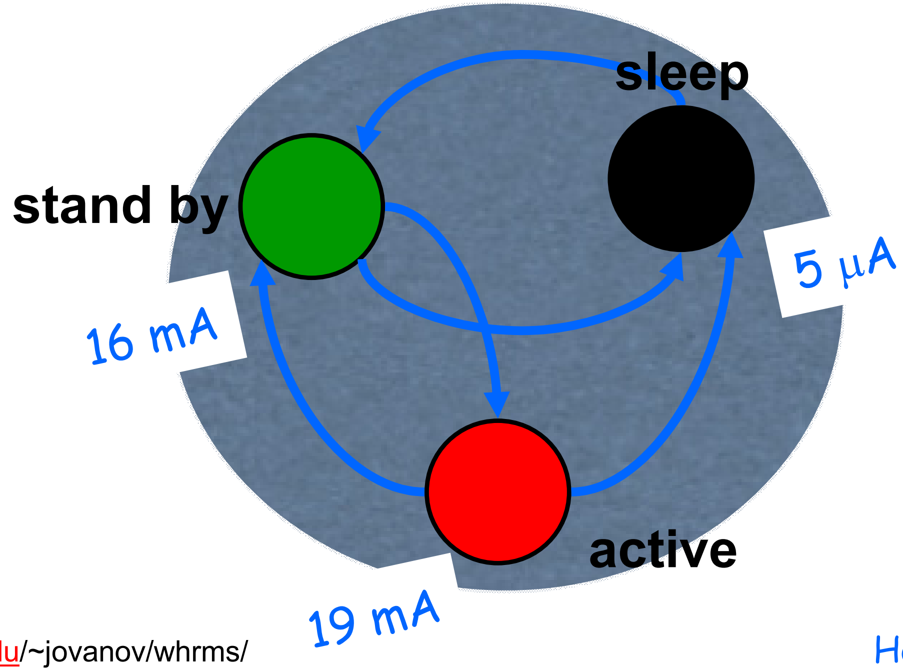
ZigBee



Characteristics:

+ very low power consumption
+ small range
+ simple comm. protocol

- low data rate



source: The University of Alabama in Huntsville www.ece.uah.edu/~jovanov/whrms/

ZigBee MAC Layer: IEEE 802.15.4

Configurable channel access types

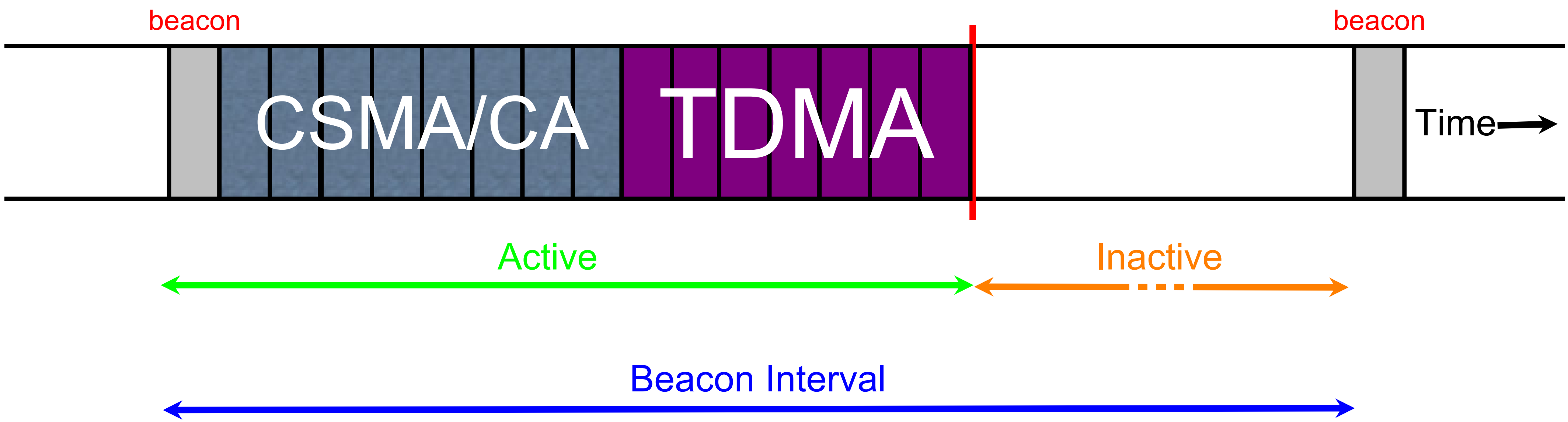
Guaranteed time slots

- TDMA
- Slotted CSMA/CA
- Unslotted CSMA/CA

Beacon determines frame boundaries

Probabilistic medium access

Superframe Structure

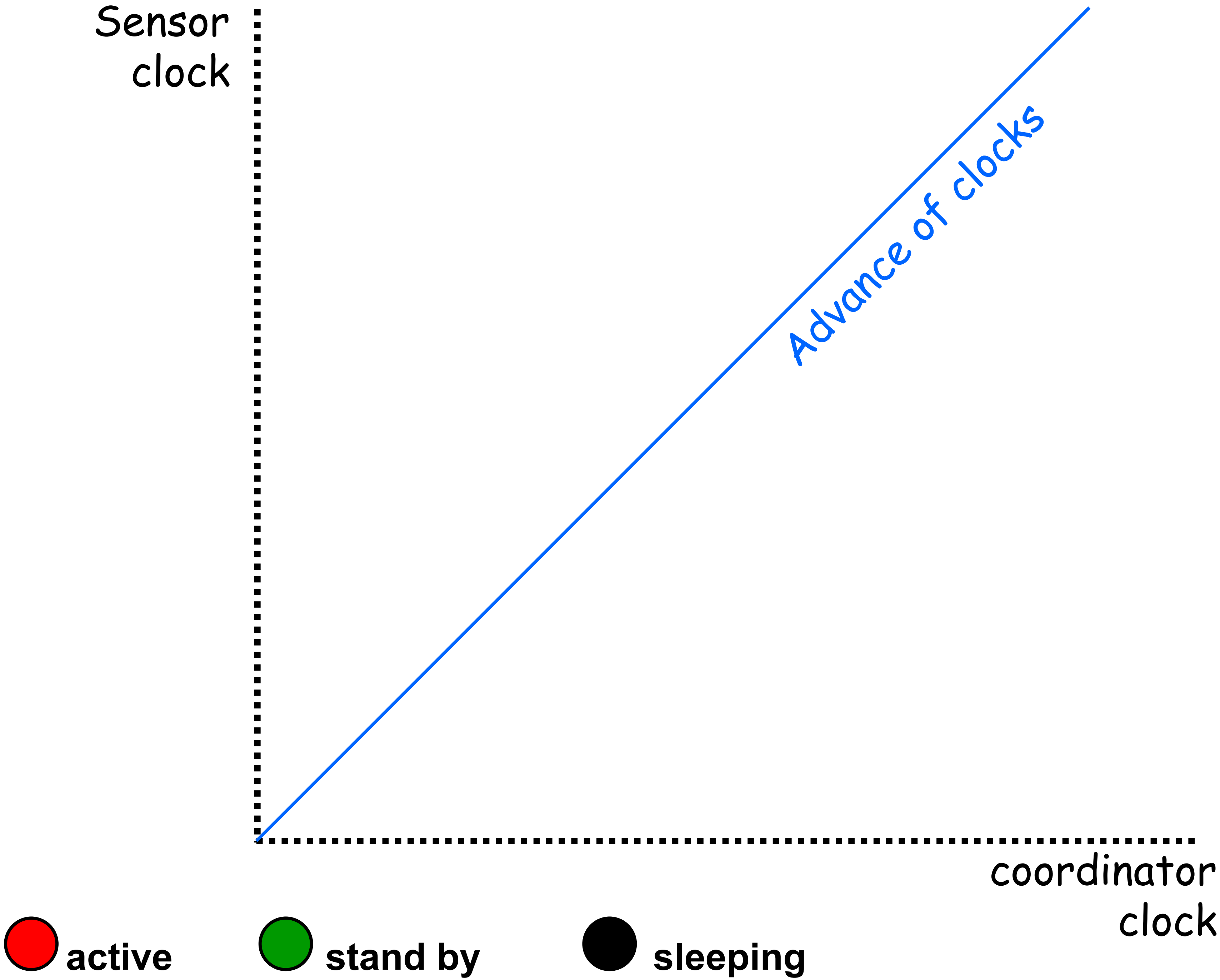


Sleep, but do not oversleep

Principle:

