Implementing a real-time MAC protocol under RIOT OS: running on Zolertia Z1 motes

Kévin ROUSSEL
INRIA Nancy Grand-Est / LORIA

IoT-Lab, Grenoble, 6 novembre 2014

Table of contents

Goals

- Develop new MAC/RDC protocols for WSN and IoT, that :
  - shall offer high QoS ; and
  - minimize energy consumption

  For this → we need real-time features
- These new protocols shall be integrated in a WSN OS (no "bare metal" programming) for portability and ease of use
- Part of project LAR, in the domain of home-care for aged/dependent people

Limitations of TinyOS

- Specific language (nesC) more akin to a DSL than general-purpose imperative languages
- Has its own specific paradigms:
  - statically linked callbacks (components)
  - unique stack for the whole system
  - decomposition into "tasks" that are run in a fixed order

  → complex to understand and program
- Now losing momentum

Limitations of Contiki OS

- Cooperative-only multithreading, with low granularity scheduler (128 Hz on MSP430)
- Deals with "protothreads", that impose limitations (no separate stacks, `switch`...)
- Imperfect, limited real-time mechanism (`rtimer`):
  - Only one instance for the whole system
  - Apart from a limited set of "interrupt-safe" functions, no Contiki code should be called from `rtimer`; otherwise → crash

The RIOT OS and its advantages

- Efficient, interrupt-driven, tickless microkernel
- Preemptive multitasking, thanks to a priority-aware scheduler
- Efficient use of hardware timers (all of those available on hardware)

  → It *is* a full-fledged real-time OS
- Coded in standard C (v. ISO 1999)
- Clean and modular design

RIOT OS: current limitations

- Very recent → some "moving targets"
- A network stack is bundled (contrary to FreeRTOS), *but no MAC/RDC layer* (yet)
- Needs a bit more resources than TinyOS and Contiki, especially RAM-wise (design choice: separate stacks for each thread)
  → make use of modularity on low-end HW
- Historically developed on ARM architecture
  → other ports a bit less "polished"

Our contributions

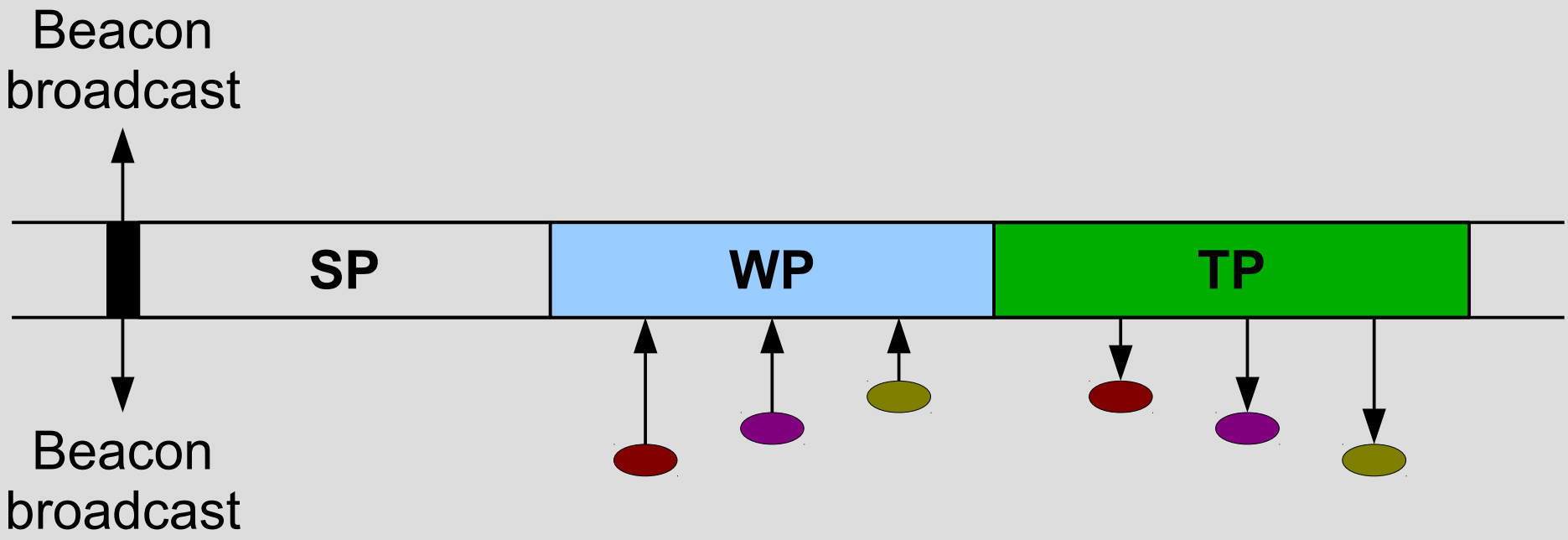
- Debugging features, especially a mechanism to handle fatal errors (`core_panic`)
- Porting to a production ready MSP430-based mote: Zolertia Z1
- Debug the HAL of the system for MSP430 architecture → *MSP430-based systems are now usable in a robust way under RIOT*

