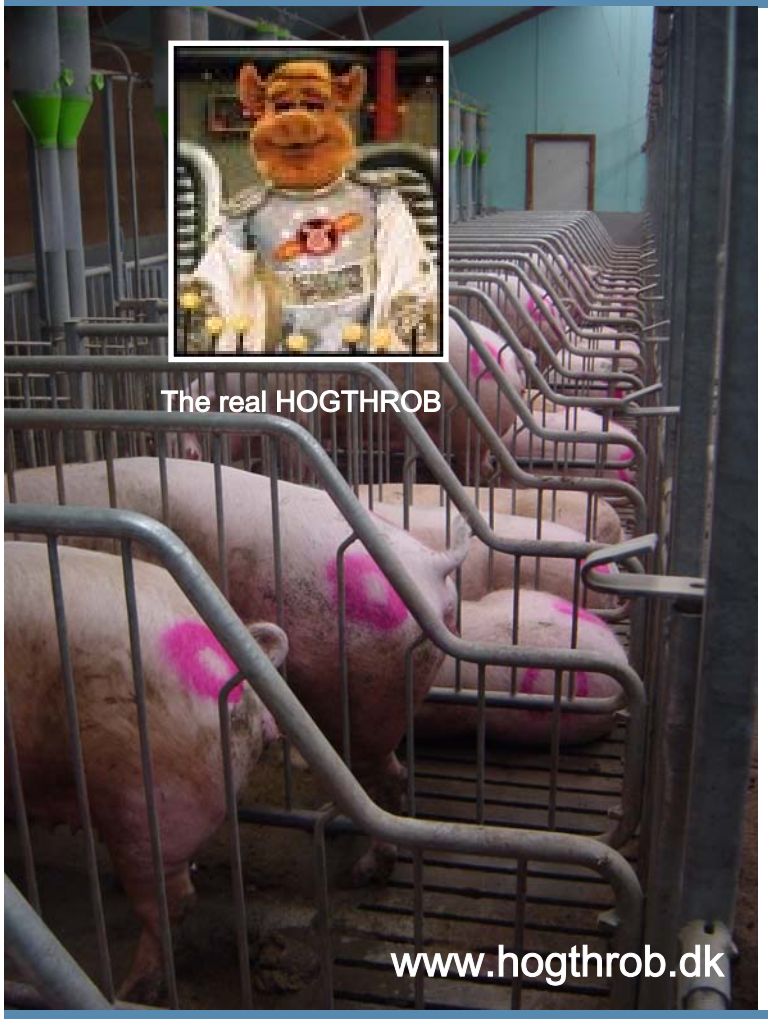
System-level Modeling for Wireless Sensor Networks

Jan Madsen
Kashif Virk, Knud Hansen

Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads, Building 321
DK2800 Lyngby, Denmark
jan@imm.dtu.dk

Funded by Hogthrob (STVF 2059-03-0027)

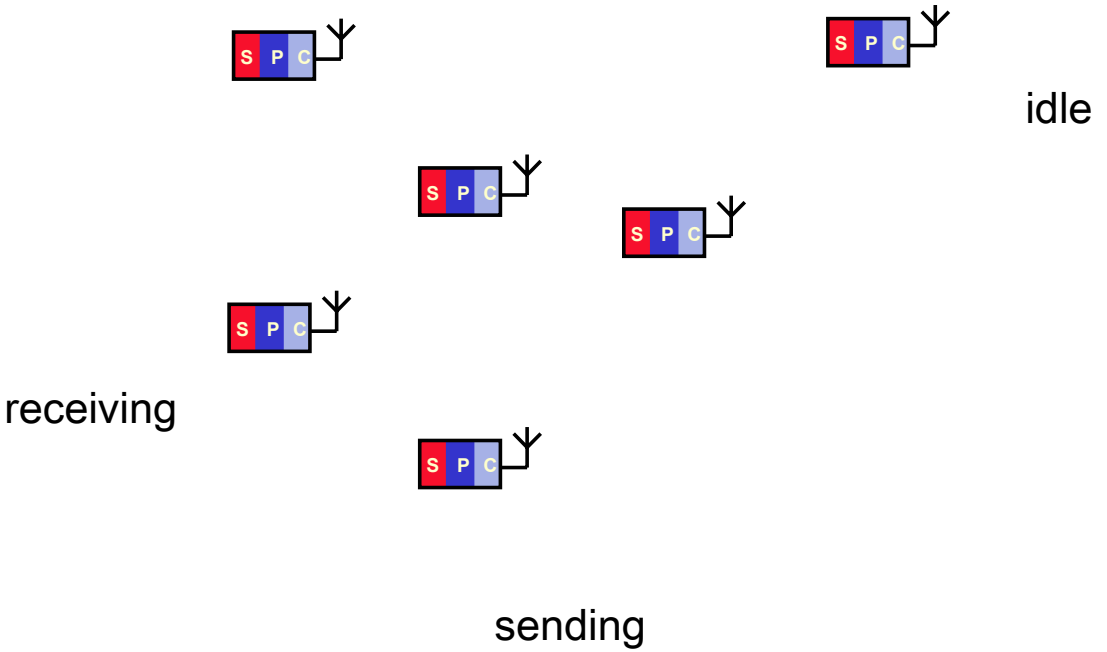
Sensor networks?



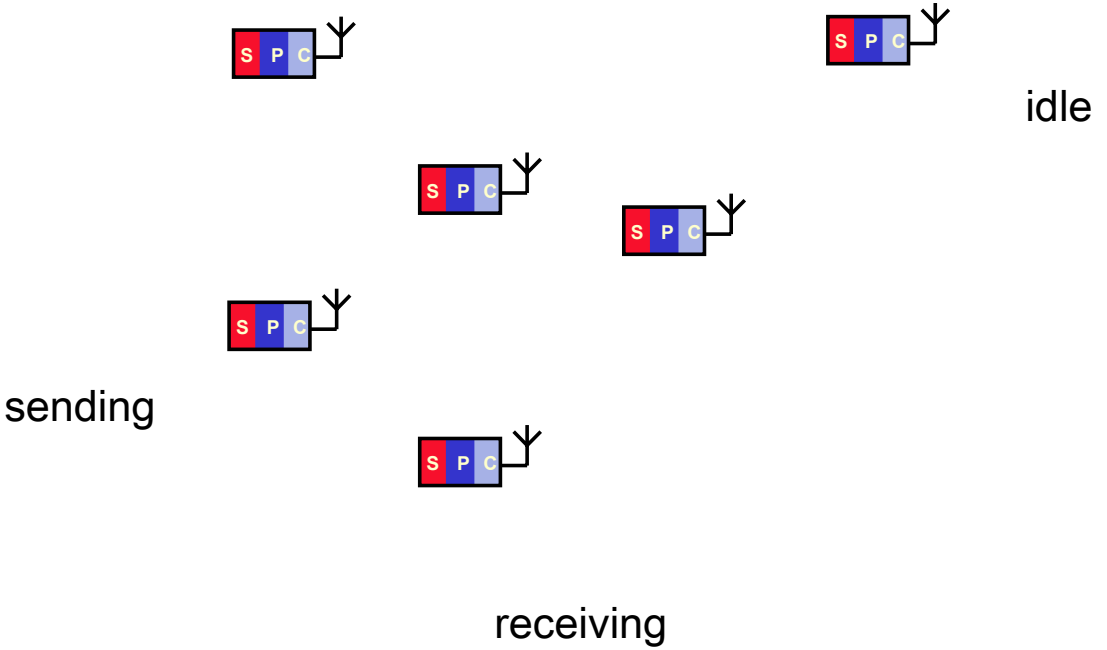
The Hogthrob project

- Developing a sensor network infrastructure for sow monitoring
- Functionalities
  - Tracking
  - Detecting *heat* period
  - …
- Low Cost (~1 €)
- Low Energy (2 years lifetime)
- Consortium:
  - DTU, DIKU, KVL
  - National Committee for Pig Production
  - IO Technologies