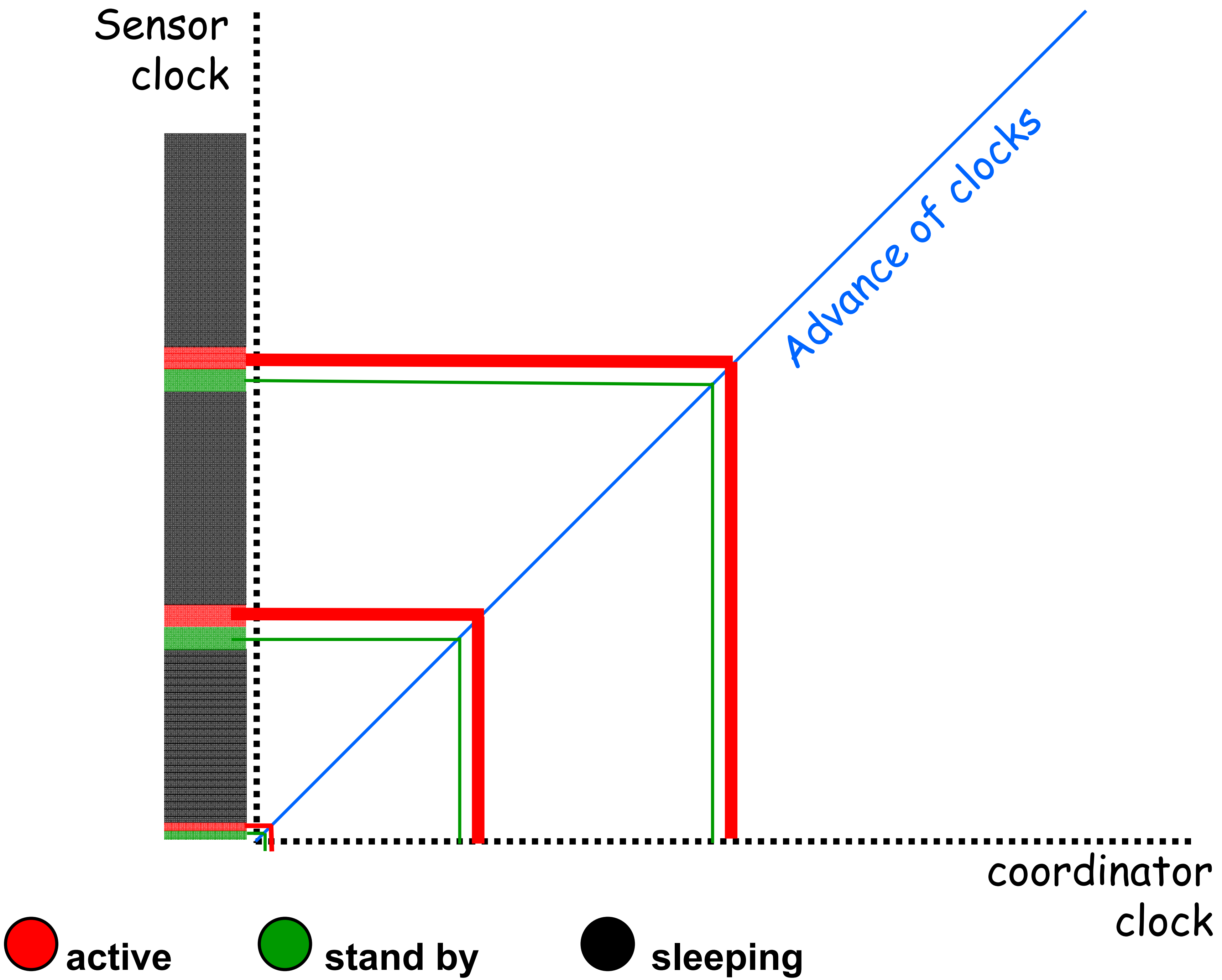
- Coordinator sends periodic beacon signal to indicate slot boundaries.
- Beacon interval ranges from 0.015 sec to 251.657 sec.
- Sensors go sleep in between whenever possible.
- Sensors are responsible for waking up on time, have their own clocks.

Wake up!



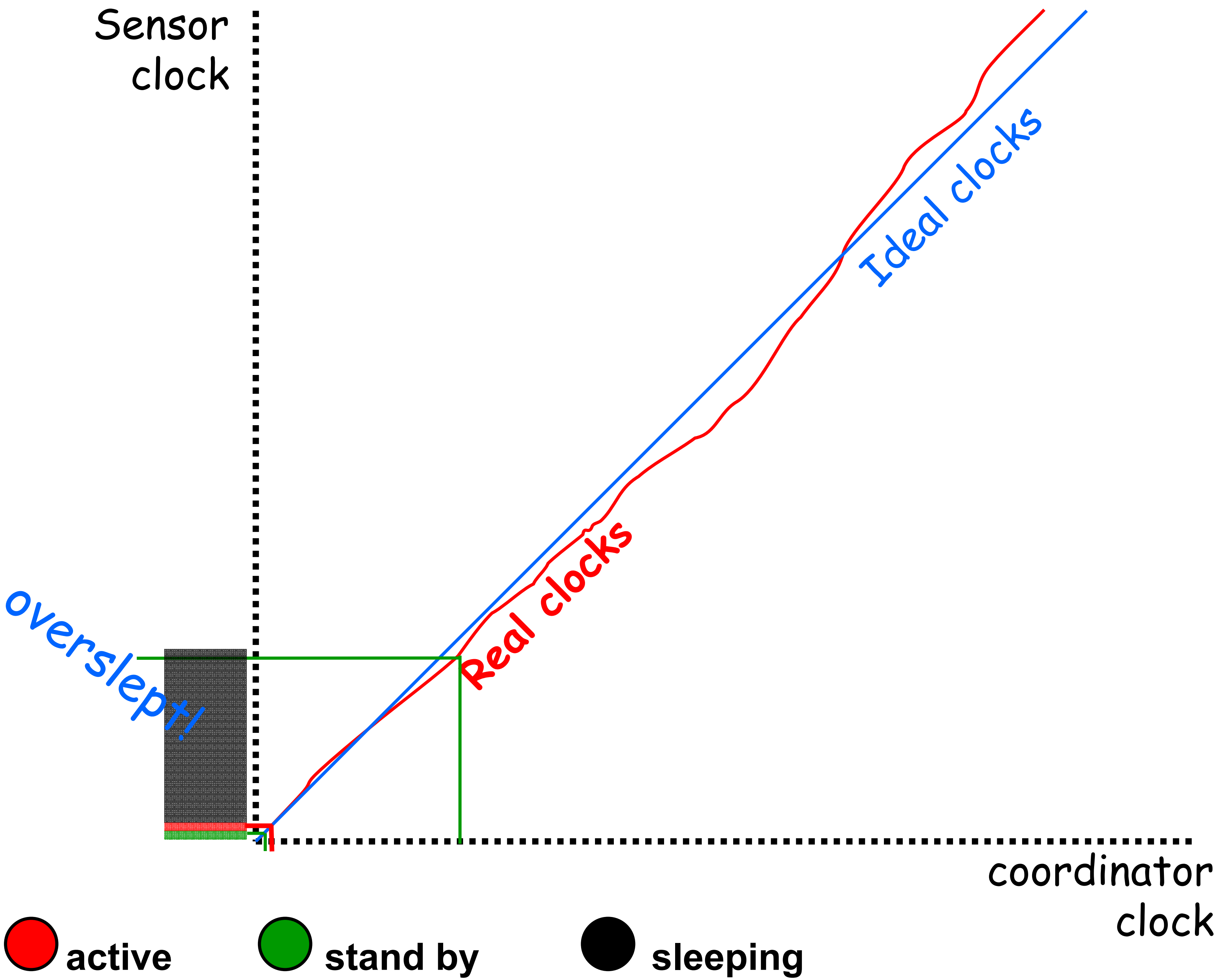
Holger Hermanns

Wake up!



