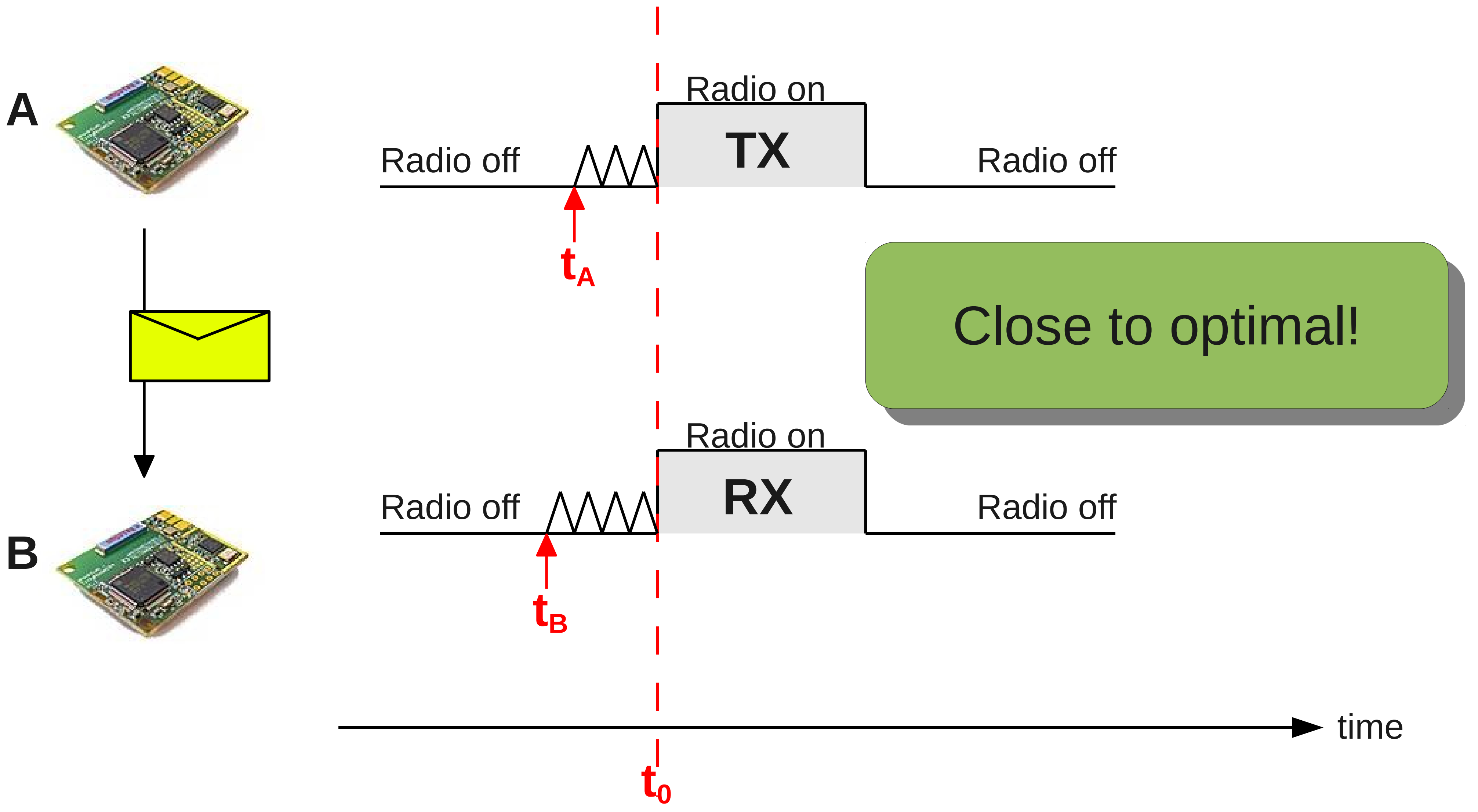
Slotted Programming for Sensor Networks

Roland Flury
Roger Wattenhofer

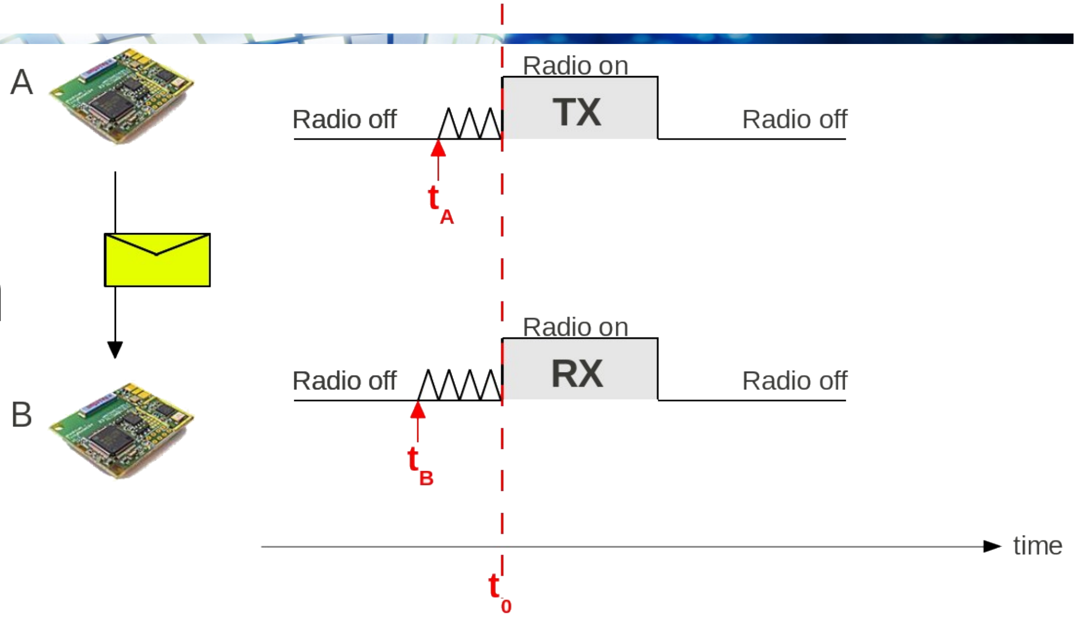
Distributed Computing Group
ETH Zurich, Switzerland

Energy Efficient Communication



Close to optimal!

Requirements



- Good Synchronization
- Exact time execution
  - No delay
  - If B is too late, it misses the message
- Radio must be free to use
  - No other task may be using the radio

Slotted Programming can ensure these properties

Programming Sensor Nodes

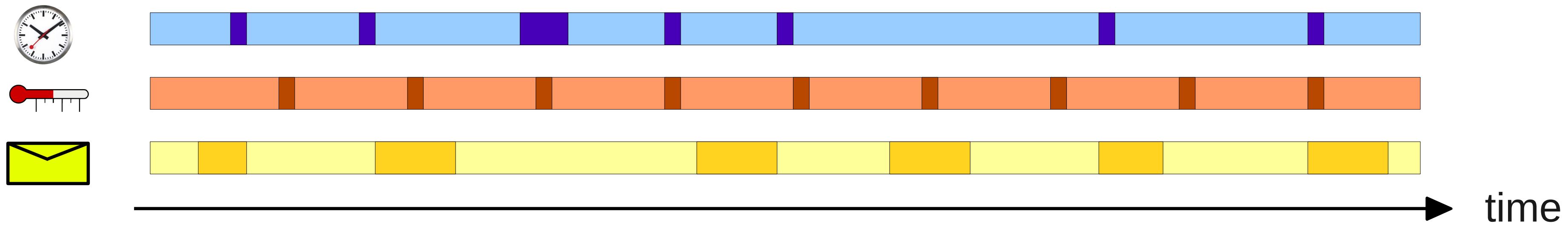
- What are the time critical sections?
  - E.g. wake up the radio at t_0 to receive a message
- Can they be delayed by another task?
- Can we avoid it?
- Check access to any device, not only radio