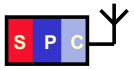
Sensor networks

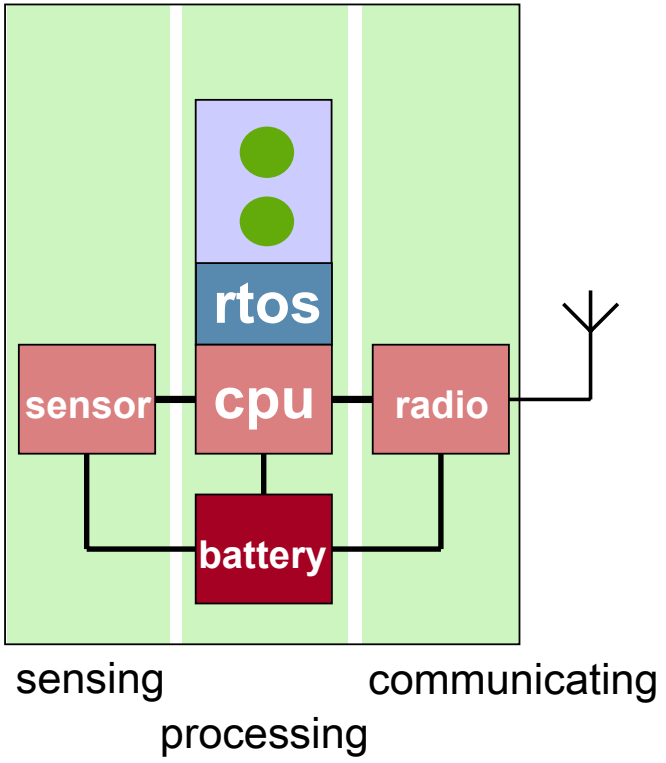


Sensor networks



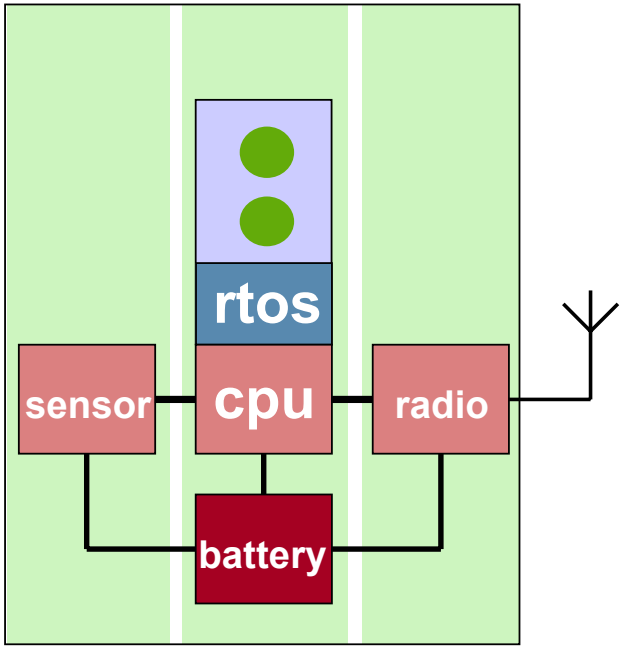
Sensor node


S
P
C

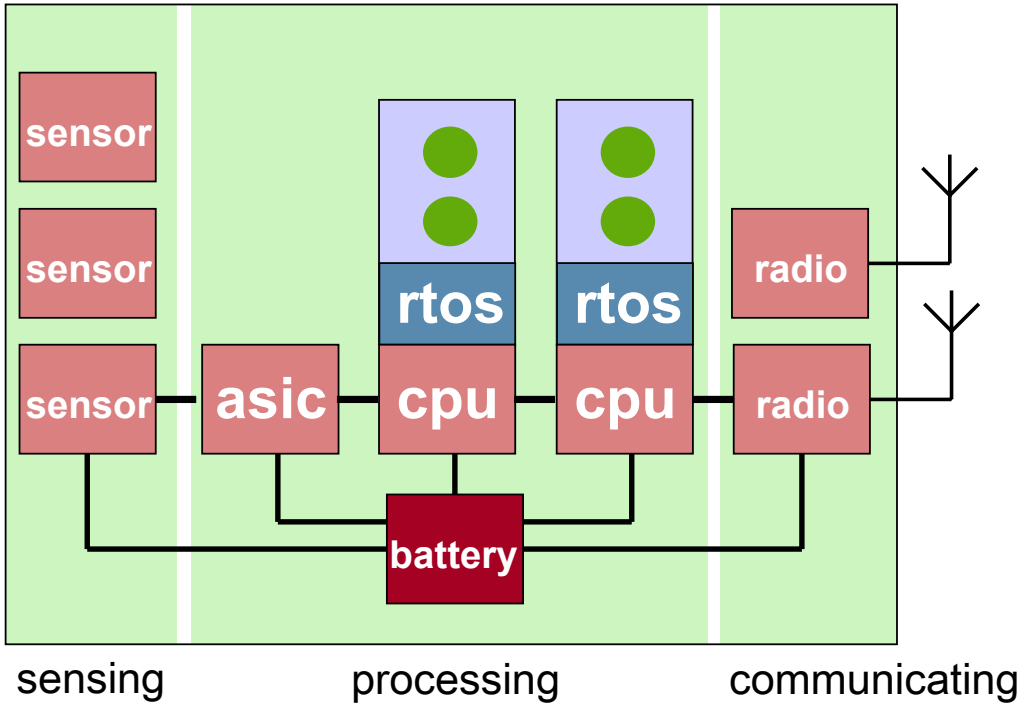

rtos
sensor
cpu
radio
battery
sensing
processing
communicating

Sensor node

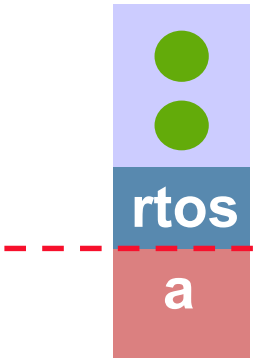
- Ultra low energy
- Low flexibility
- Ultra low cost (1€)
- Small size (1..10 Mtr)
- Low clock frequency
- CPU/DSP and RF dominated
- Limited memory
- Hardware/software codesign



Sensor node design



Sensor node: Uni-processor ...



Framework to experiment with different RTOS strategies

Focus on analysis of **timing**, **energy** and resource sharing

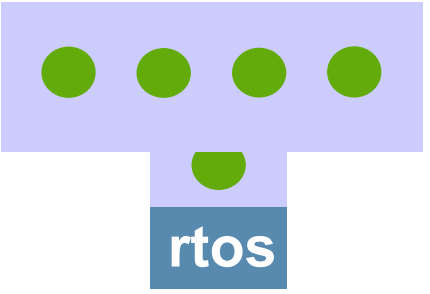
Abstract software model, i.e. no behavior/functionality