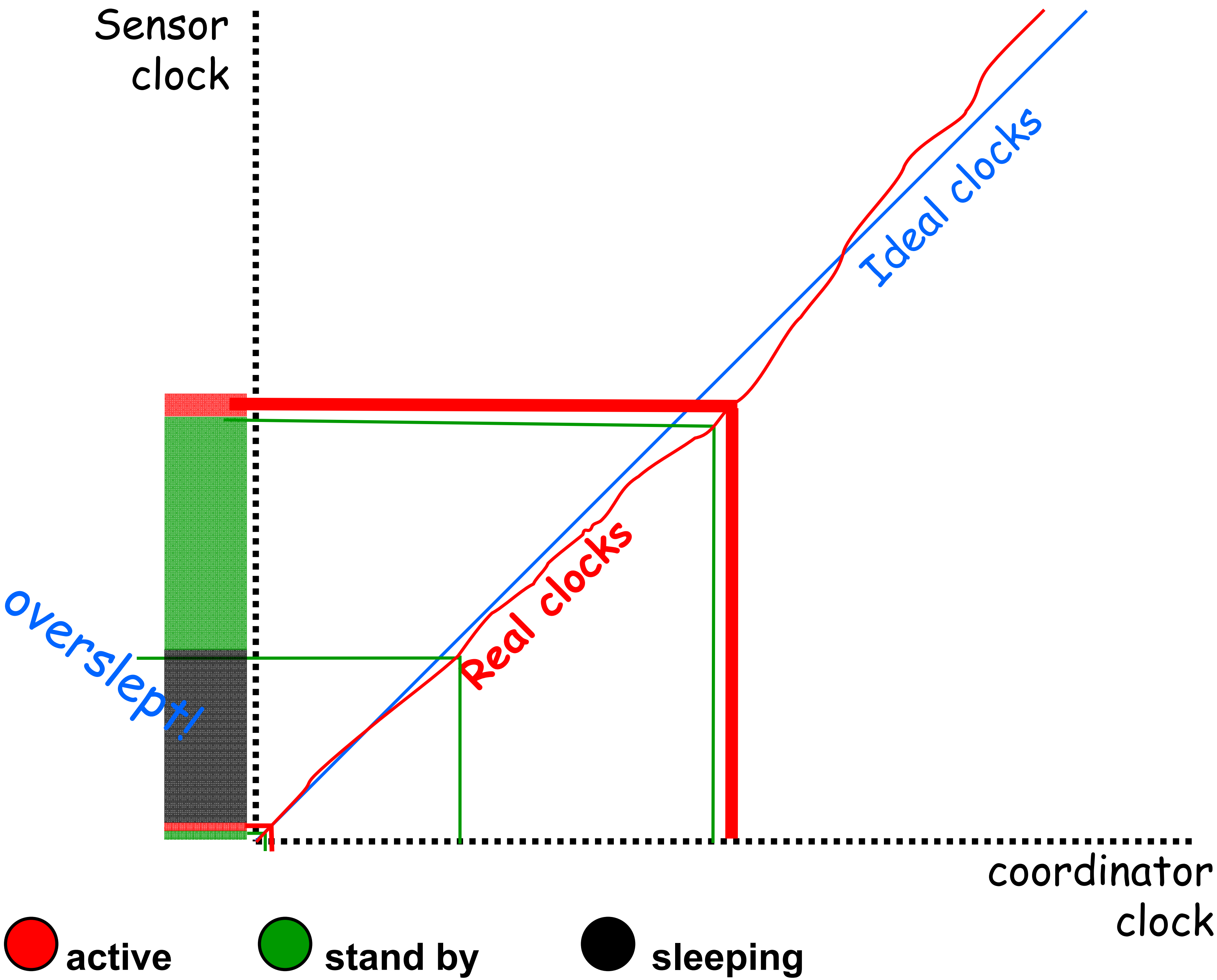
Holger Hermanns

Wake up!



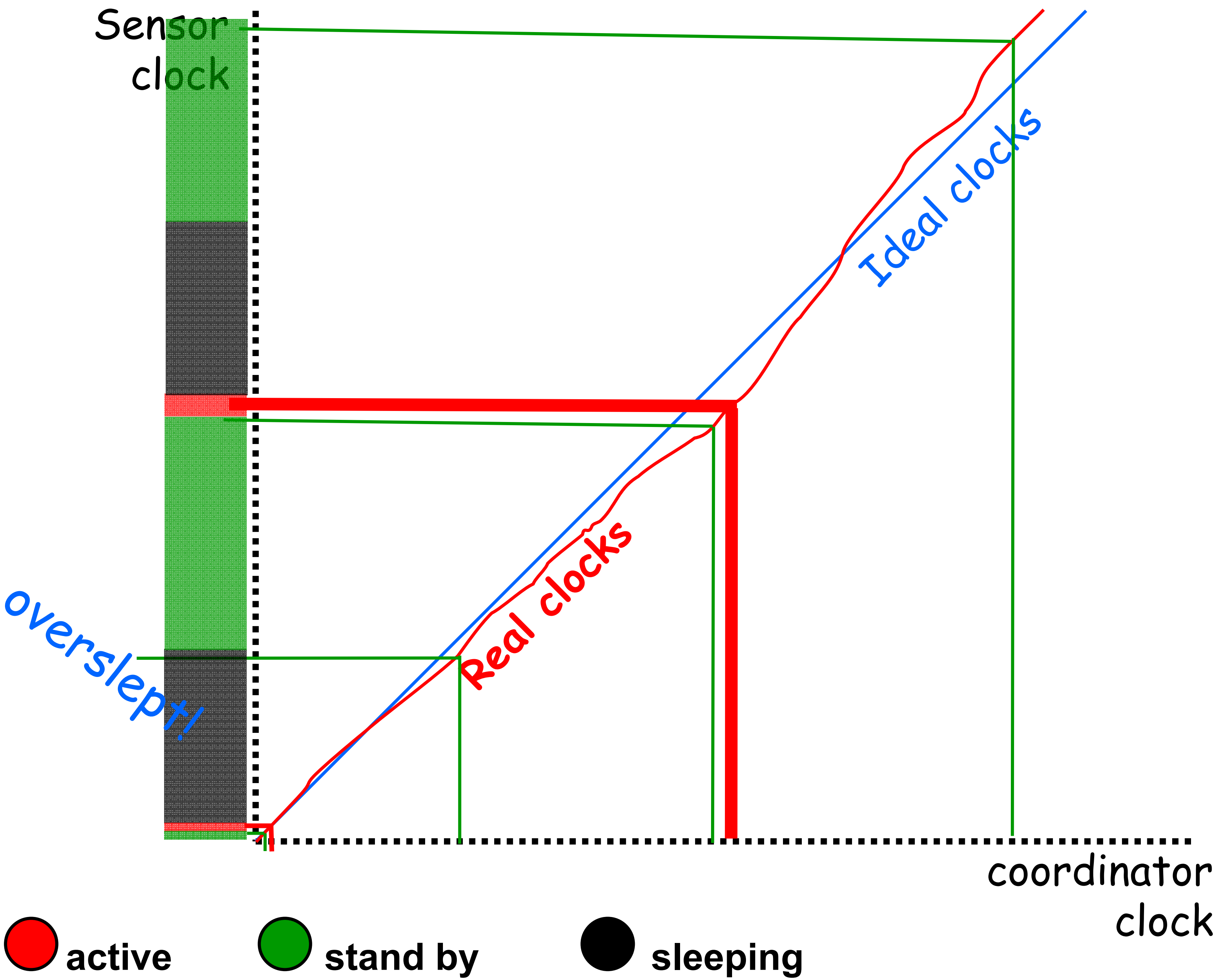
Holger Hermanns

Wake up!



