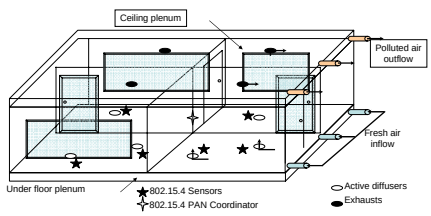
A simple application protocol for embedded controllers over WPAN networks

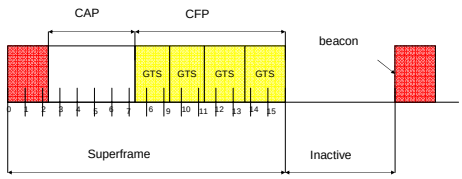
Rémy Feltrin, Stéphane Mocanu
GIPSA-lab

Sample application : under floor air distribution (UFAD)



802.15.4 Real-time features

- GTS transmission : RT but limited number/size
- CAP transmission : no RT CSMA/CA



Data transfer protocol

- Superframe design using “classical” formulas
- Real-time data exchange protocol
  - sensor data sent by slaves using GTS slots
  - actuator data sent by coordinator using CAP (coordinator is the only emitter during CAP)
- 802.15.4 MAC is not modified

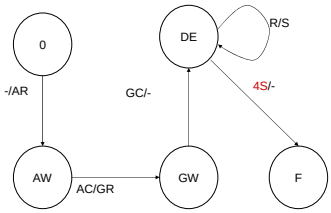
GTS transmissions issues

Slaves are supposed to be sporadic, therefore :

- GTS slots expire after 4 missed supertrames
- slaves cannot observe 4 missed slots
- blocking can be expected

Slaves are supposed to “miss” GTS only when going in power save mode, the reconnect

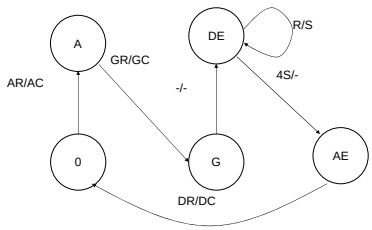
Original slave behavior model



Slave generated events:
AR – association req
GR – GTS request
DR – disassociation req
R/S – data receive/send

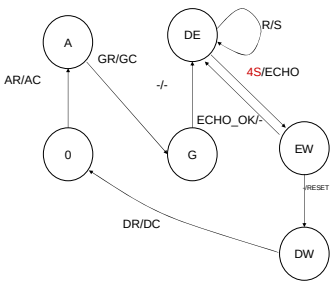
Environment generated event:
4S – four GTS missed

Original coordinator behavior model



Coordinator generated events:
AC – association confirm
GC – GTS confirm
DC – disassociation conf
R/S – data receive/send
4S – four GTS missed

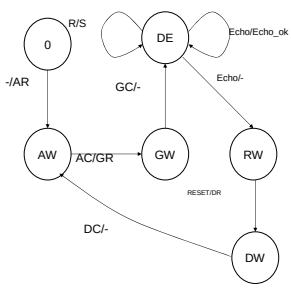
Controlled coordinator



Coordinator control events:
Echo (generated on 4S)
RESET : force disassociation

Controller will simply force slave reassociation after 4 missed GTS

Fast GTS recovery on server side



Slave control events:
Echo_OK (alive)

Controller mechanics

- on 4S event coordinator generates ECHO
- if ECHO_OK is received the GTS is renewed
- else RESET is generated
- on RESET, slaves will disconnect then reconnect
- 802.15.4 MAC is not modified

Concluding remarks

Two parts of the application protocol :

- Control oriented data exchange (mix of GTS/CAP)
- Three PDU (ECHO/ECHO_OK/RESET)

In similar test conditions :

- we decrease GTS loss (due to echo)
- we avoid blocking