Easy to create tasks and implement RTOS models

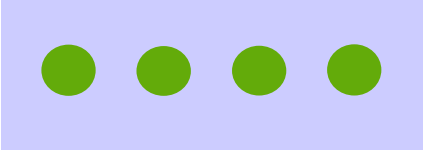
Based on **SystemC**

System model

System model

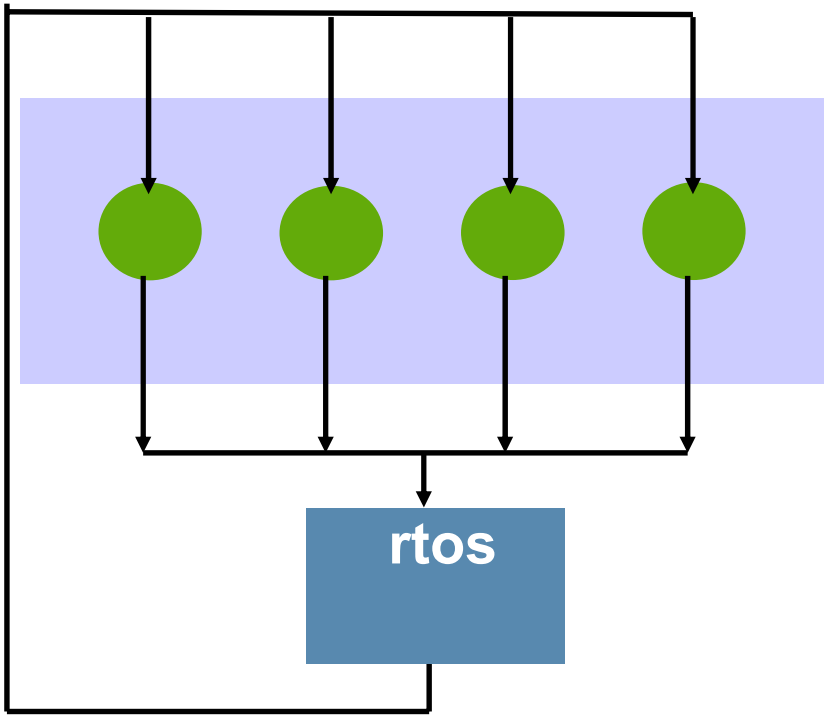


System model



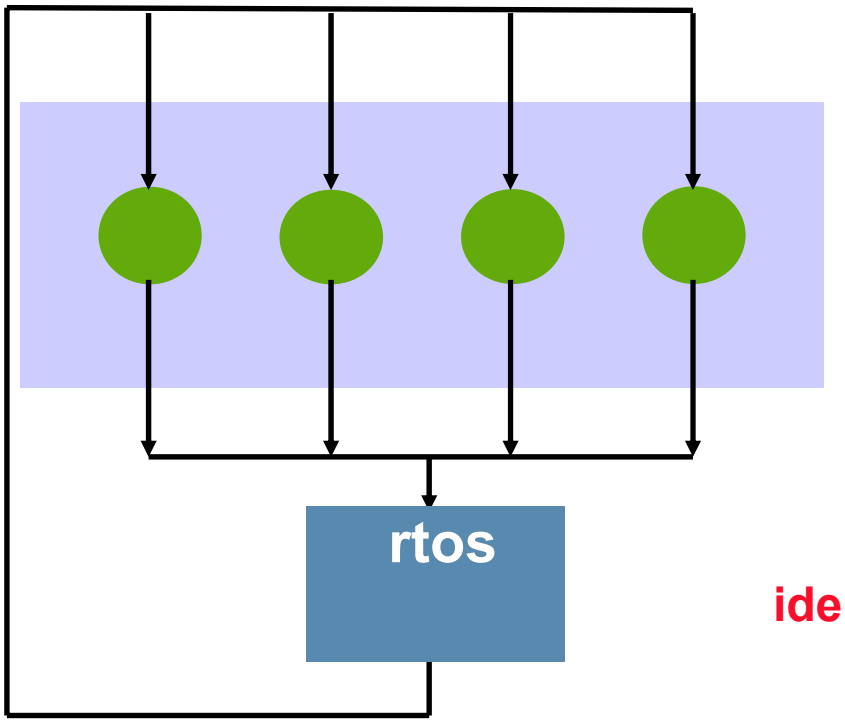
rtos

System model



- Task messages:
  - **`ready`**
  - **`finished`**
- RTOS commands:
  - **`run`**
  - **`preempt`**
  - **`Resume`**

System model - SystemC




```
pa = new
   task("task_a",1,50,3,12,0,ready);
registerTask(pa);

pb = new
   task("task_b",2,40,2,10,0,ready);
registerTask(pb);

pc = new
   task("task_c",3,30,1,10,0,ready);
registerTask(pc);
```

identifier

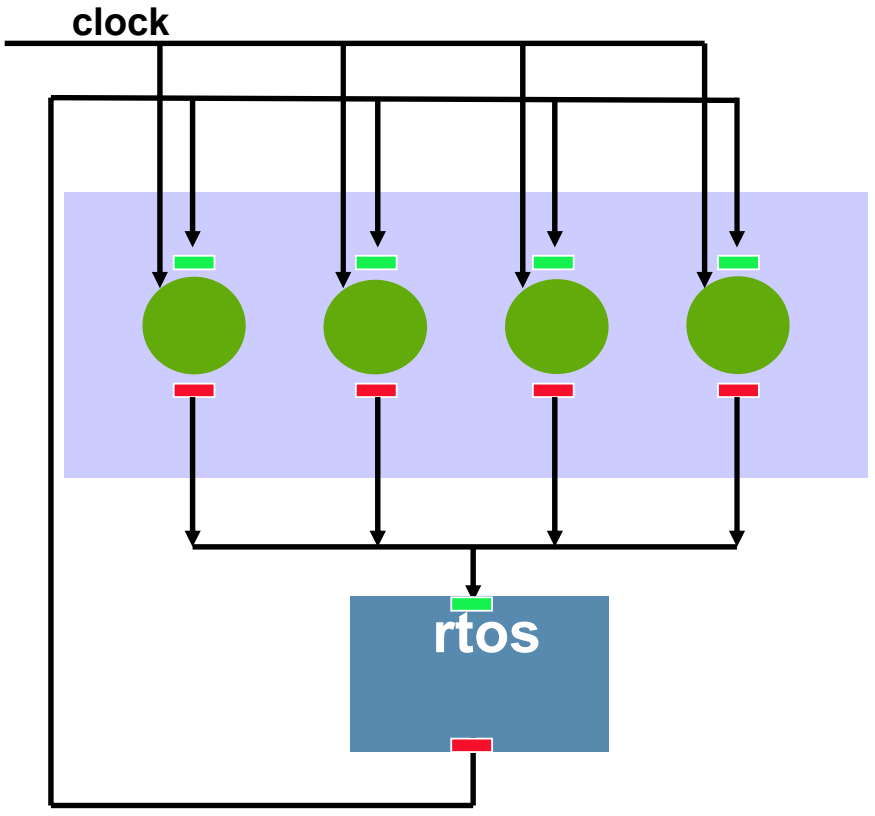
period

priority

WCET

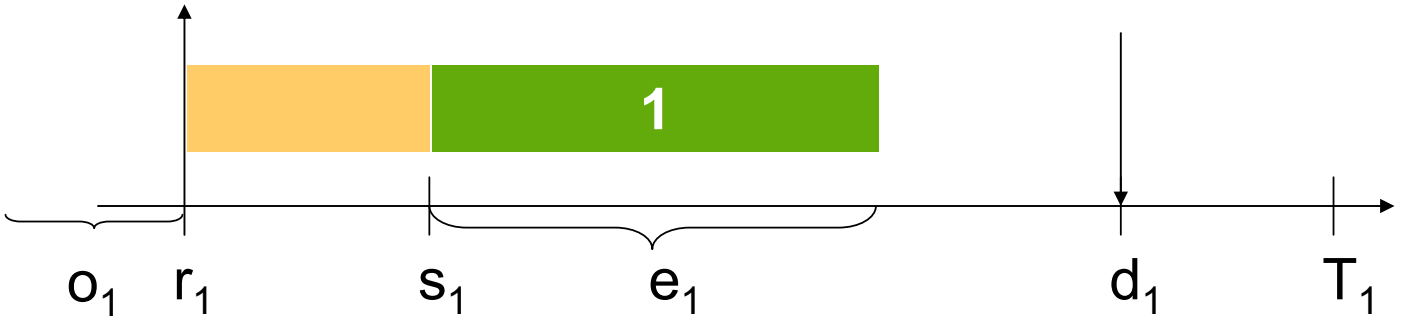
offset

Link model



- Aim: Adding tasks without having to create seperate communication links
- Uses the SystemC master-slave library
- If two tasks send a message at the same time – they are executed in sequence, but in undefined order
- Global ”clock” is used to keep track of time