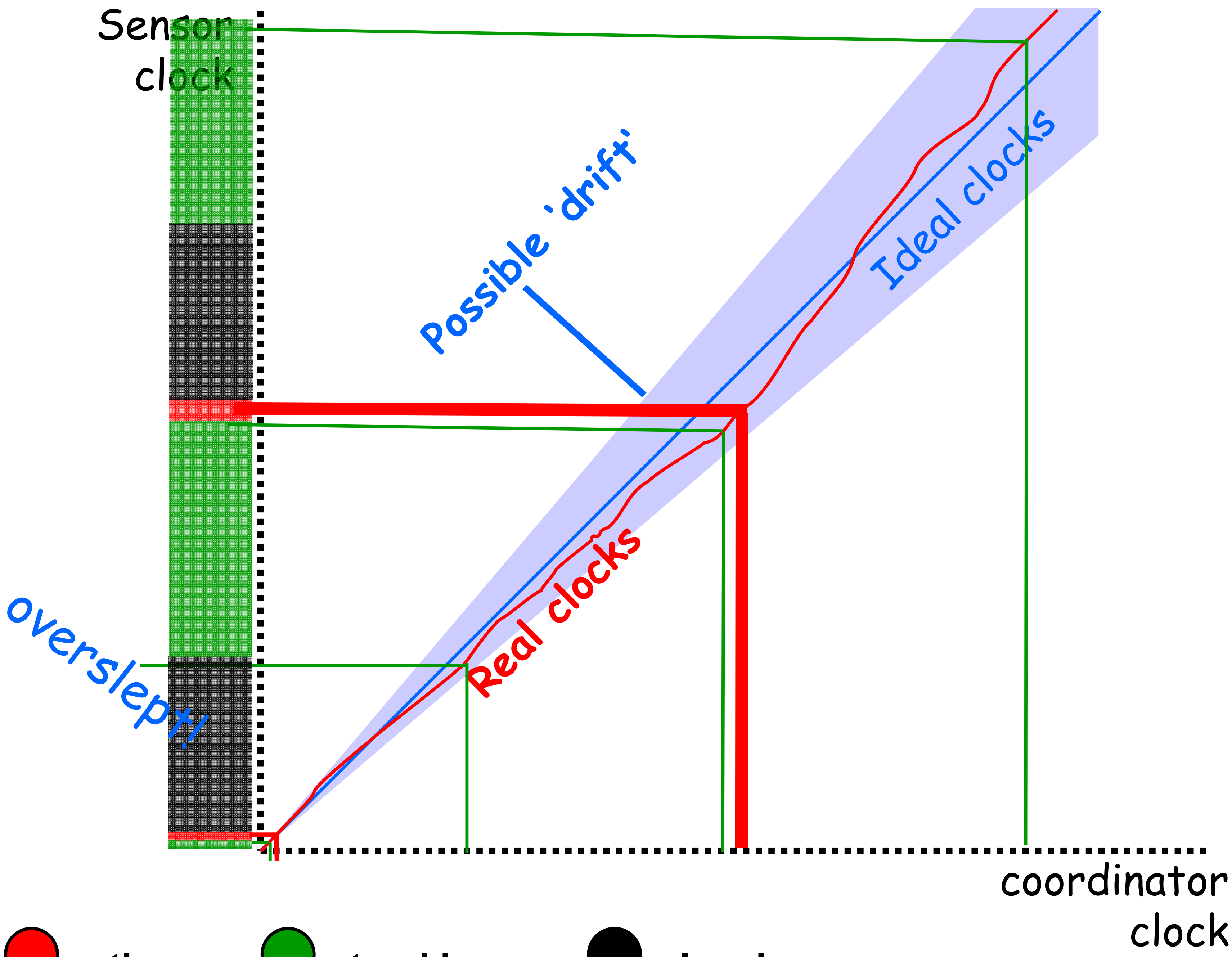
Holger Hermanns

Wake up!



Holger Hermanns

Wake up!

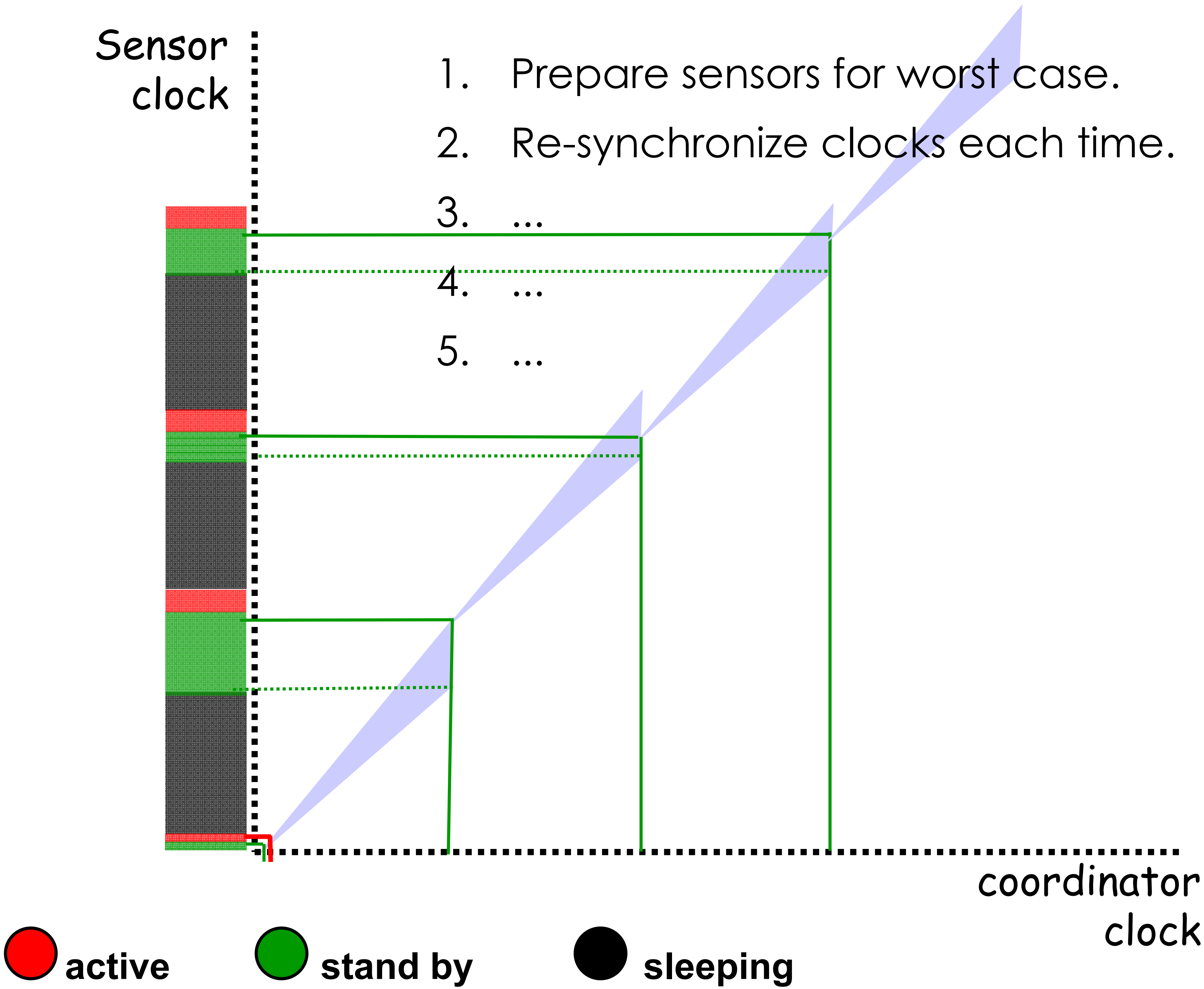


active stand by sleeping

Holger Hermanns

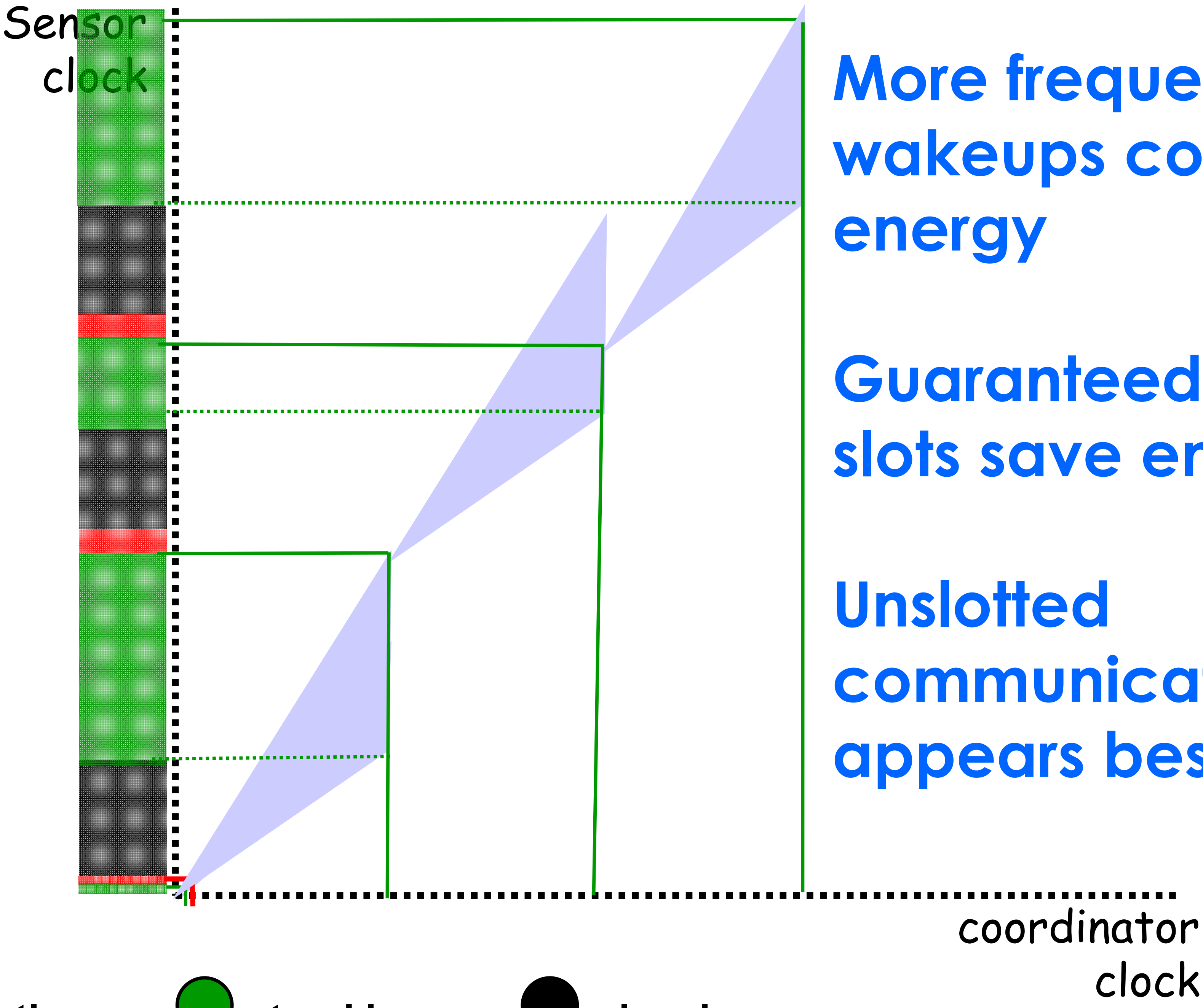
Make it better!