Need to analyze the entire application

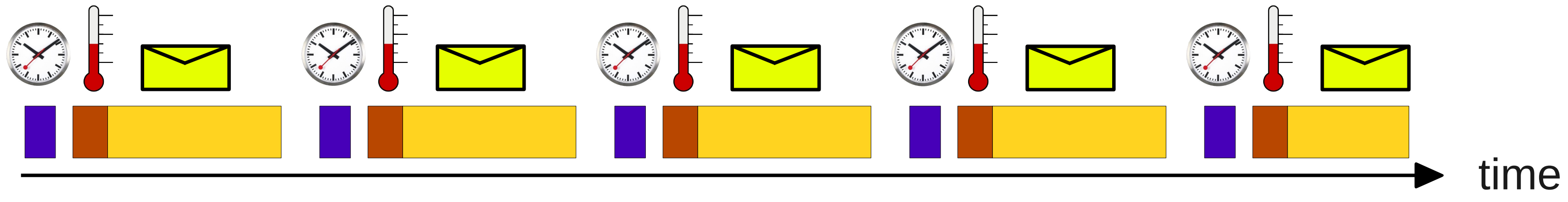
Any part may interact with any other part

Traditional vs Slotted

Traditional Application

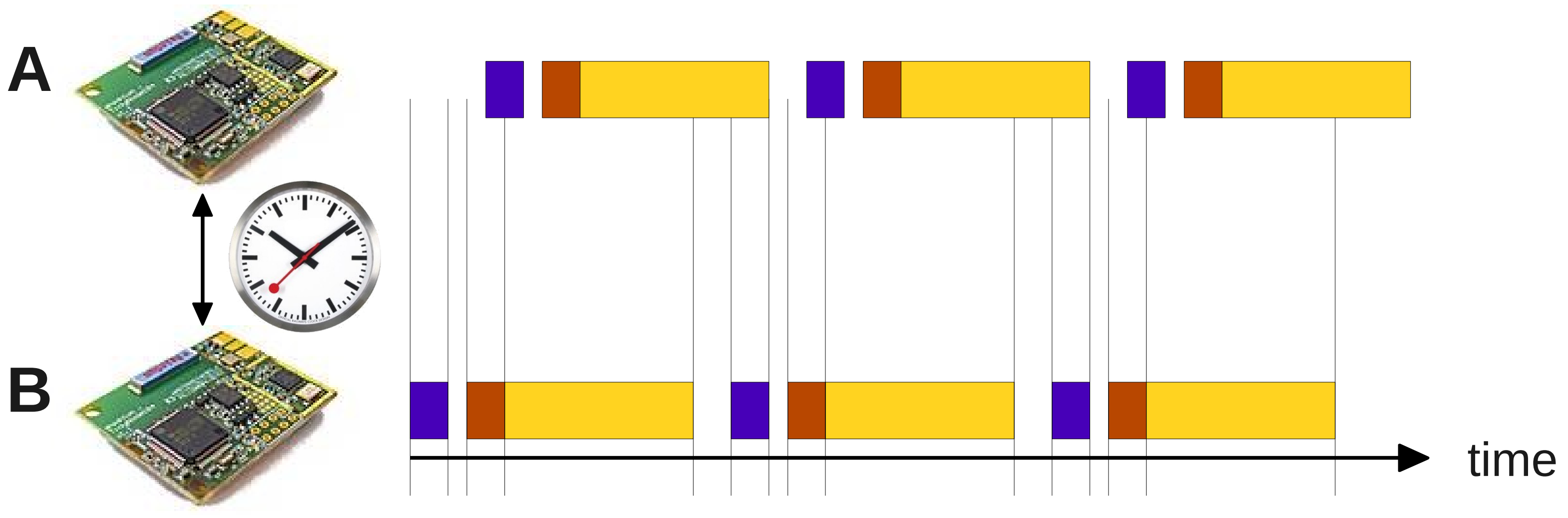


Slotted Application: Temporal separation of the tasks



- No interference of other tasks
- Can analyze each task separately

Slotted & Synchronized

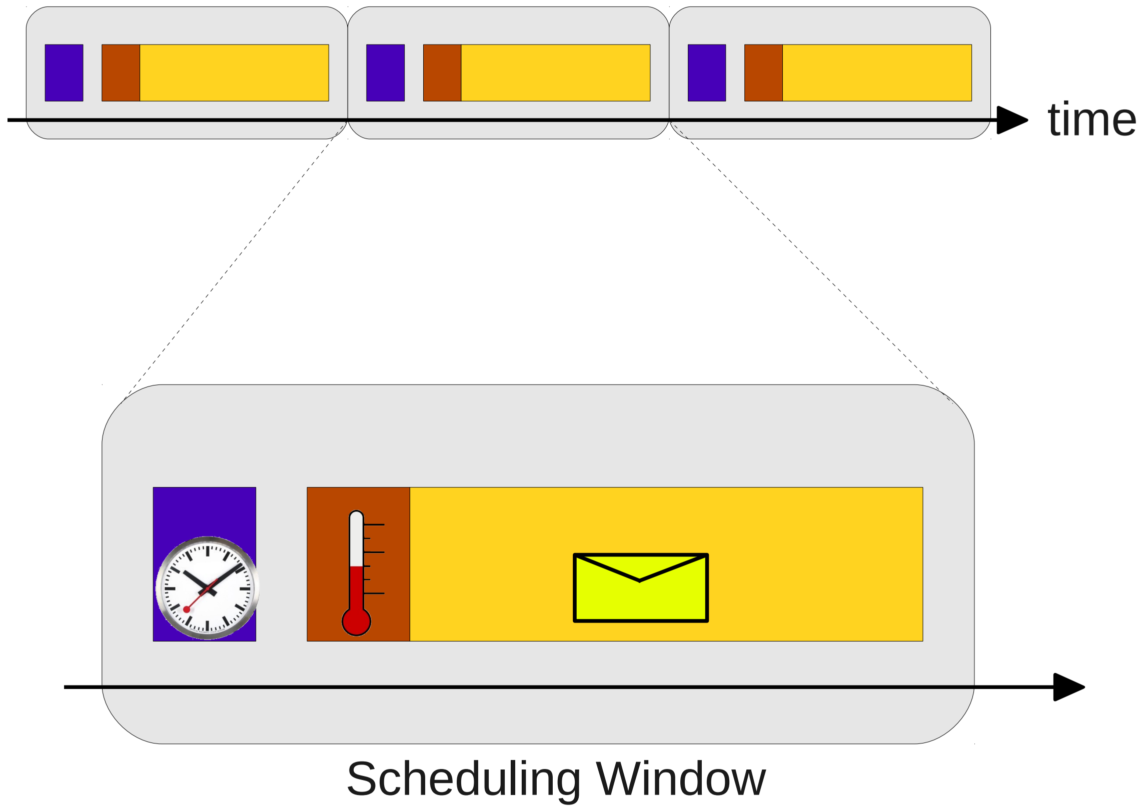


All nodes need to be synchronized

- Clock sync module does all the work
- Transparent to the other modules

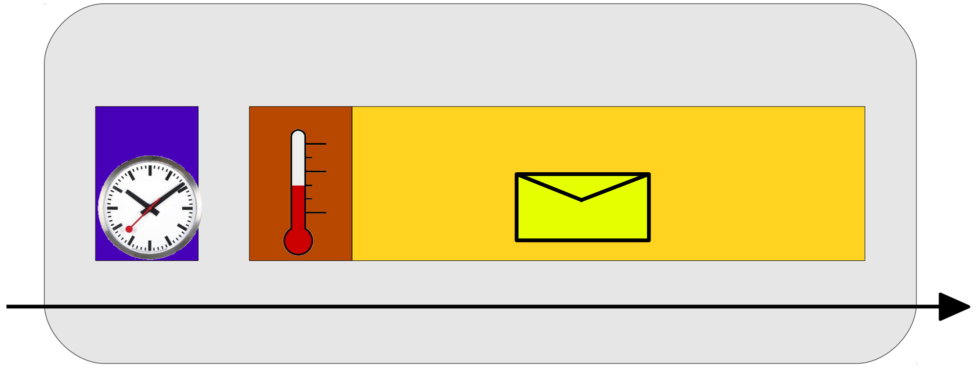
Slotted Programming

Easy case: repetitive schedule



Advanced schedules are possible

Slotted *Programming*



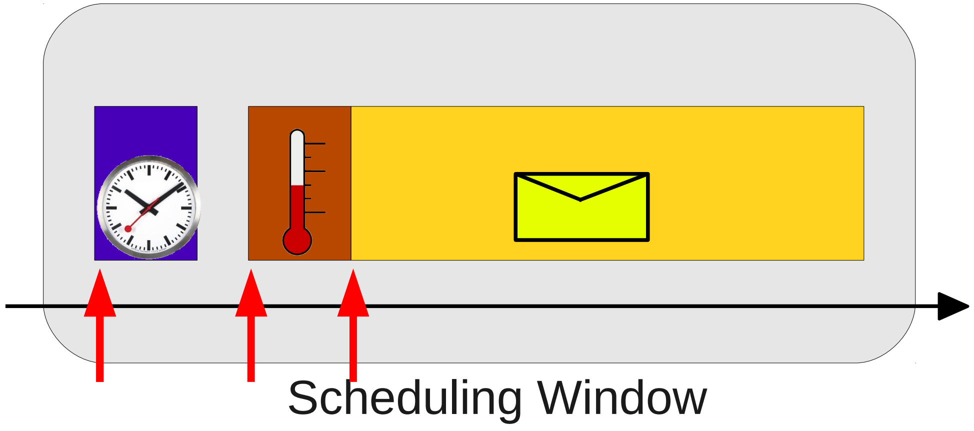