Task model



r_1 = time at which task becomes ***released (or active)***

s_1 = time at which task ***starts*** its execution

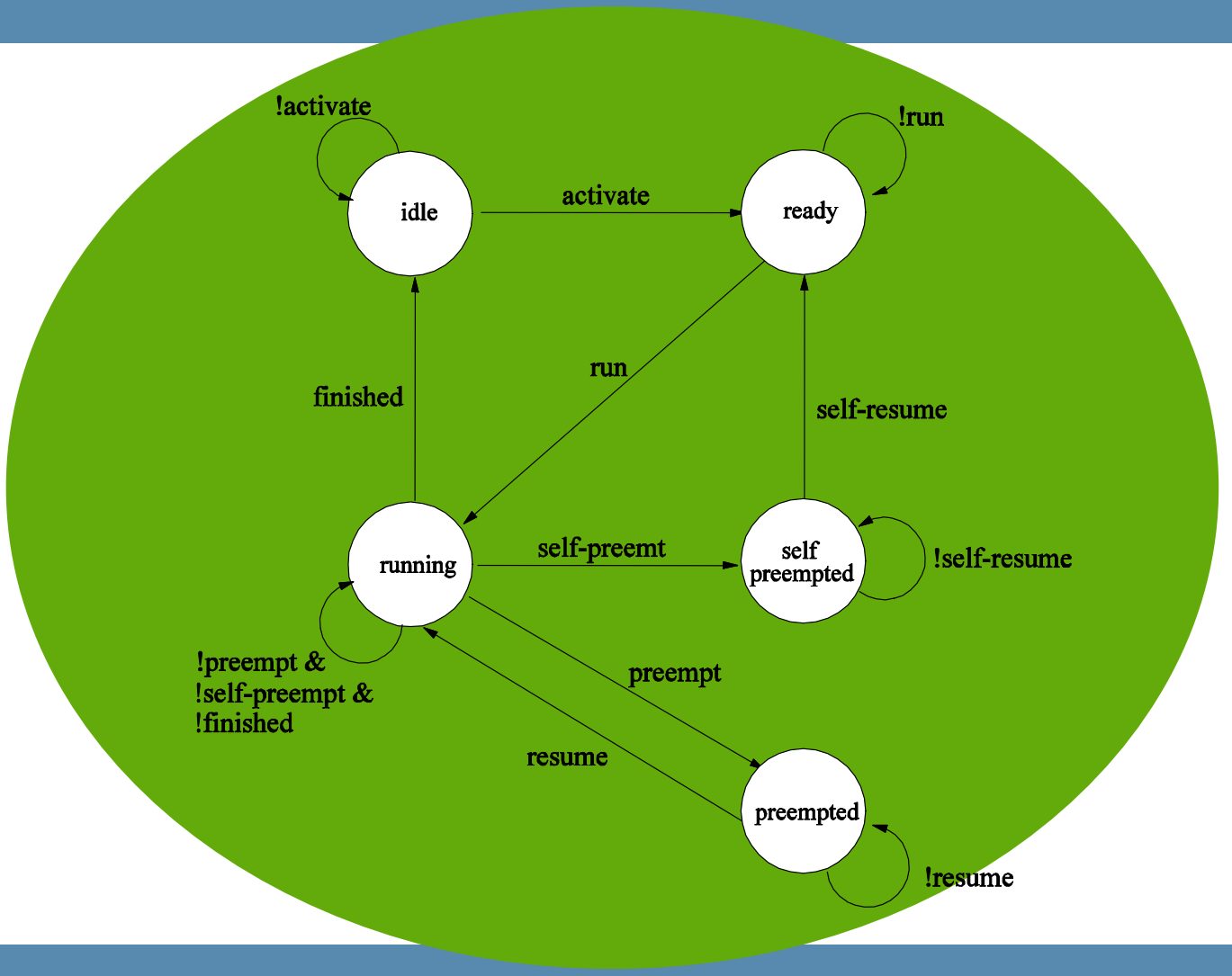
e_1 = worst case ***execution*** time (WCET)

d_1 = ***deadline***, task should complete before this!

T_1 = ***period***, minimum time between task releases

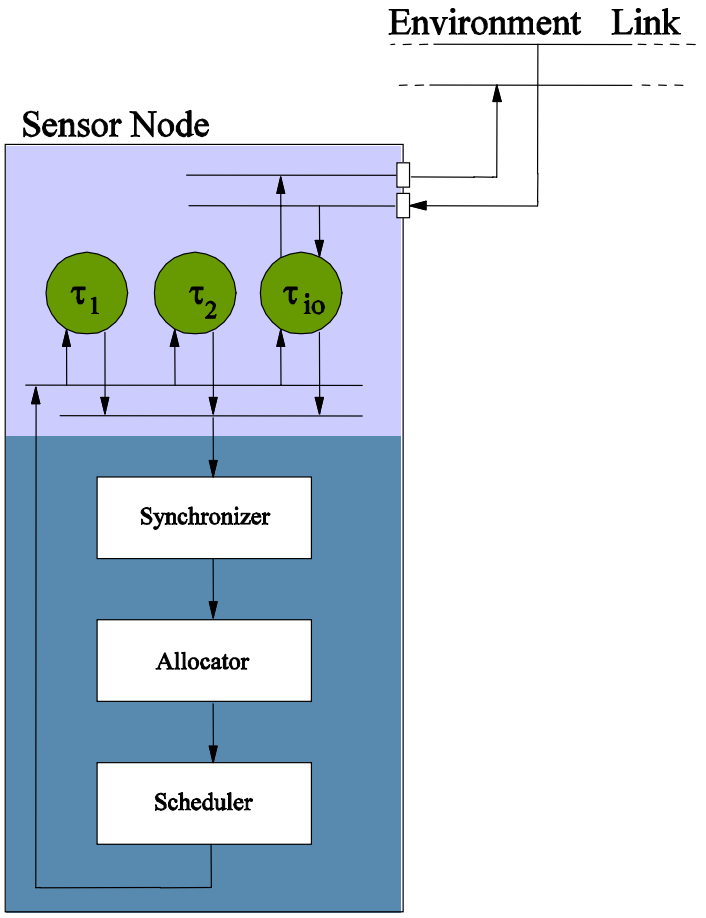
o_1 = ***offset (or phase)*** for first release

Task model

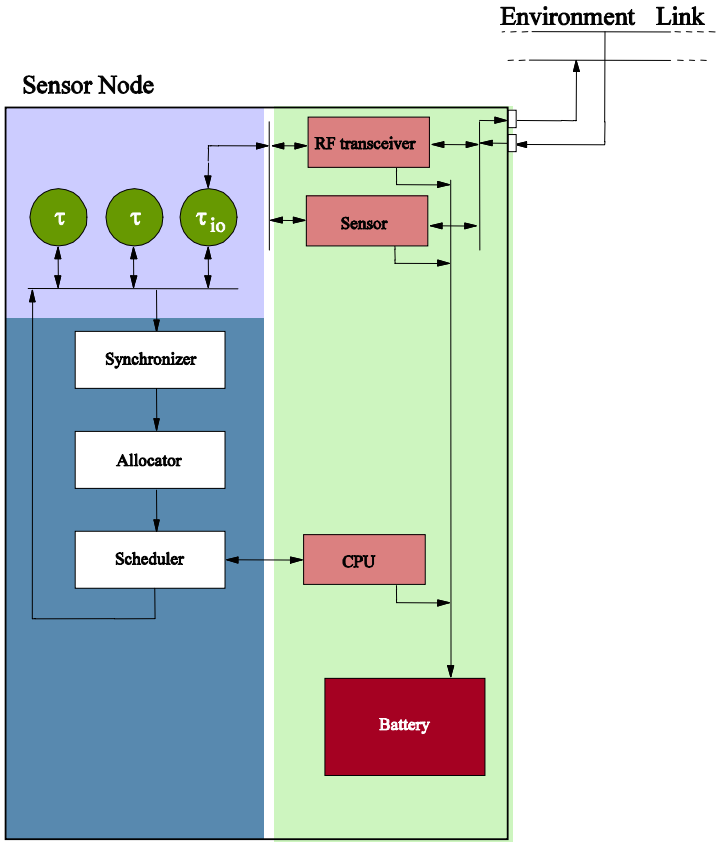

!activate
idle
activate
ready
!run
finished
run
self-resume
running
self-preemt
self
preempted
!self-resume
!preempt &
!self-preempt &
!finished
preempt
resume
preempted
!resume