1. Prepare sensors for worst case.
2. Re-synchronize clocks each time.
3. ...
4. ...
5. ...



Results

Bad clocks cost energy



More frequent wakeups cost energy

Guaranteed time slots save energy

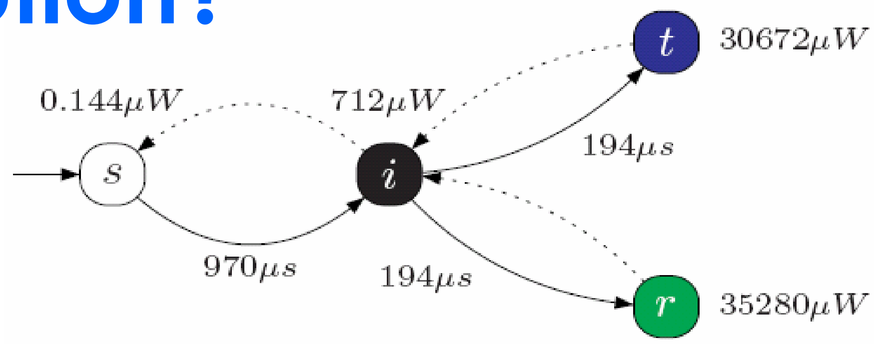
Unslotted communication appears best

active stand by sleeping

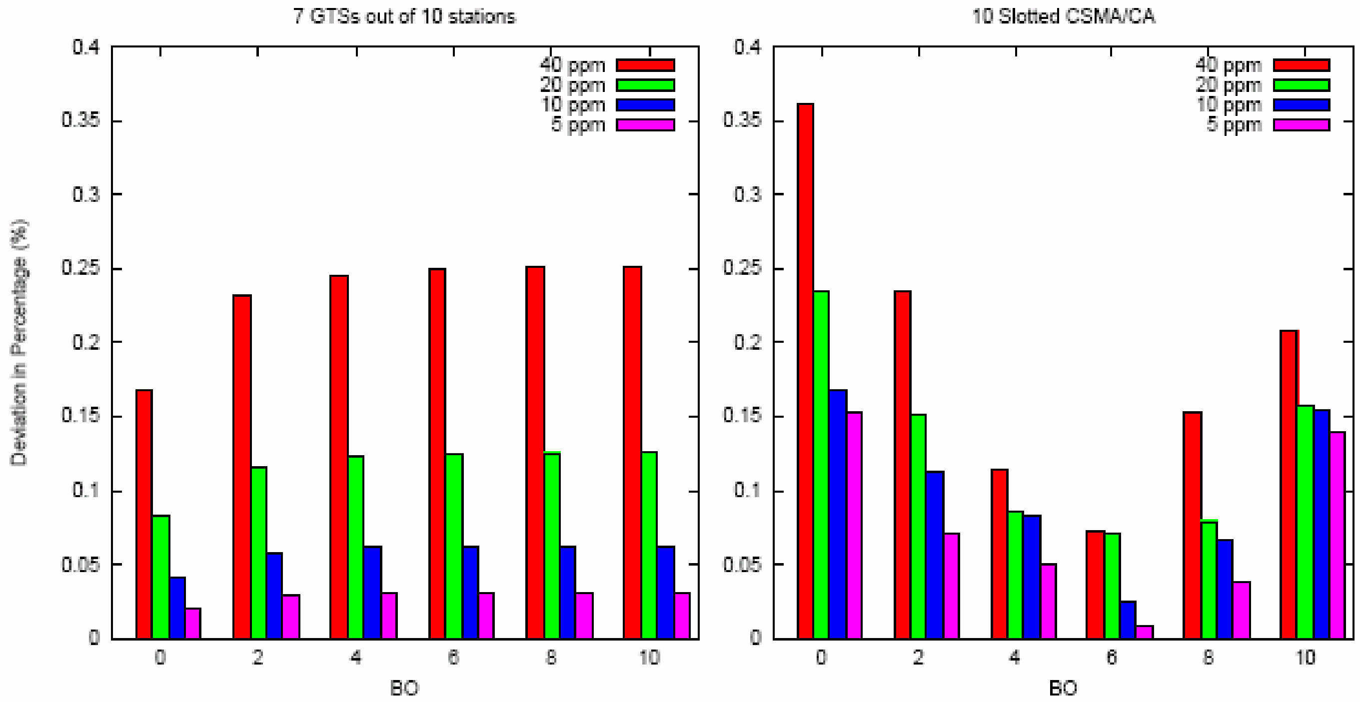
Holger Hermanns

Does Clock Precision Influence ZigBee's Energy Consumption?

- Well, yes.
  To a marginal extent.
- Drift is amplified by
  a factor of up to 60.



Energy Modes of CC2420



Half a day per year battery lifetime.

[Gross, H, Pulungan: OPODIS 2007]

Problem Synopsis

The particular question considered here can not be answered by lab experiments, mainly because

clock drift is uncontrollable.

Furthermore

battery lifetime effects take long to manifest themselves.

lab conditions are very difficult to establish.

Modelling

Modeling

```
process coordinator() {

  alt{:: do_act1
      :: do_act4
  }

}
```

```
process station() {

   do{:: do_act2
      :: do_act3
   }

}
```

```
par{
   ::coordinator()
   ::station()
}
```

Modeling
Probabilistic Choice

```
process coordinator() {

  palt{:1: do_act1
       :9: do_act4
  }

}
```

```
process station() {

   do{:: do_act2
      :: do_act3
   }

}
```

```
par{
   ::coordinator()
   ::station()
}
```

Modeling
Time

```
process coordinator() {

   do{:: do_act1
      :: comm_starts ; comm_ends
   }

}
```

```
process station() {
   clock c;
   do{:: do_act2
      :: comm_starts {= c=0 =} ;
         when (c==10) comm_ends
   }
}
```

```
par{
   ::coordinator()
   ::station()
}
```

Modeling
Random Variable

```
process coordinator() {

   do{:: do_act1
      :: comm_starts ; comm_ends
   }

}
```

```
process station() {
   clock c;
   do{:: do_act2
      :: comm_starts {= c=0 =} ;
         when (c==Exponential(10)) comm_ends
   }
}
```

```
par{
   ::coordinator()
   ::station()
}
```

Modeling
State Variable

```
process coordinator() {

   do{:: do_act1
      :: comm_starts ; comm_ends
   }

}
```

```
process station() {
   clock c;
   int sent = 0;
   do{:: do_act2
      :: comm_starts {= c=0 =} ;
         when (c==Exponential(10))
         comm_ends {= sent+= 1 =}
   }
}
```

```
par{
   ::coordinator()
   ::station()
}
```