Scheduling Window

Slotted *Programming*

1) @boot: allocate slots

2) schedule slots:

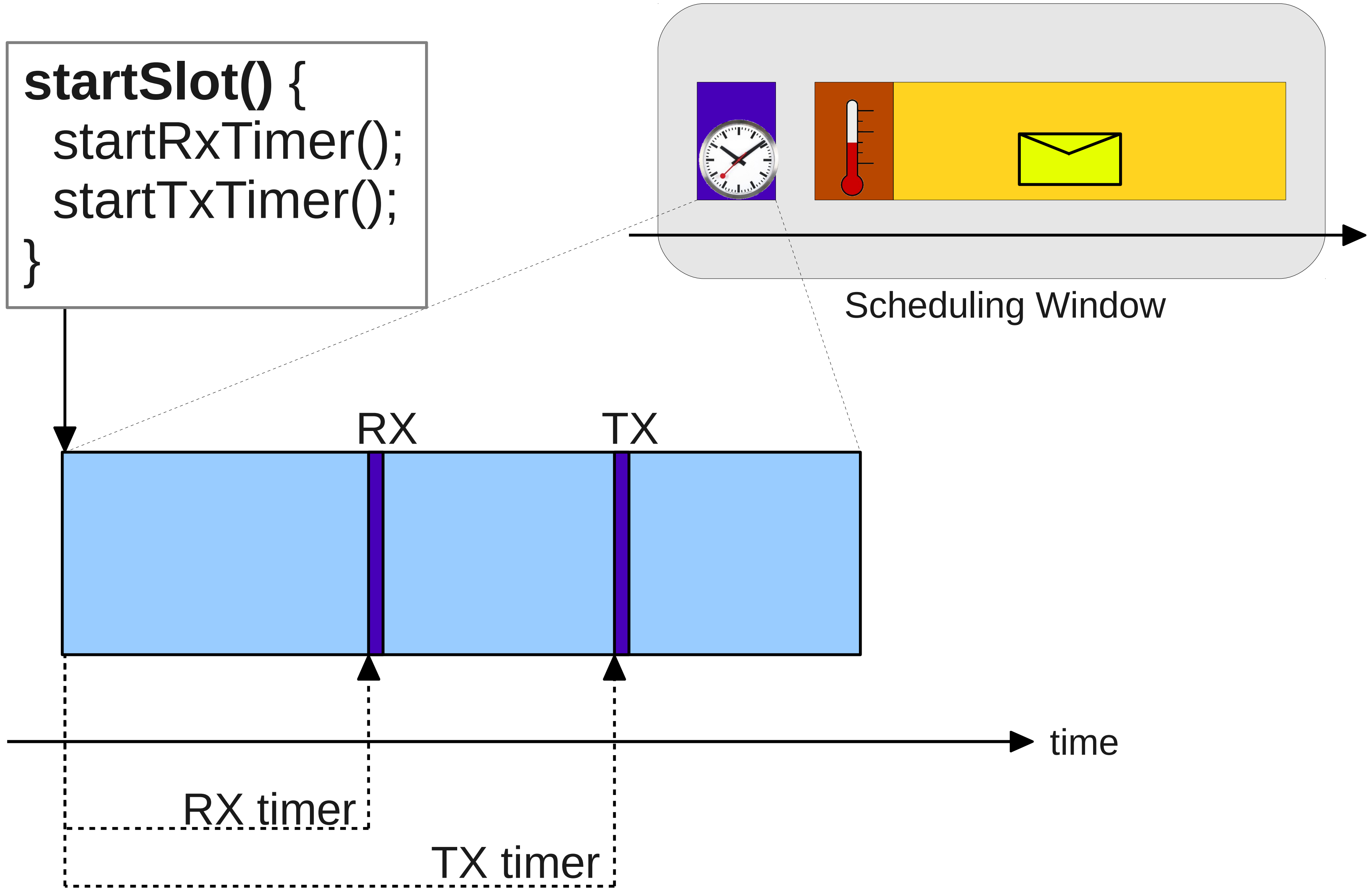
    startSlot() - stopSlot()



	@boot	@runtime	
Clock Sync	init()	startSlot()	stopSlot()
Sampling	init()	startSlot()	stopSlot()
Routing	init()	startSlot()	stopSlot()

Slotted *Programming*

```
startSlot() {
  startRxTimer();
  startTxTimer();
}
```



Slotted Programming

Very simple: Time division

Too simplistic?

- No side effects
  - Additional module does not disturb existing app
- Guaranteed access to resources
- Simplified software structure
  - Step *towards* provably correct software
- Energy efficient applications
- Clock Sync is transparent