Sensor network model

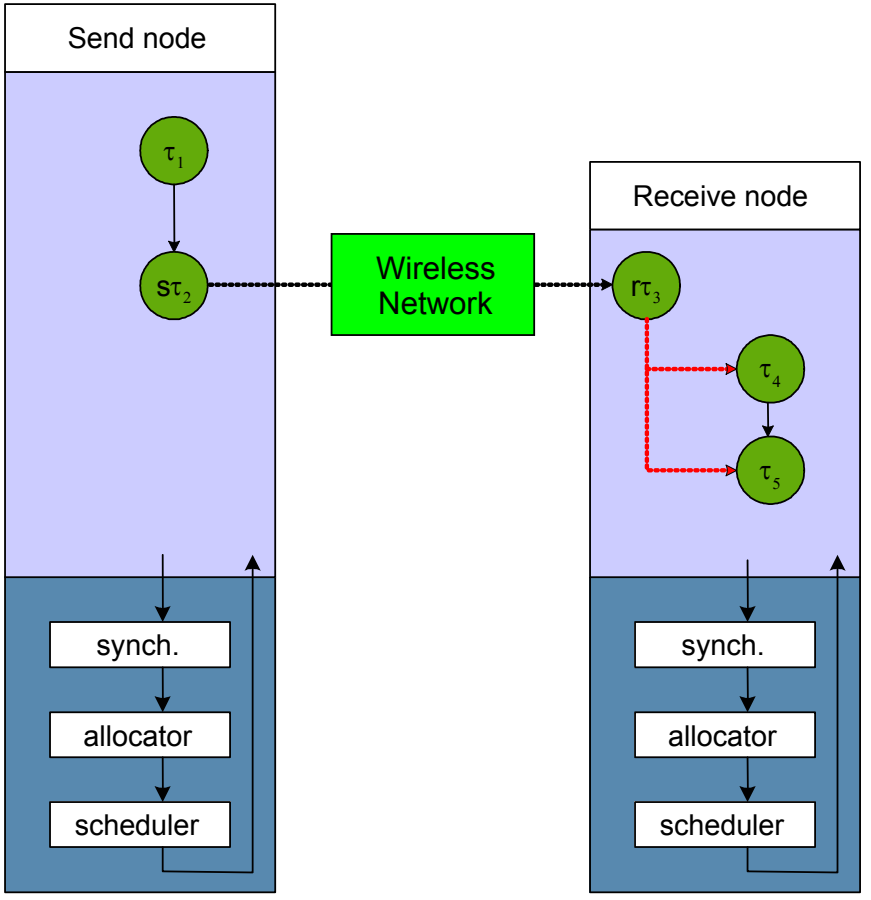
Sensor node model



Energy modeling



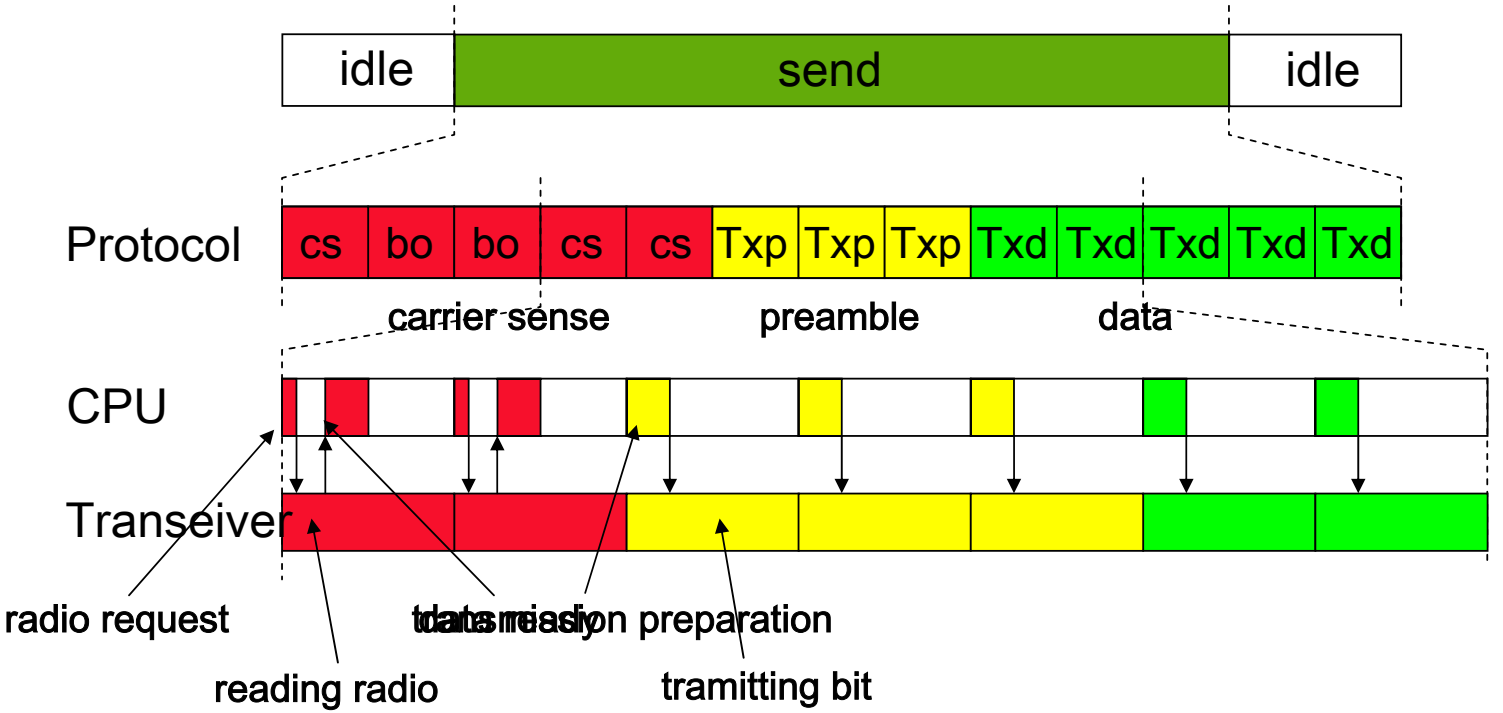
Communication example