Modeling
The Real System Model

```
par{
    ::coordinator()

    ::station_1()
    ::station_2()
...
    ::station_10()
}
```

Modeling
Relabeling

```
par{
   ::coordinator()

   ::relabel{...} by {....} station(1)
   ::relabel{...} by {....} station(2)
...
   ::relabel{...} by {....} station(10)
}
```

Modeling
The Coordinator

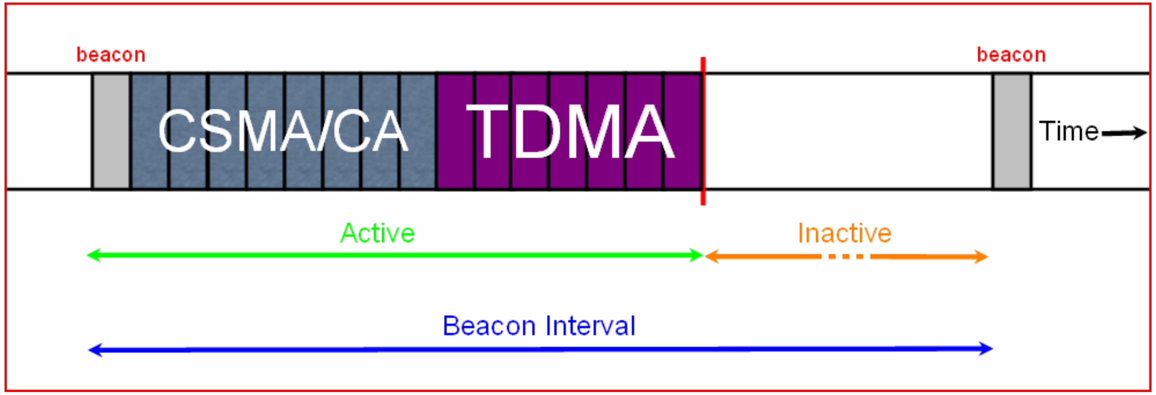
```
process coordinator() {
  clock btimer, c;

  do{:: sendb_start
            {= c=0, btimer=0, bintheair=true =} ;
        when(c==52) sendb_end
            {= bintheair=false =} ;
        when(btimer==binterval)
  }
}
```

Modeling

The Station

```
process station(int id) {
...
   do{::when(bintheair)
          beacon_received {= mainclock=0, ttosend=sendingtime =} ;
          when(!bintheair) attempt_sending ;
          do{::when(ttosend==0 || !enoughtime)
                 do_nothing {= enoughtime=true =} ; break
               ::when(ttosend>0 && enoughtime)
                 start_csmaca
...              \\ Slotted CSMA/CA Code Here
          }
...
   }
...
}
```

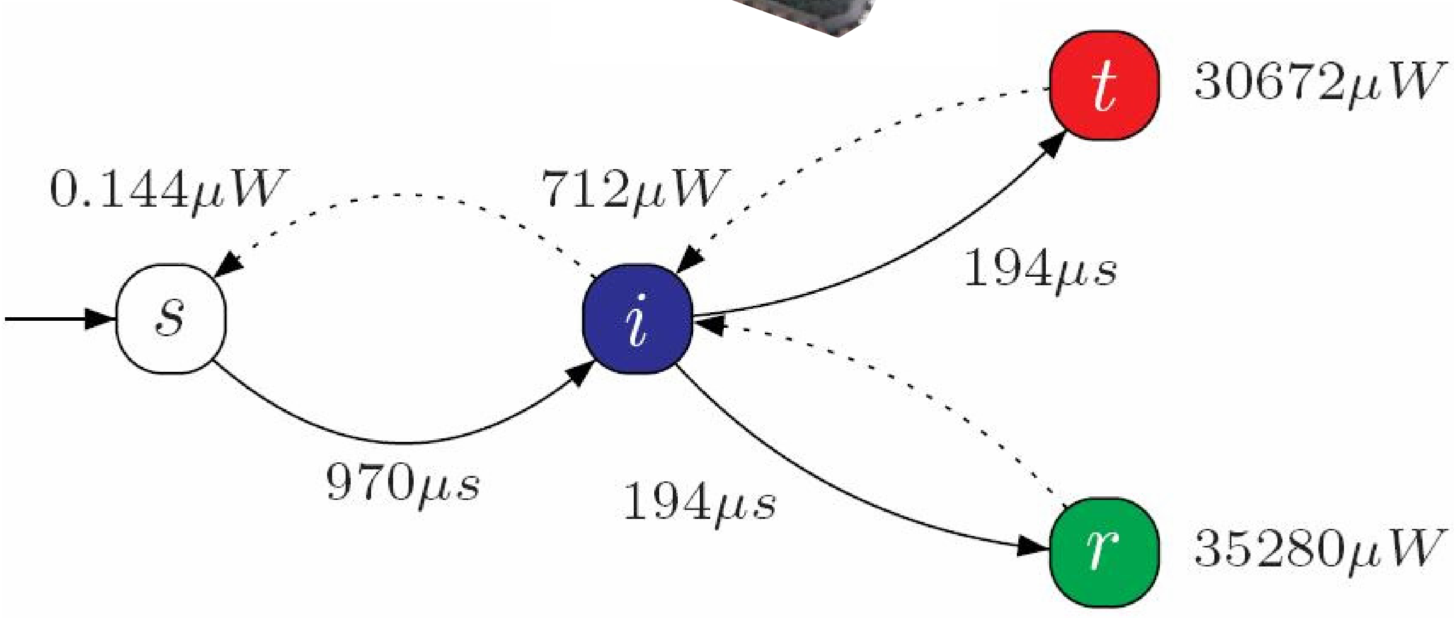
Modeling
The Station (contd.)




```
.. ...
alt{
  ::when(mainclock>=CAP-backofftime-ccatime) {= enoughtime=false =} ; break
  ::when(mainclock<CAP-backofftime-ccatime)
    when(c==backofftime) {= c=0 =} ;
    do{::when(c==ccatime)
         alt{::when(sending>0)
         alt{::when(NB<=maxbackoff)
                channel_busy {=CW=2,NB=NB+1,BE=(BE<6)?BE+1:6=} ; break
             ::when(NB>maxbackoff) {= restart=true =} ; break }
       ::when(sending==0 && mainclock<CAP-attosend-960)
          count_down_CW {= CW=CW-1 =} ;
         alt{::when(CW==0) wait_for_boundary ;
               when (c==320) send_message_start {= sending+=1, c=0 =} ;
               when (c==attosend) send_message_end
                 {= ttosend-=attosend, sending-=1, restart=true =} ; break
             ::when(CW>0) wait_for_boundary ;
               when(c==320) {= c=0 =} }
       ::when(sending==0 && mainclock>=CAP-attosend-960)
         {= enoughtime=false, restart=true =} ; break }
    } ; alt{::when(restart) {= restart=false =} ; break
  ::when(!restart)
} } } } } }
```

Modeling
Energy Consumption

Chipcon CC2420
from Texas Instruments




```
...do{:: when (CW==0) start_on_next_backoff_boundary ;
         when (c1==320) start_send_message {= sending+=1, c2=0 =} ;
         when (c2==actual_to_send) finish_send_message
                         {= still_to_send-=actual_to_send, sending-=1,
                            messages_sent[id]+=1, continue_backoff=0,
                            message_to_send=0,
                            time_in_tx_mode[id] += actual_to_send + 194
                          =} ; break    :: ...
}
```

Modeling
Clock Precision

- Inaccuracy p ppm means at time t :

$$[t - p \cdot 10^{-6} \cdot t, t + p \cdot 10^{-6} \cdot t]$$

- Maximal clock drift allowed is 40 ppm
- Waiting time:

$$W' = W - W \cdot p \cdot 10^{-6} + W \cdot p' \cdot 10^{-6}$$

```
...when(c1 == wait_in_backoff
            - wait_in_backoff * precision[id]
            + wait_in_backoff * random_drifts[id])
perform_cca  {= c2=0, ... =}...
```

Modest, a language for stochastic timed system

- Rooted in classical process algebra:
  - atomic actions,
  - sequential & parallel composition,
  - nondeterministic choice,
  - looping
- Probabilistic choice,
  exception handling, and
  hard and soft real-time aspects