Examples

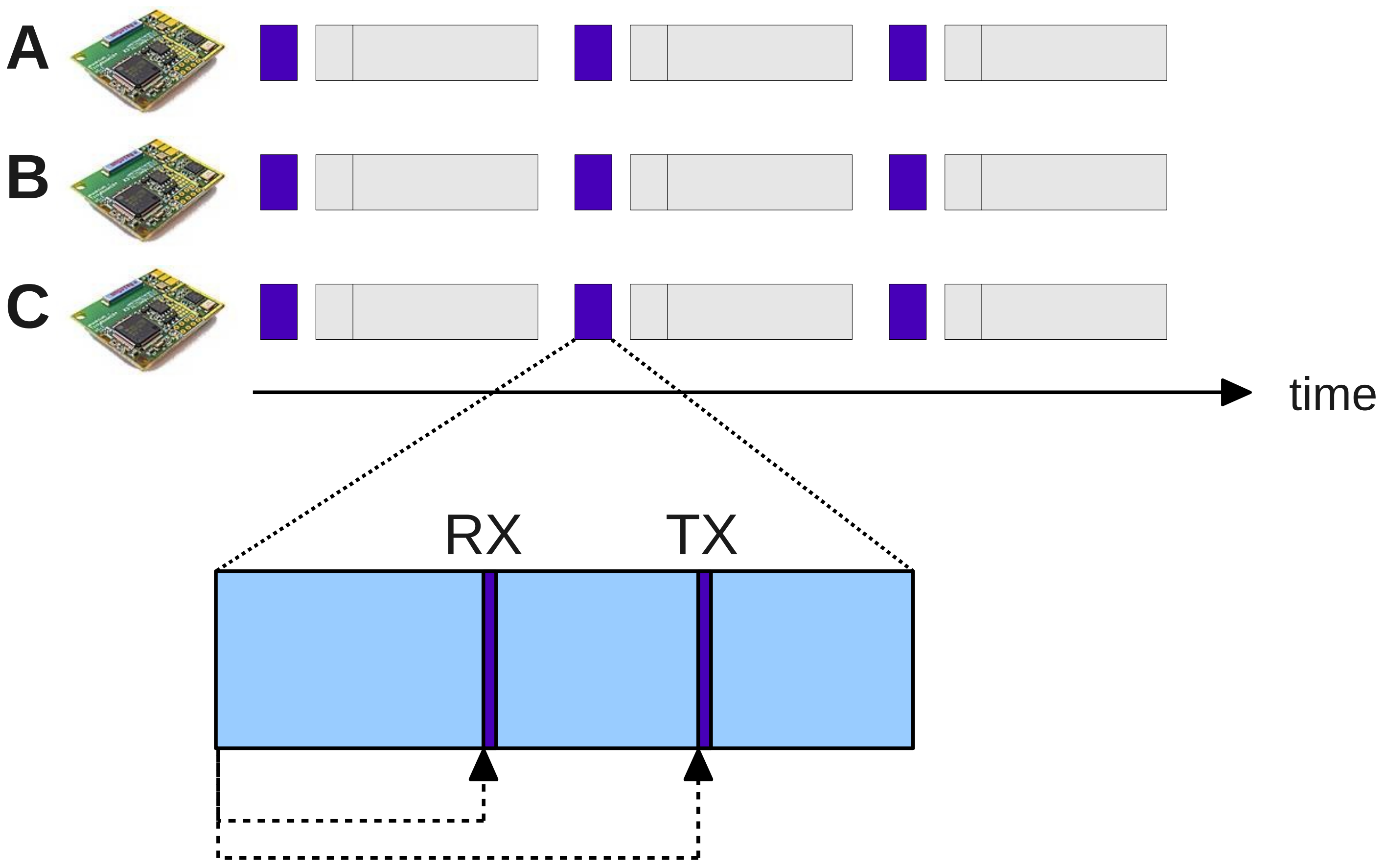
- **Slotted Clock Synchronization**
- Energy Efficient Alarming
- Data gathering

Goal
Synchronize all nodes of a network

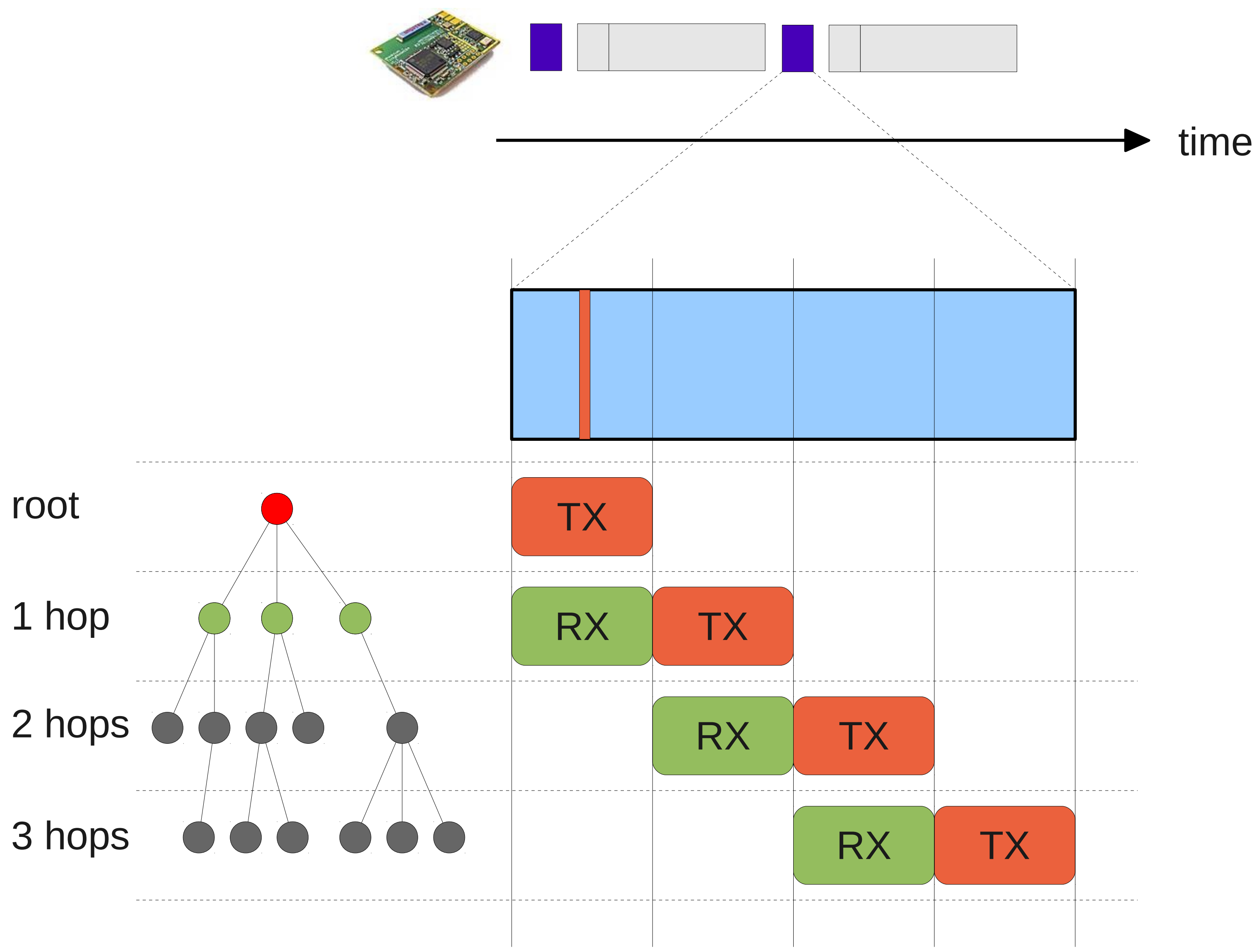
A master node dictates its time
to the remaining nodes