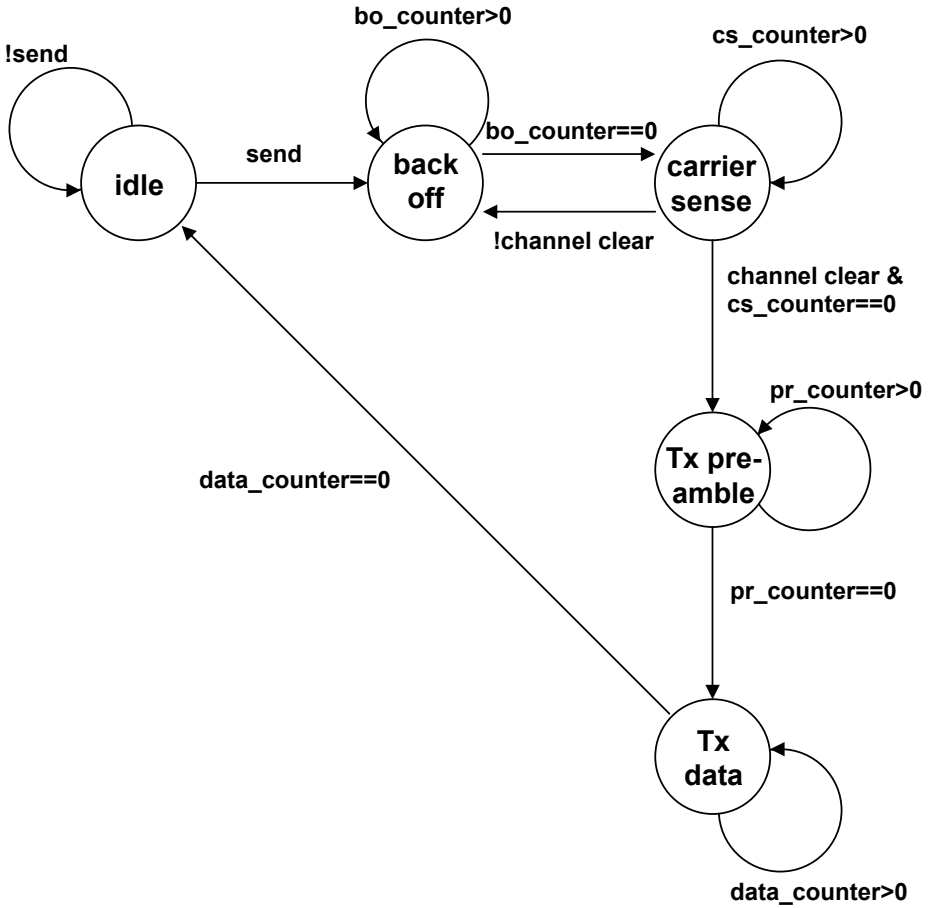
Modeling radio communication

Modeling the CSMA protocol

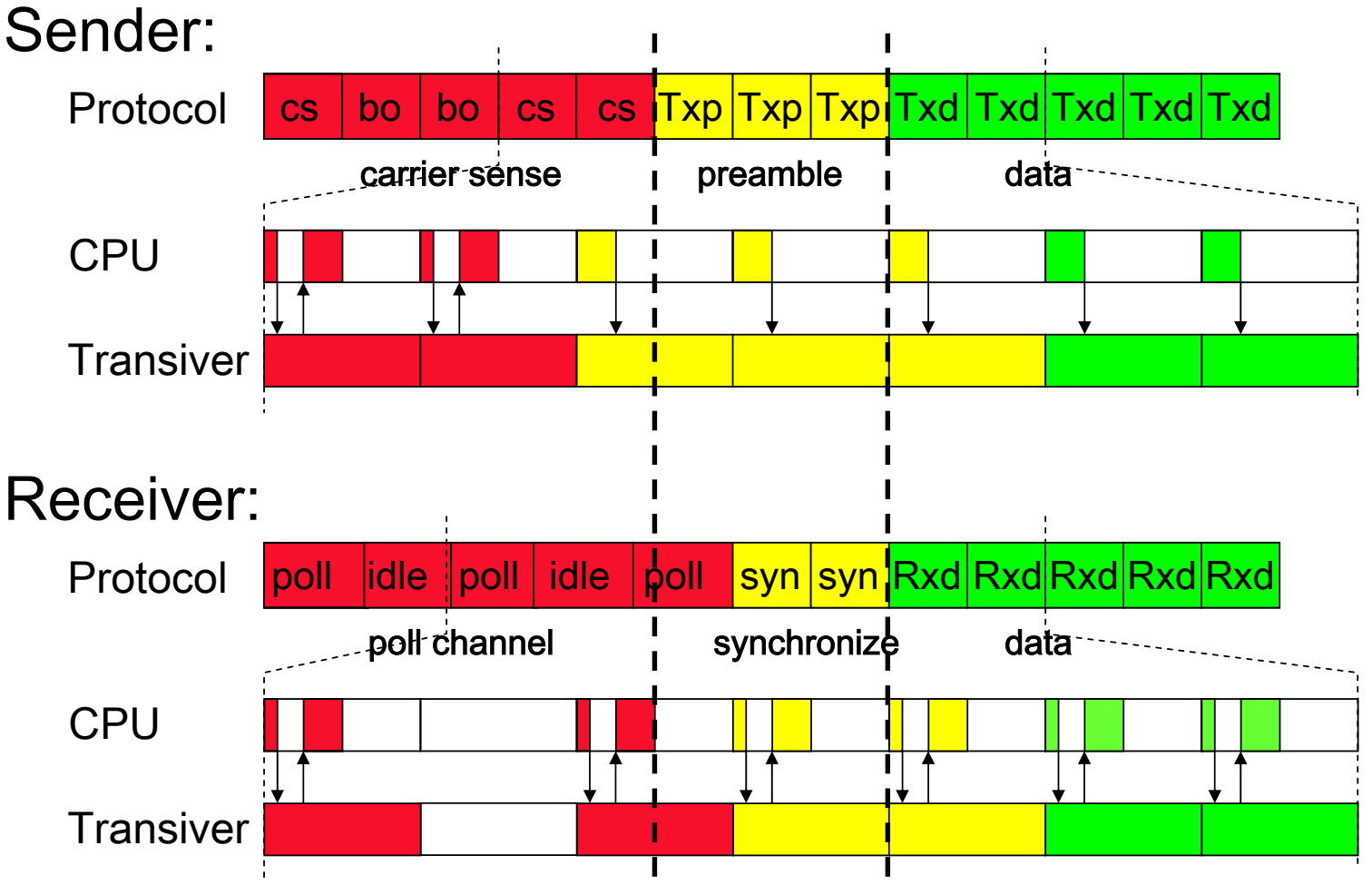
Sender:



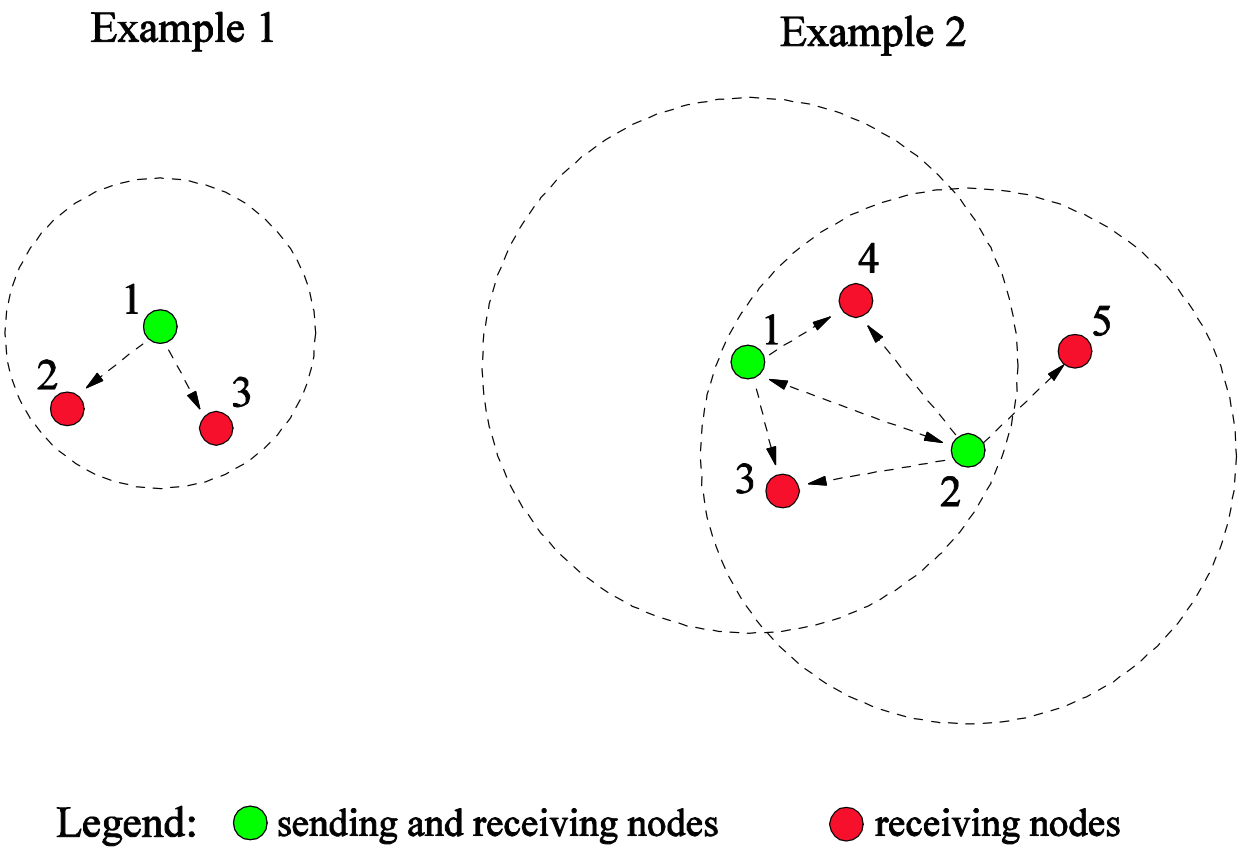
CSMA Protocol for sending



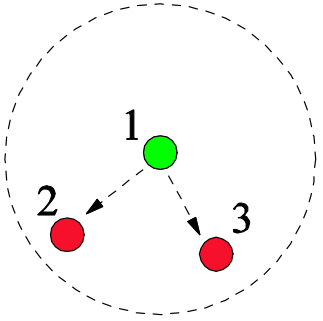
Modeling radio communication



Sensor network example



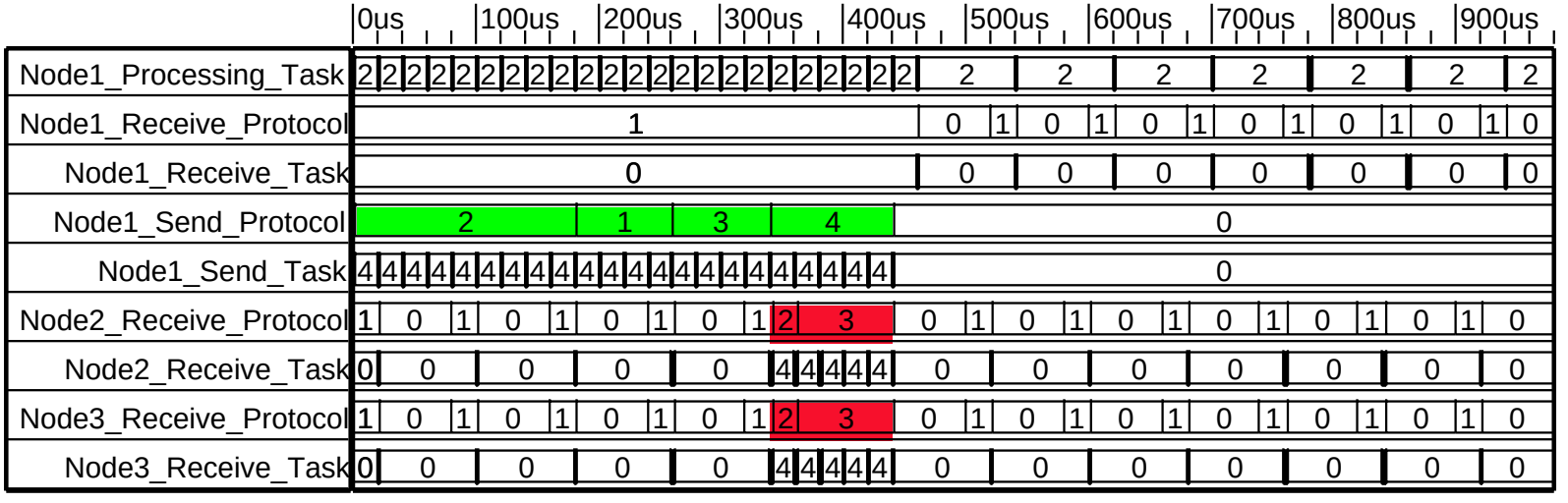
Example 1: Simple broadcast



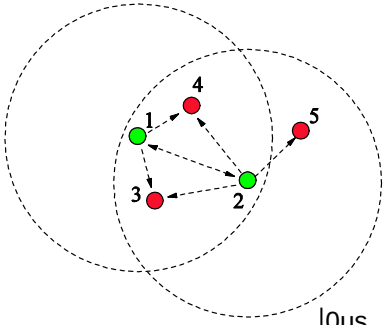
Sending task
0 = idle
1 = carrier sensing
2 = back-off
3 = transmit preamble
4 = transmit data

Receiving task
0 = idle
1 = polling
2 = synchronize
3 = receive data

Application task
0 = idle
1 = ready
2 = running
3 = preempted
4 = self-preempted



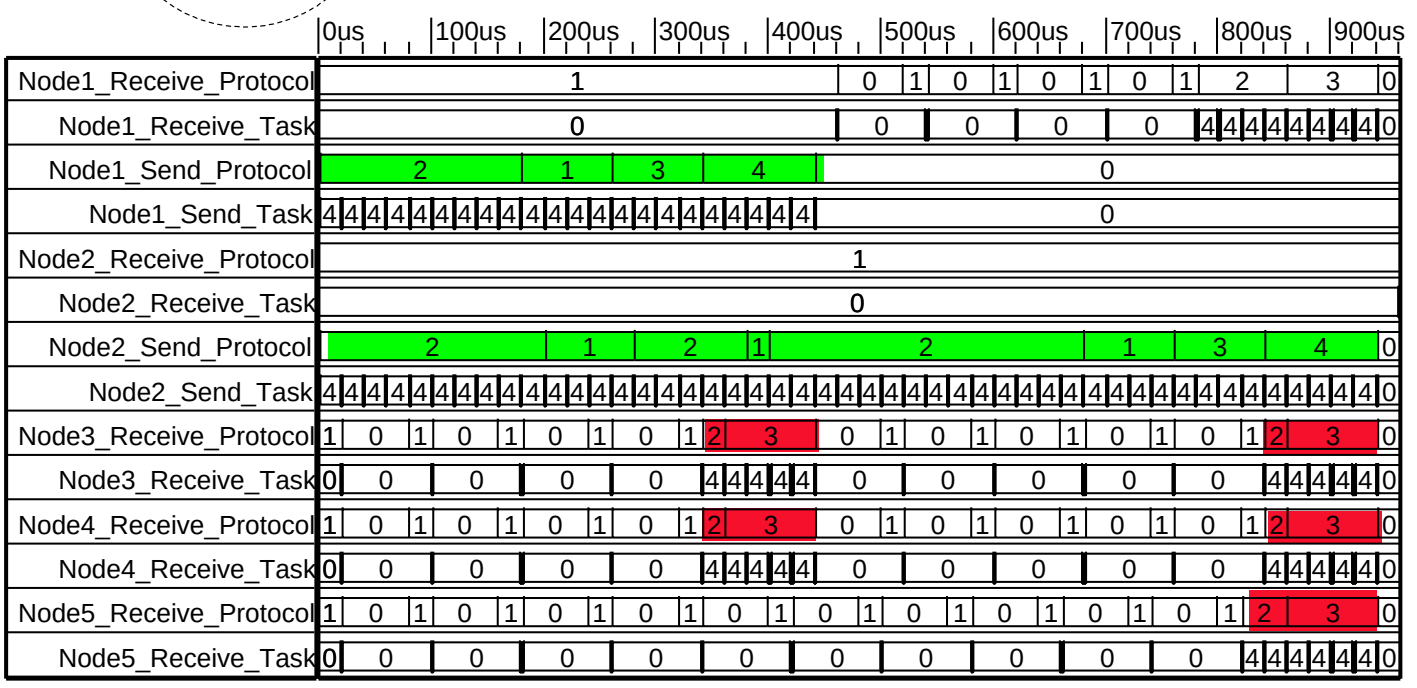
Example 2: Radio interference



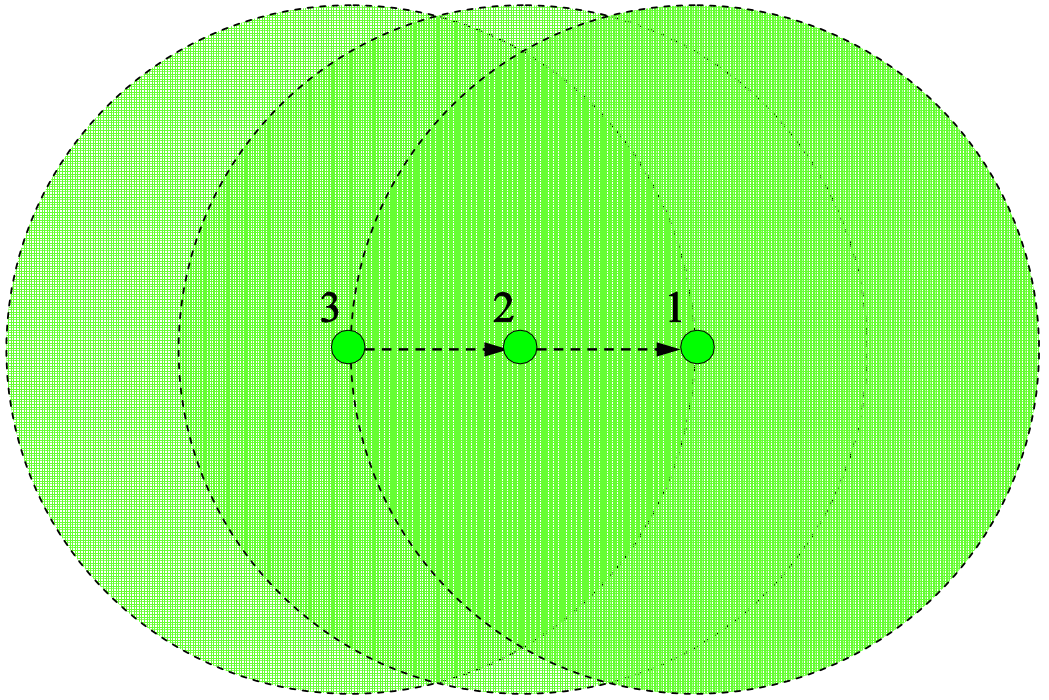
Sending task
0 = idle
1 = carrier sensing
2 = back-off
3 = transmit preamble
4 = transmit data