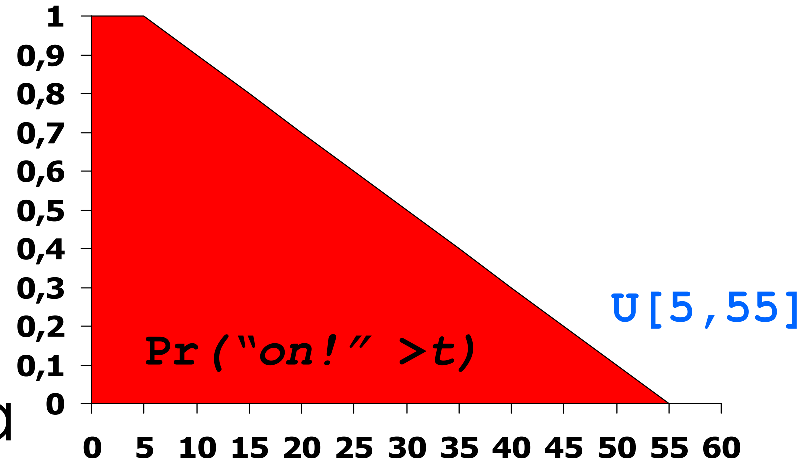
- Conventional programming constructs

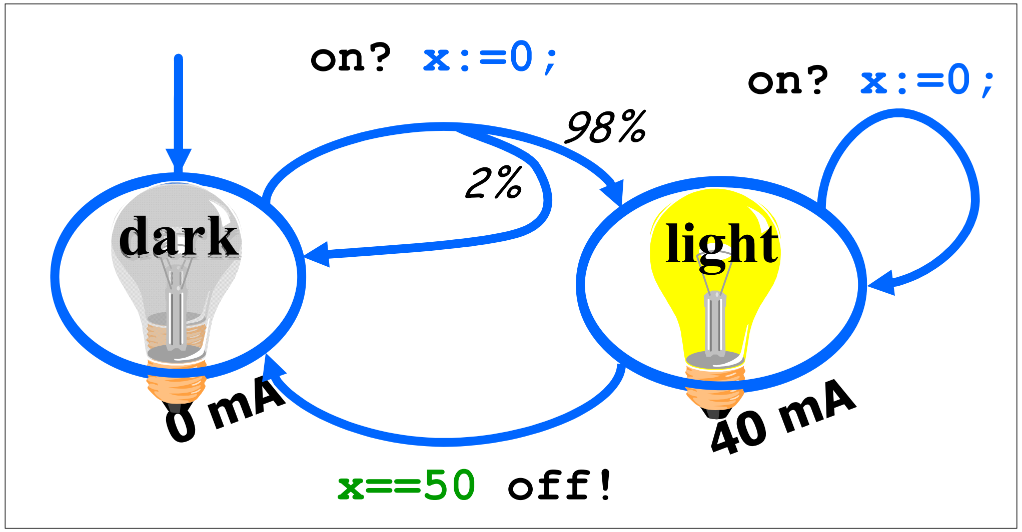
- Semantics: Stochastic Timed Automata (STA)

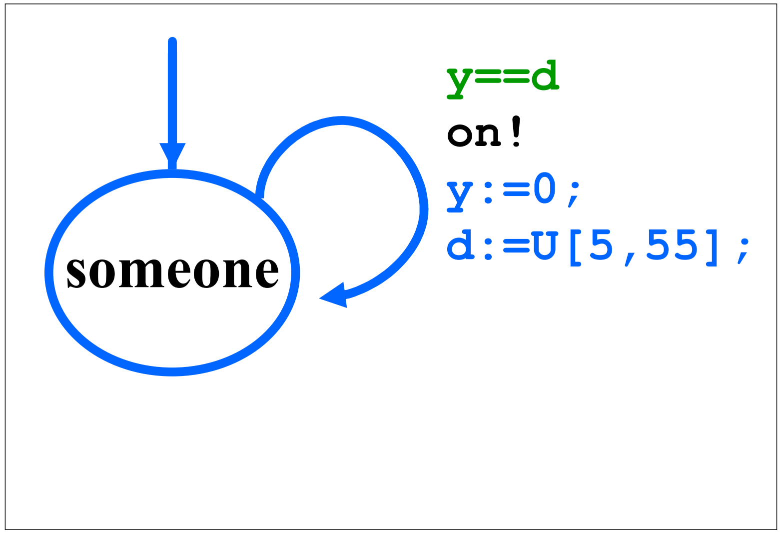
[Bohnenkamp, D'Argenio, H, Katoen: IEEE TSE 32(10)]

Stochastic Timed Automata

- finite automata
- with clocks
- and with costs
- modular: composition of automata







- with continuous probability distributions
- and with discrete probabilistic branching

[Bohnenkamp, D'Argenio, H, Katoen: IEEE TSE 32(10)]

Analysis

Holger Hermanns

Analysis Trajectory

- **STA are as yet too expressive to be analysable (by us)**
  - no nondeterminism: Discrete event simulation
  - no probabilism: real-time model checking
  - discrete probabilism only: PTA model-checking

What we do:

- **Semi-automatic abstraction**
  to arrive at a timed automata model.

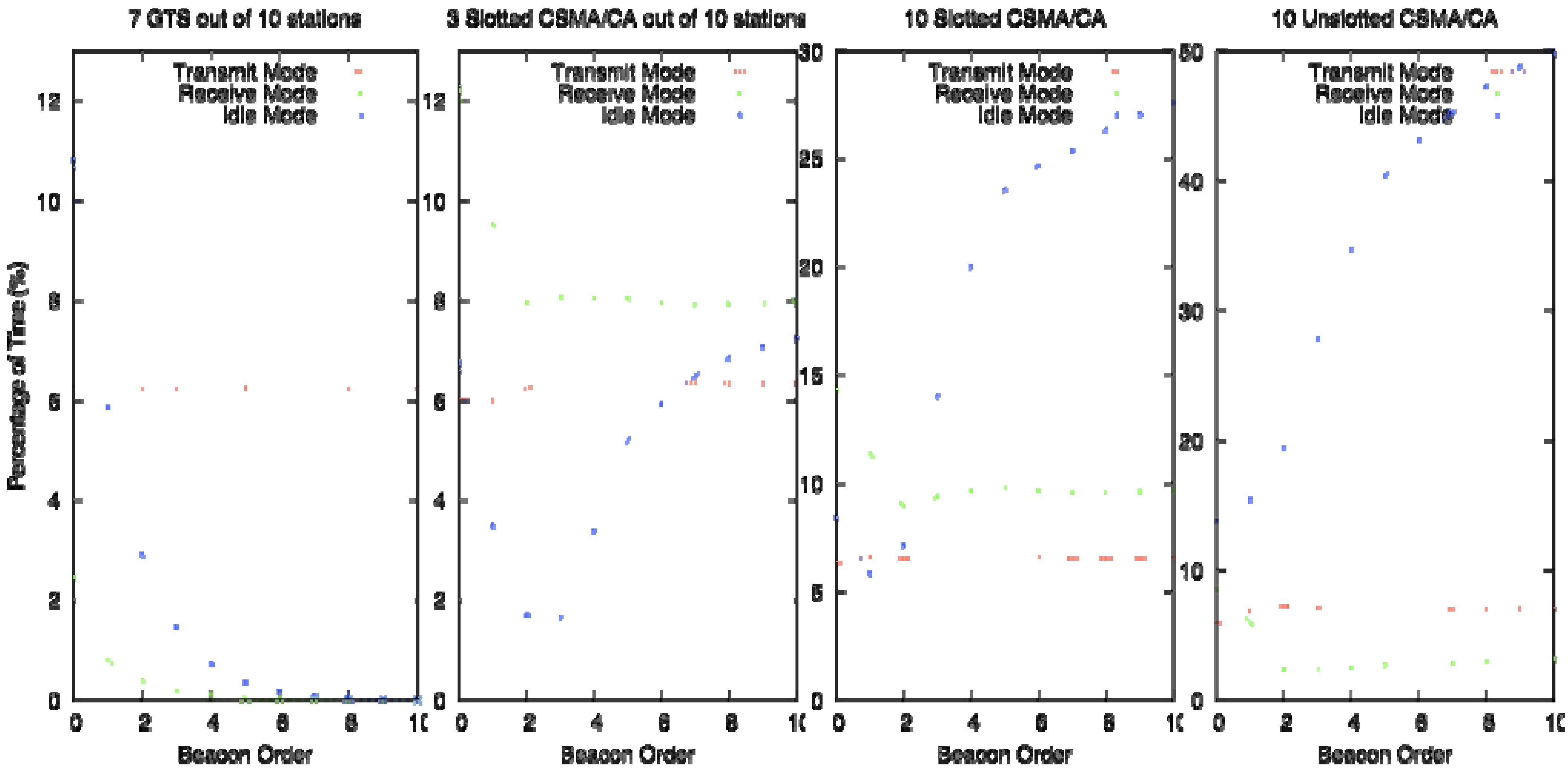
- **UPPAAL** to generate schedules with worst cost.