Slotted Clock Synchronization

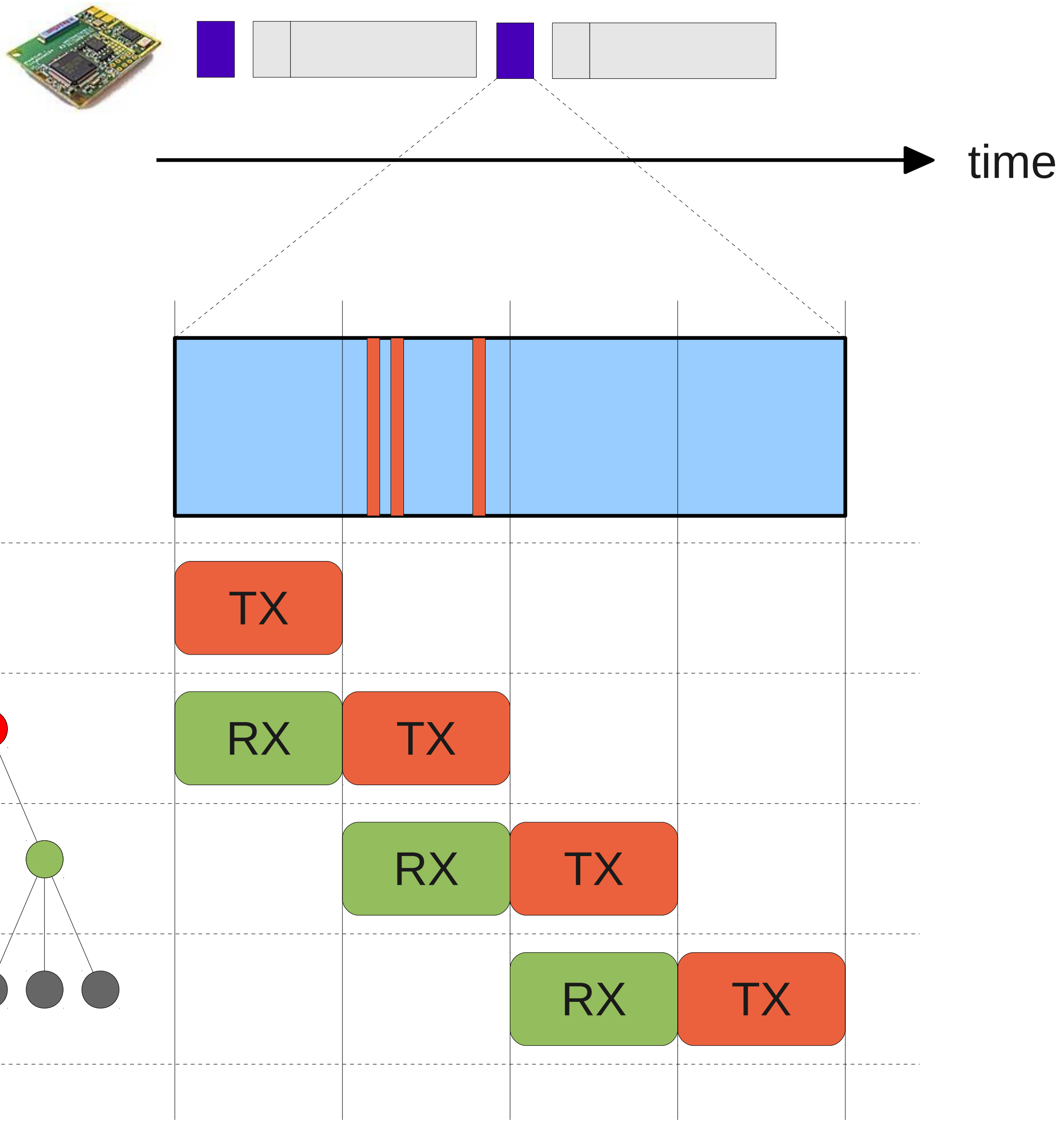
Bootstrap procedure to get rough synchronization



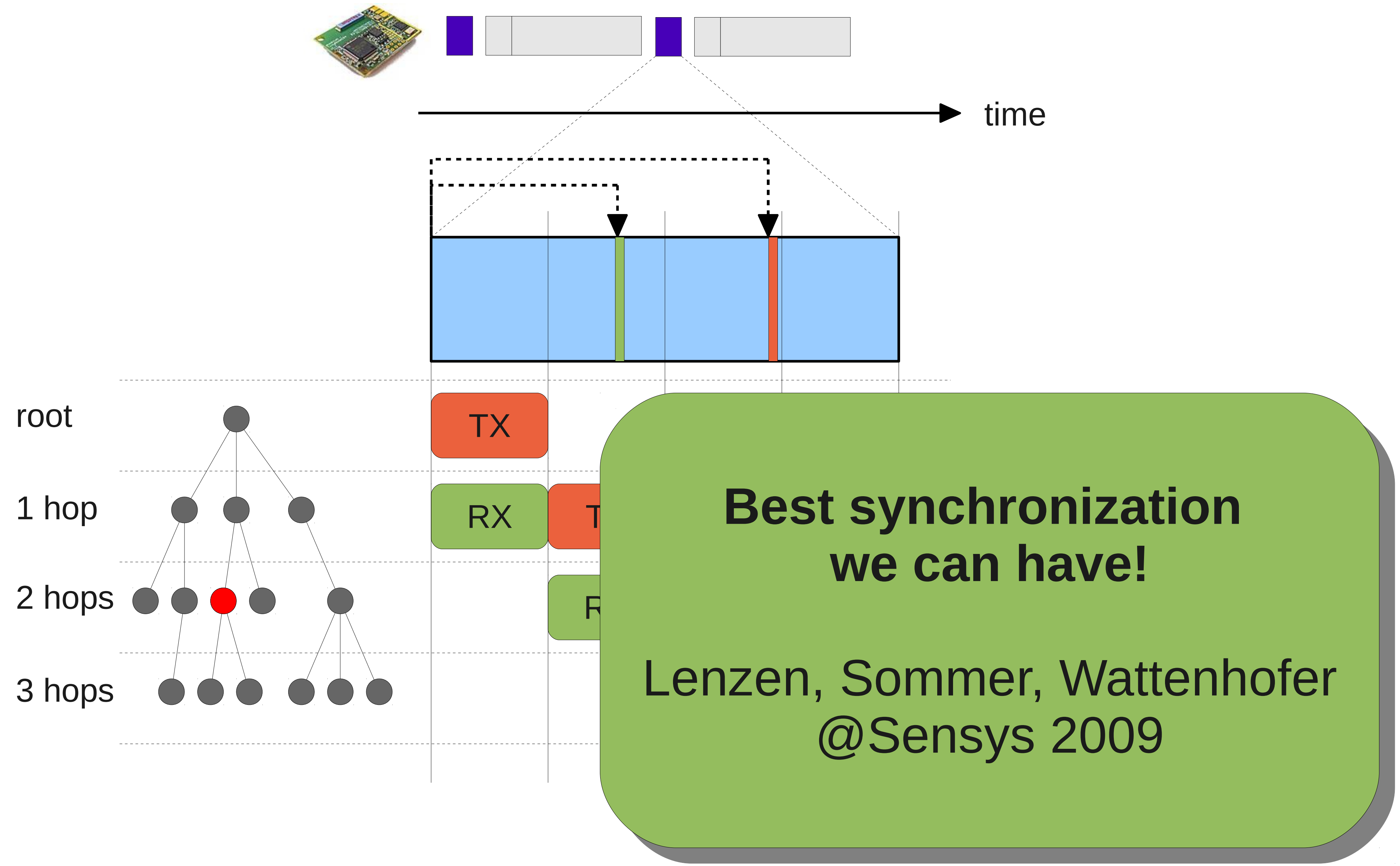
Pipelining



Pipelining



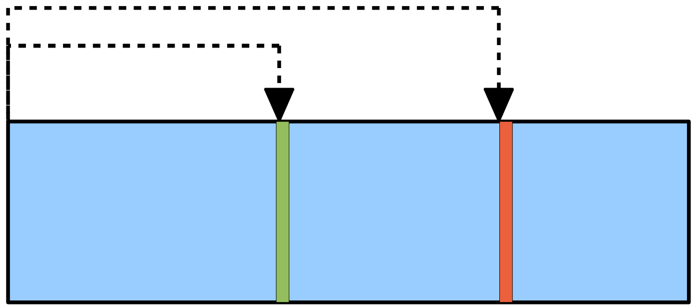
Pipelining



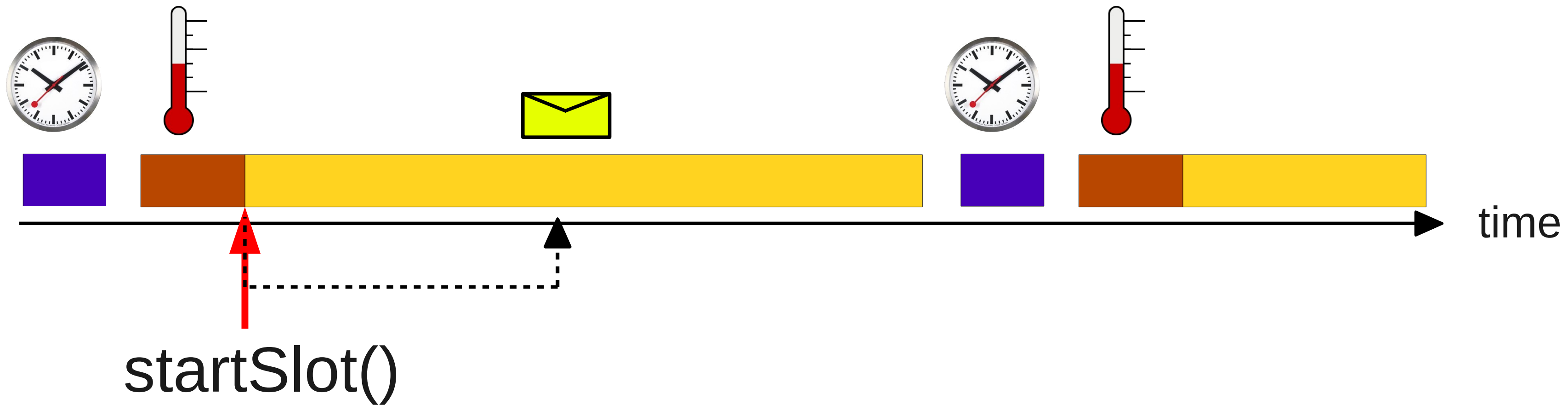
Best synchronization we can have!