Receiving task
0 = idle
1 = polling
2 = synchronize
3 = receive data

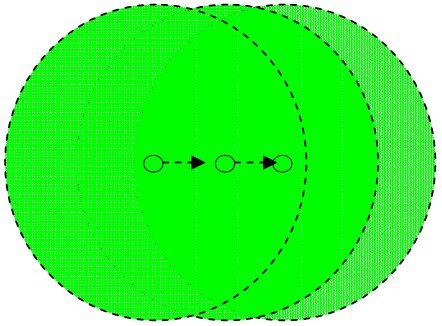
Application task
0 = idle
1 = ready
2 = running
3 = preempted
4 = self-preempted



Example 3: Network routing



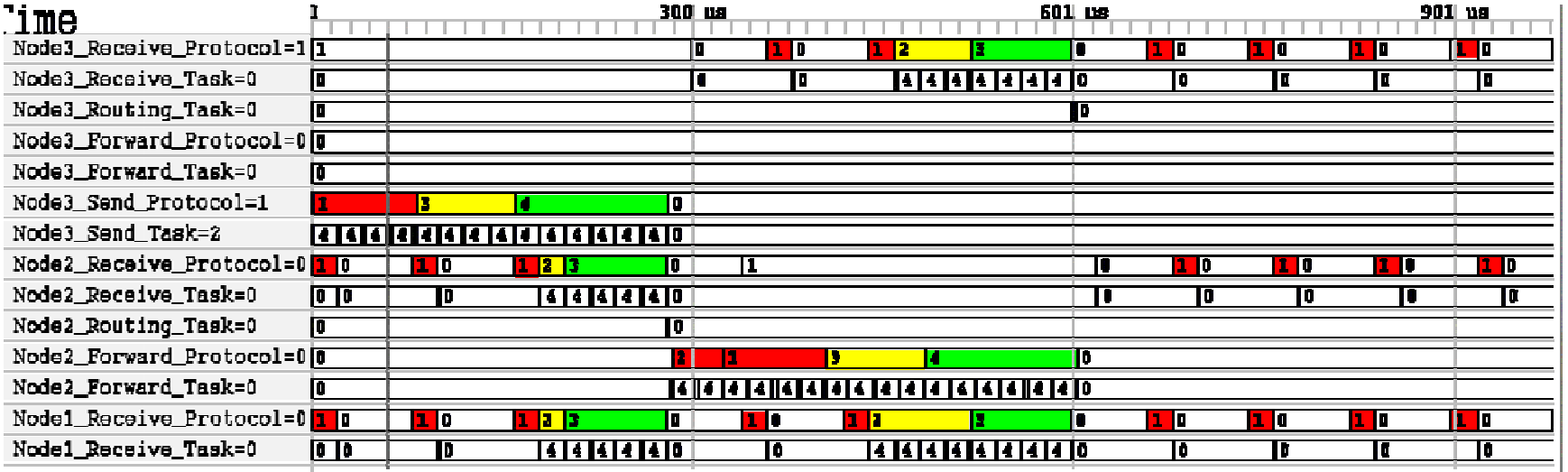
Example 3: Routing



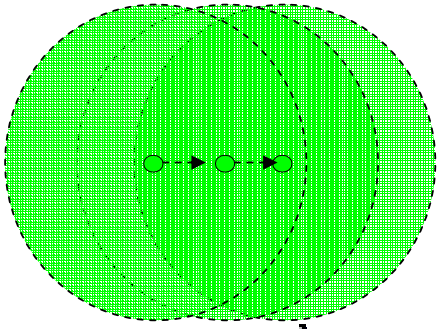
Sending task
0 = idle
1 = carrier sensing
2 = back-off
3 = transmit preamble
4 = transmit data

Receiving task
0 = idle
1 = polling
2 = synchronize
3 = receive data

Application task
0 = idle
1 = ready
2 = running
3 = preempted
4 = self-preempted



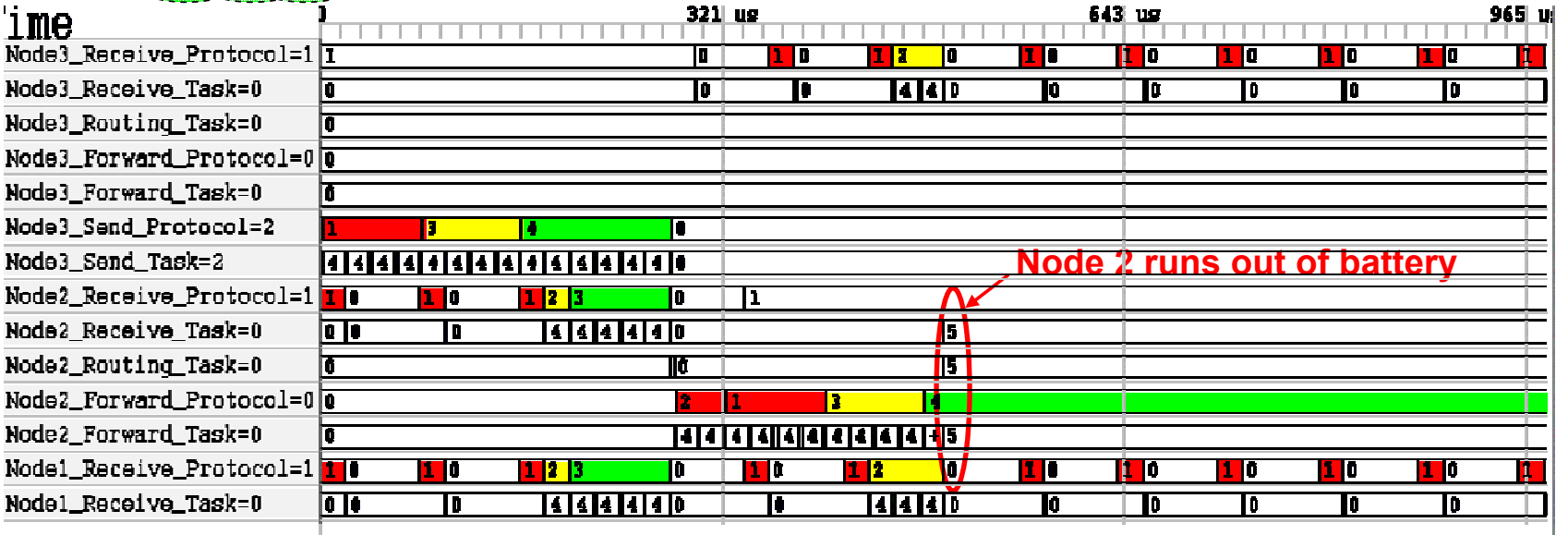
Example 3: Battery shortage



Sending task
0 = idle
1 = carrier sensing
2 = back-off
3 = transmit preamble
4 = transmit data

Receiving task
0 = idle
1 = polling
2 = synchronize
3 = receive data

Application task
0 = idle
1 = ready
2 = running
3 = preempted
4 = self-preempted



Summary

- SystemC based framework to study the dynamic behavior of a sensor network
- Exploring global effects of sensor node design
- Example sensor network based on Mica-nodes and TinyOS from UC Berkeley
- Work in progress
  - Power/energy models for power management
  - Mobile sensor nodes
  - Detailed component models
- To be used in the Hogthrob project

Hogthrob