- Derive insights on worst-case clock drifts

- Code this worst case into the model

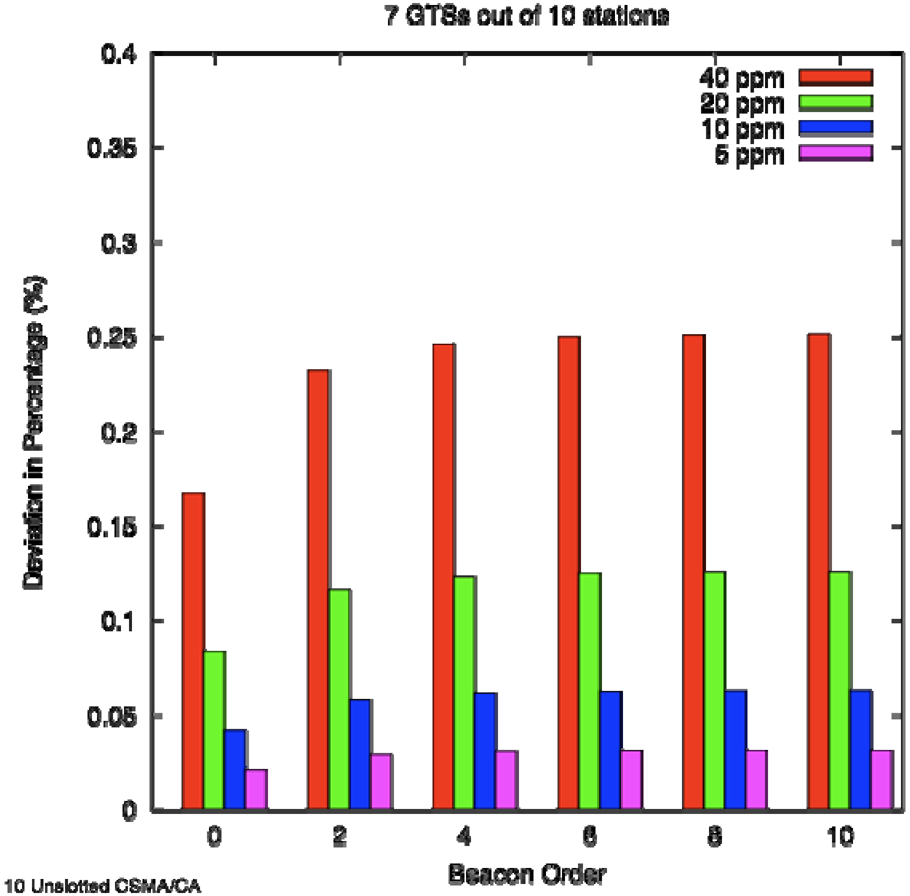
- **Motor/Möbius** to simulate scheduled Modest specs.

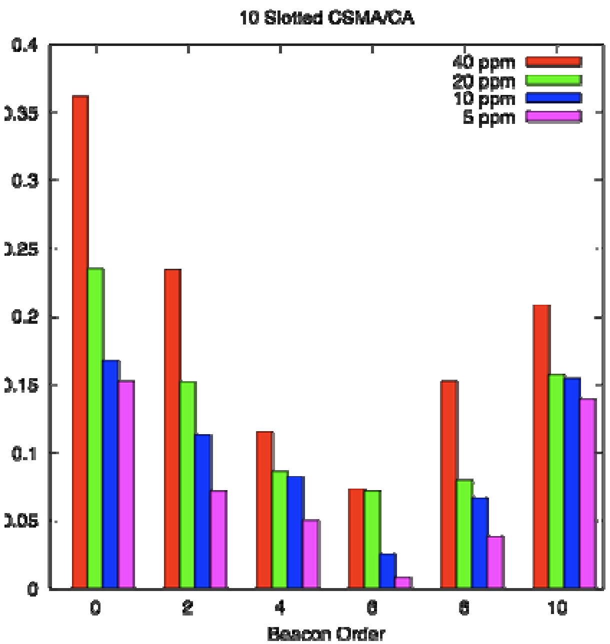
Results
Perfect Clock

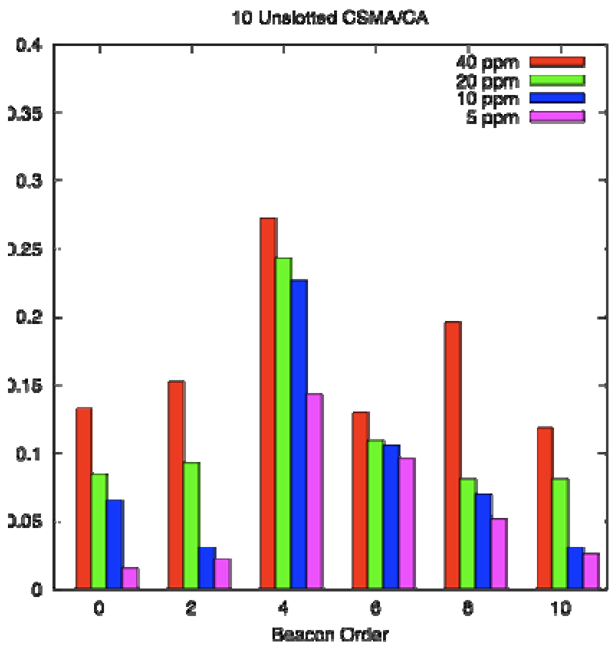


Holger Hermanns

Results
Drifting Clock







Holger Hermanns

Epilogue

Claim *revisited*

- There is an enormous spectrum of intriguing research questions out there … in embedded system design

- Many of them
  - are crying for mathematically well-founded investigations.
  - are very tough to investigate with current technology.

- Often, there is no other way.

- So, let's invest in better modelling and analysis techniques for them.

How to validate a hypothesis ?

Model validation	Simulation	Emulation	Empirical studies
UPPAAL	TOSSIM	EmStar	
PRISM	Avrora	Bee	
Motor	Glomosim	…	
	20-sim		
	Modelica		
	Matlab		
	…		

How to validate a hypothesis ?

Model validation | Simulation | Emulation | Empirical studies

good	General?	bad
good	Time/cost effective?	bad
bad	Accurate?	good
good	Reproducible?	bad
bad	Scalable?	bad

Holger Hermanns

What we need

Model validation | Simulation | Emulation | Empirical studies

From Models to Experimentation and Back

Why we need that

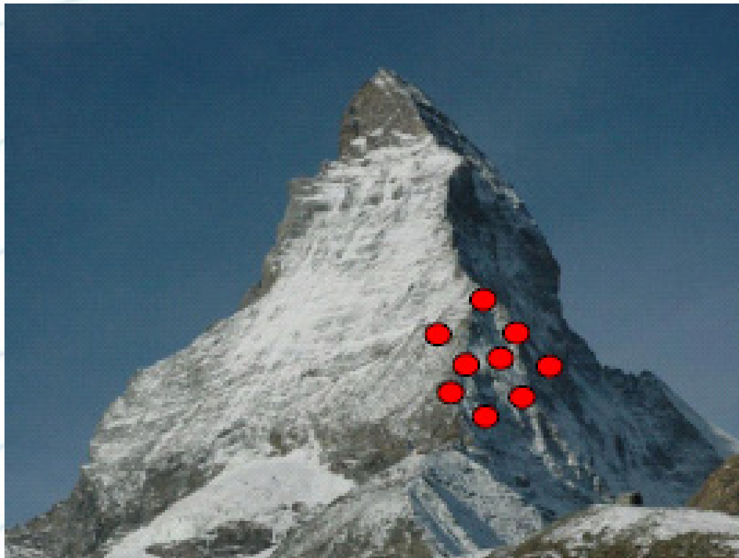
Why we need that

Remember the time when "compilers" didn't yet have error/warning messages?

Jan Beutel, ETHZ

PermaSense project @ Matterhorn

and many others



NCCR MICS
National Competence Center In Research
Mobile Information and Communication Systems

FN SNF
Fonds National Suisse
Schweizerischer Nationalfonds
Fondo Nazionale Svizzero
Swiss National Science Foundation

What we need

Model validation | Simulation | Emulation | Empirical studies

From Models to Experimentation and Back

- The holy grail:

Sound models and sound abstractions

Go!