Lenzen, Sommer, Wattenhofer @Sensys 2009

Slotted Clock Sync



- Timers not delayed
- Access to radio not blocked
- Transparent to remaining modules:
  - All timers are relative to the time when the current slot started



Examples

- Slotted Clock Synchronization
- **Energy Efficient Alarming**
- Data gathering

Goal

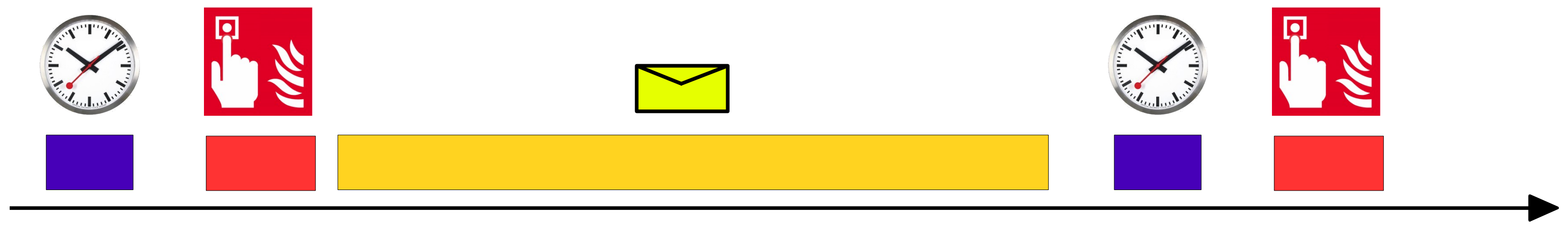
a) Inform the root node about an event
b) Inform all nodes about an event

energy efficient and reliable

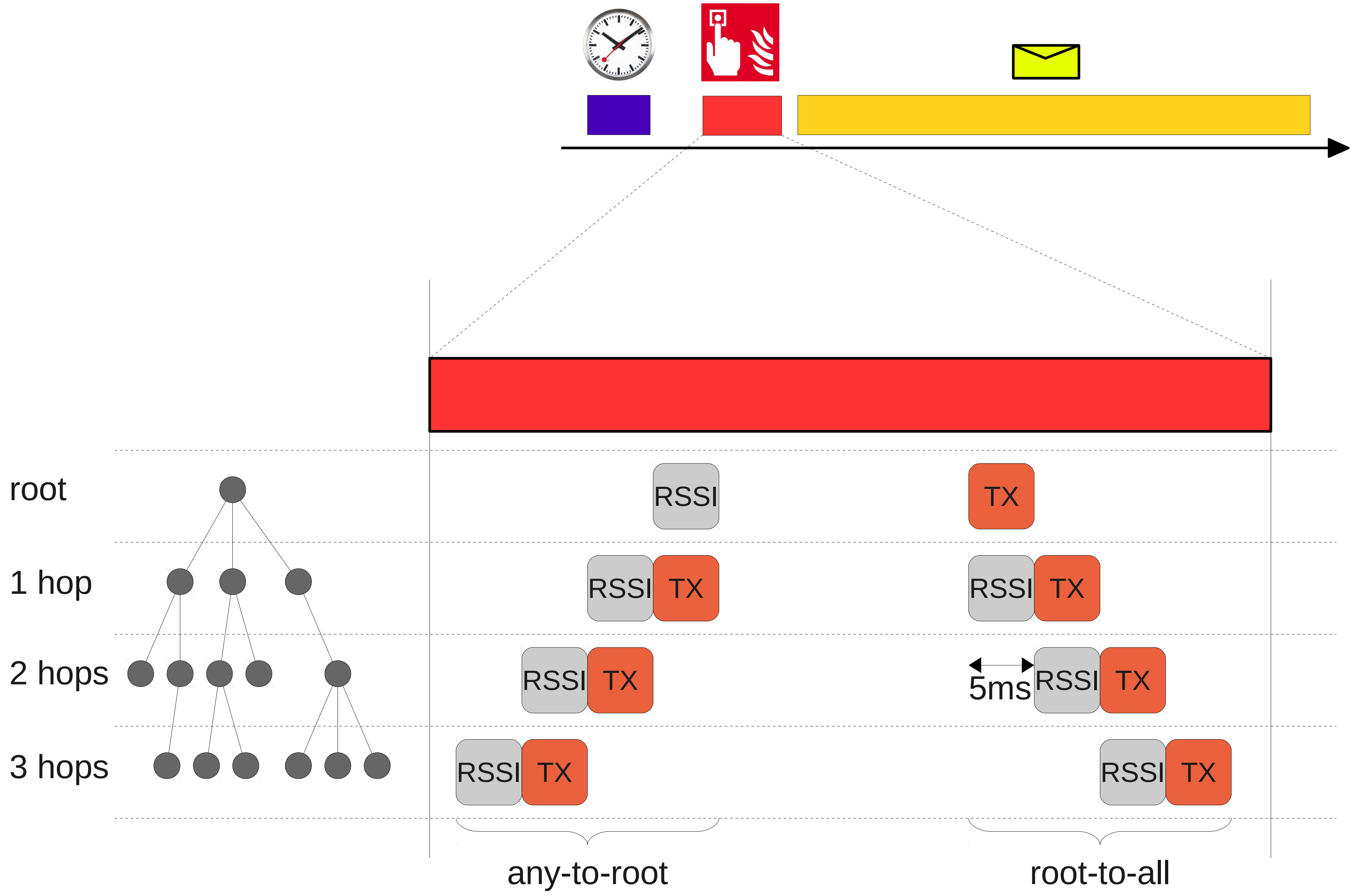
Energy Efficient Alarming

Initial idea: Pipelined RSSI sniffs

- Send a message to transmit an alarm
- Measure the RSSI value to detect an alarm
- Several nodes may indicate an alarm in parallel



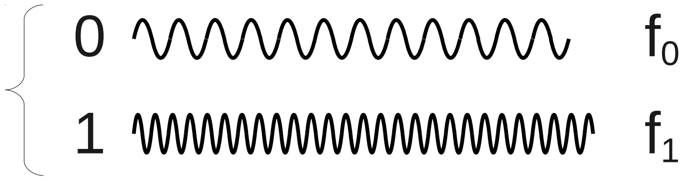
Alarm!



root
1 hop
2 hops
3 hops
RSSI
TX
RSSI
TX
RSSI
TX
RSSI
TX
RSSI
TX
5ms
RSSI
TX
RSSI
TX
any-to-root
root-to-all

False Alarm!

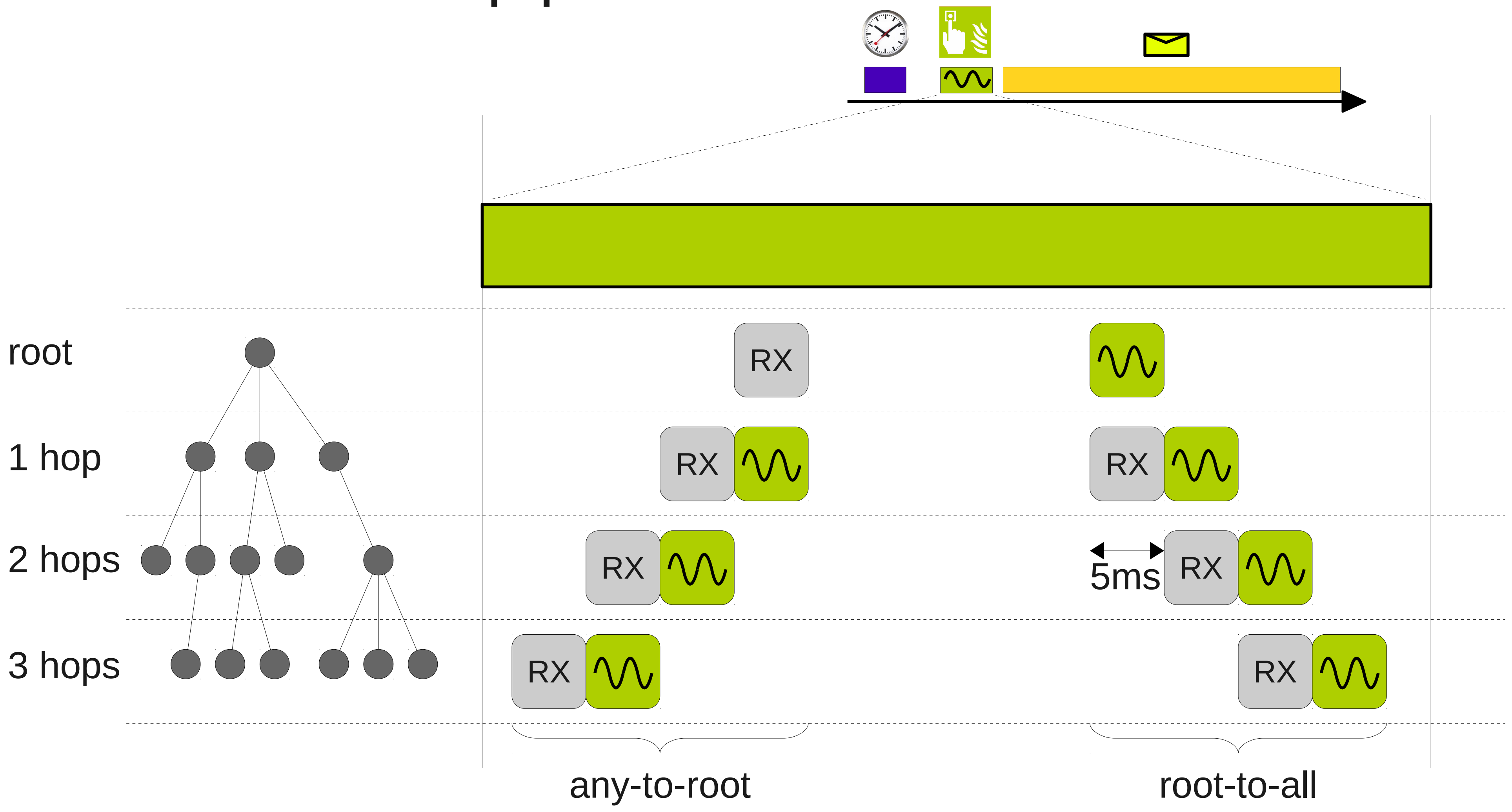
- RSSI works fine – indoors
  - Around 30% false alarms when deployed outdoors
- TinyNode with Semtech XE1205 Radio
  - FSK modulation:



  - Announce an alarm: Send at f_0
  - No alarm: be quiet
  - Detection:
    - No alarm: white noise (50% '1' and 50% '0')
    - Alarm: > 75% '0'
  - Several nodes may indicate an alarm in parallel

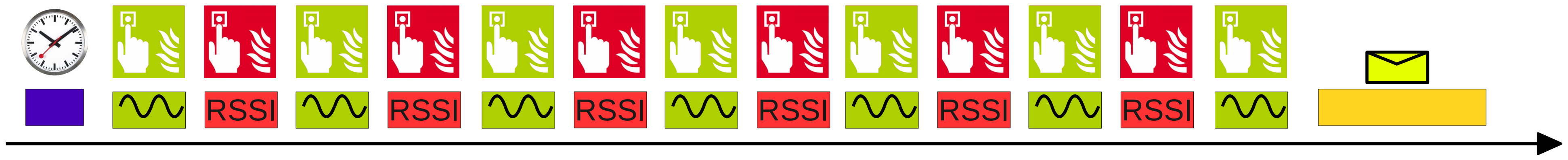
Signaling

- Reuse same pipeline as for RSSI:



Slotted Programming

- How to compare the RSSI and Signaling?



- Slotted Programming = Modularity
  - Easy reuse of software modules
- Energy Efficient: Tight pipelining
  - No delays for wakeup, guaranteed access to radio
- False Alarms: 30% RSSI vs 1% Signaling

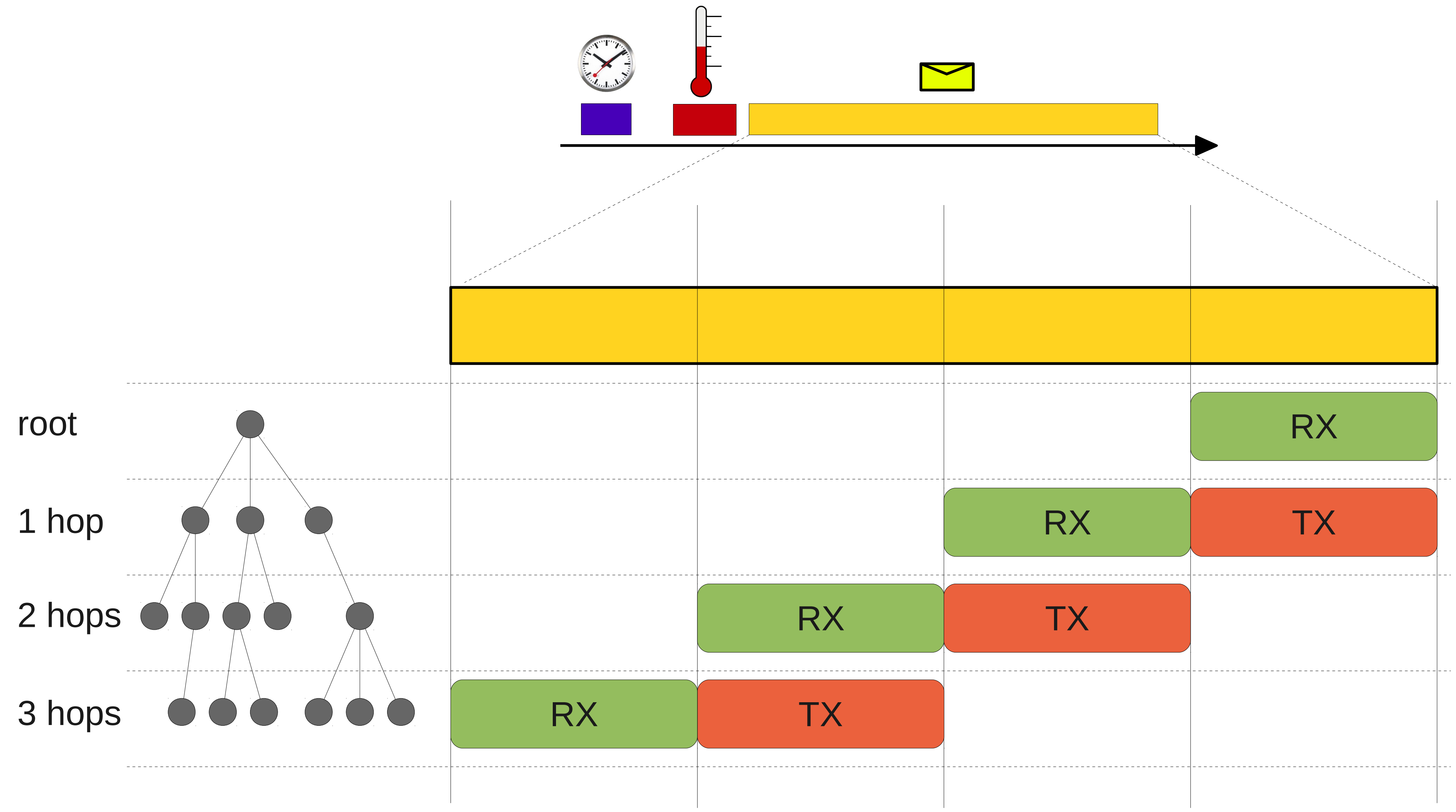
Examples

- Slotted Clock Synchronization
- Energy Efficient Alarming
- **Data gathering**

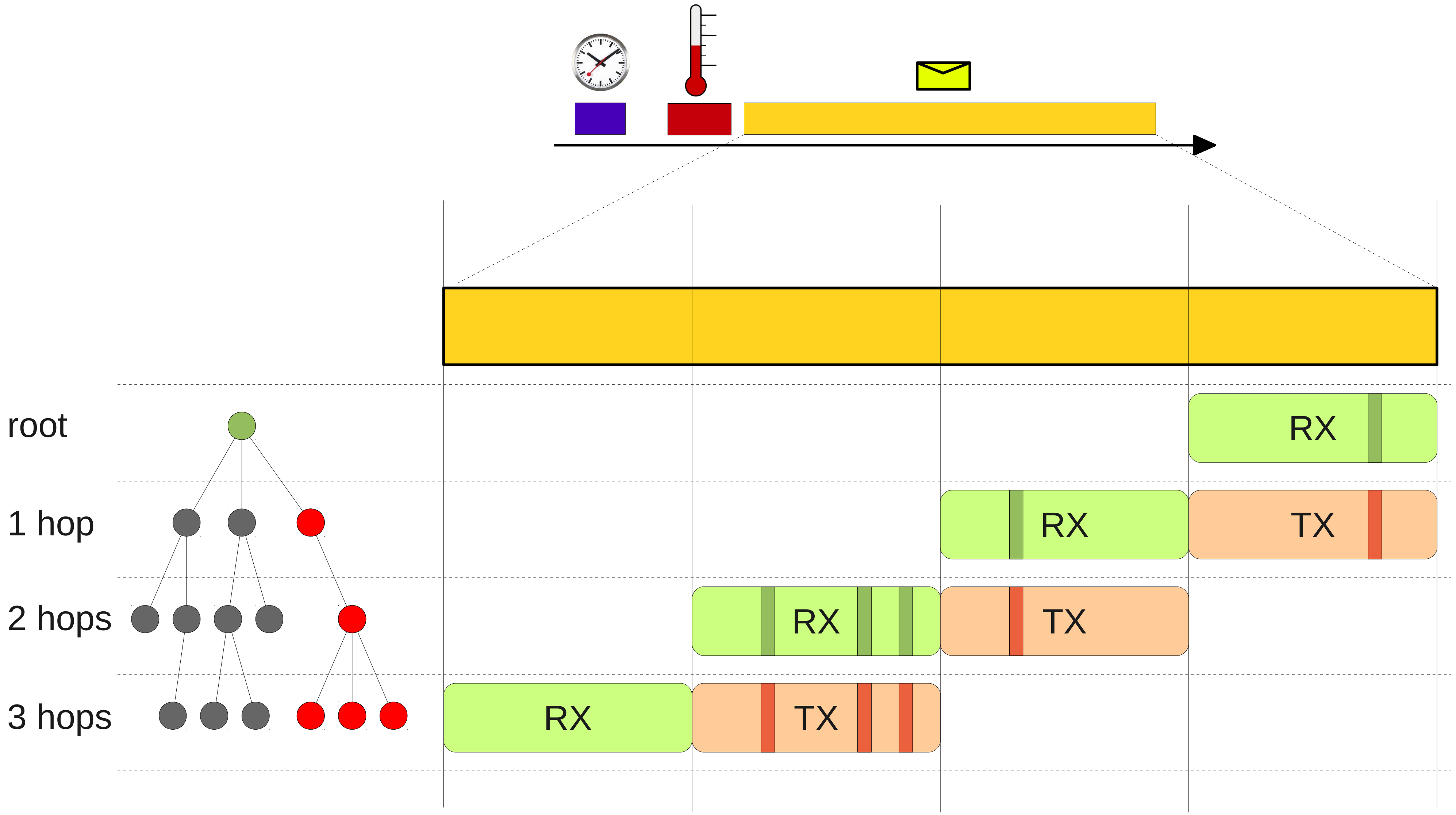
Goal

Collect sensor readings at a base station

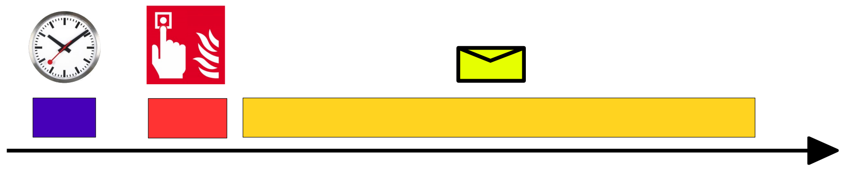
Slotted Data Gathering



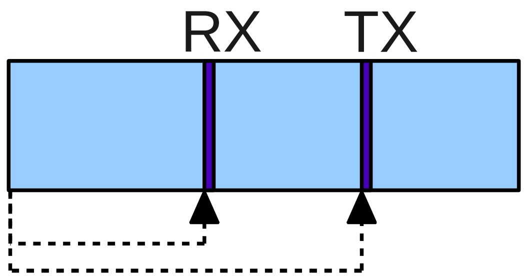
Slotted Data Gathering



Slotted Programming - Recap



- Simple time division
- Modularity
- No delays, guaranteed access to resources
- Energy efficient applications
  - Tight scheduling & wakeup patterns



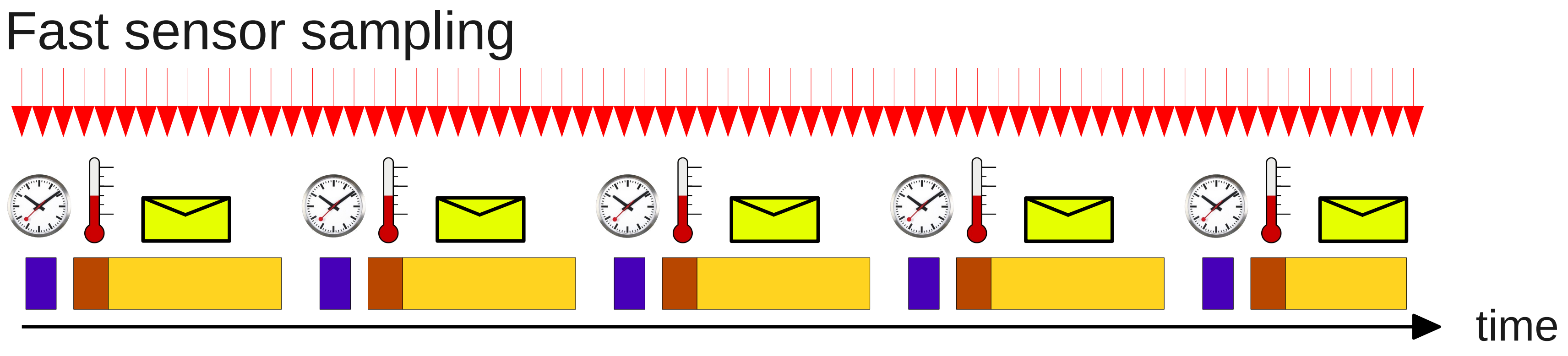
- Transparent Clock Sync
- Simplified software structure
  - Each module can be analyzed independently

Thank You!

slotos – an extension to TinyOS that supports slotted programming will be available online soon.

Slotted Programming For Everything?

- External asynchronous events
- Fast sampling
- Multitasking required?


Fast sensor sampling
time