  → we now have a solid software platform for developing high-performance MAC/RDC protocols

Proof of concept: the S-CoSenS protocol under RIOT

- Improvement on CoSenS [LCN'10]
- Based on the Receiver-Initiated (RI) paradigm
- Idea: **co**llect incoming paquets, then **sen**d them in burst, **s**leep when nothing has to be done
- A duty cycle then consists in three periods...

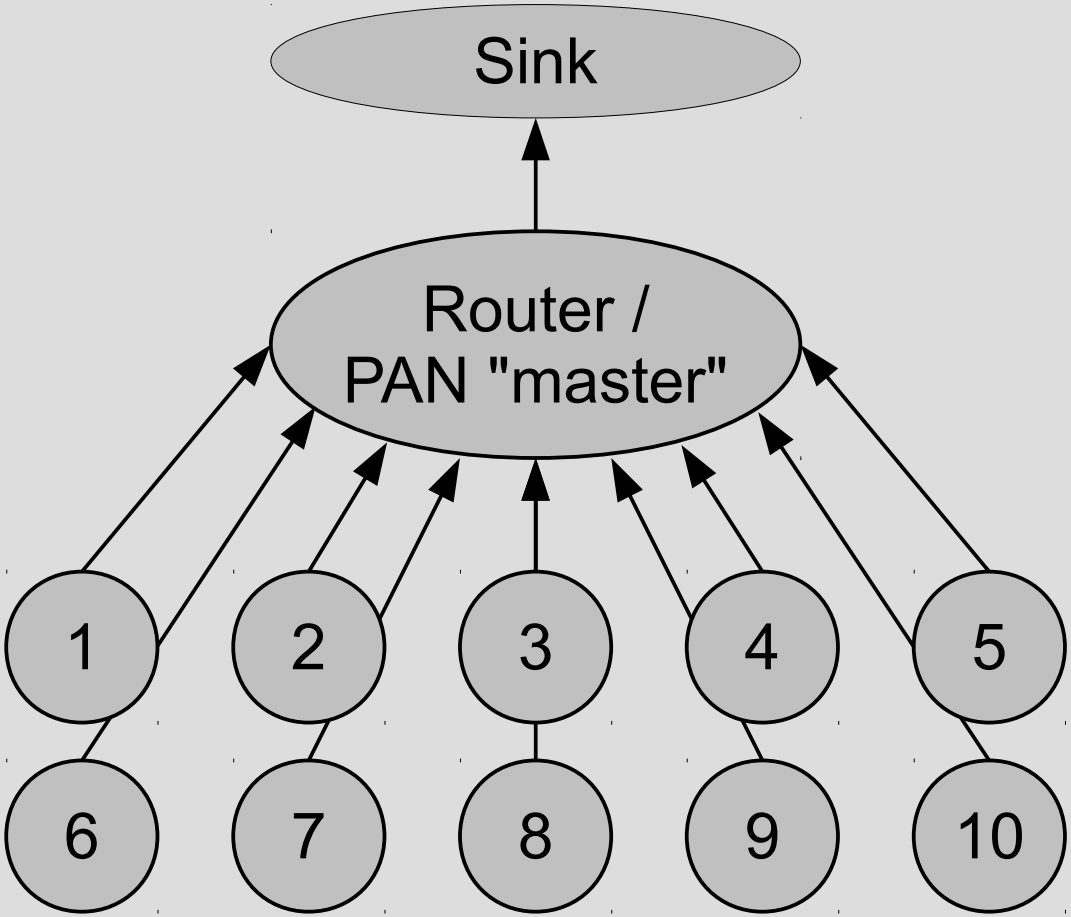
A typical S-CoSenS duty cycle



- SP = Sleep Period
- WP = Waiting (reception) Period
- TP = Transmit Period

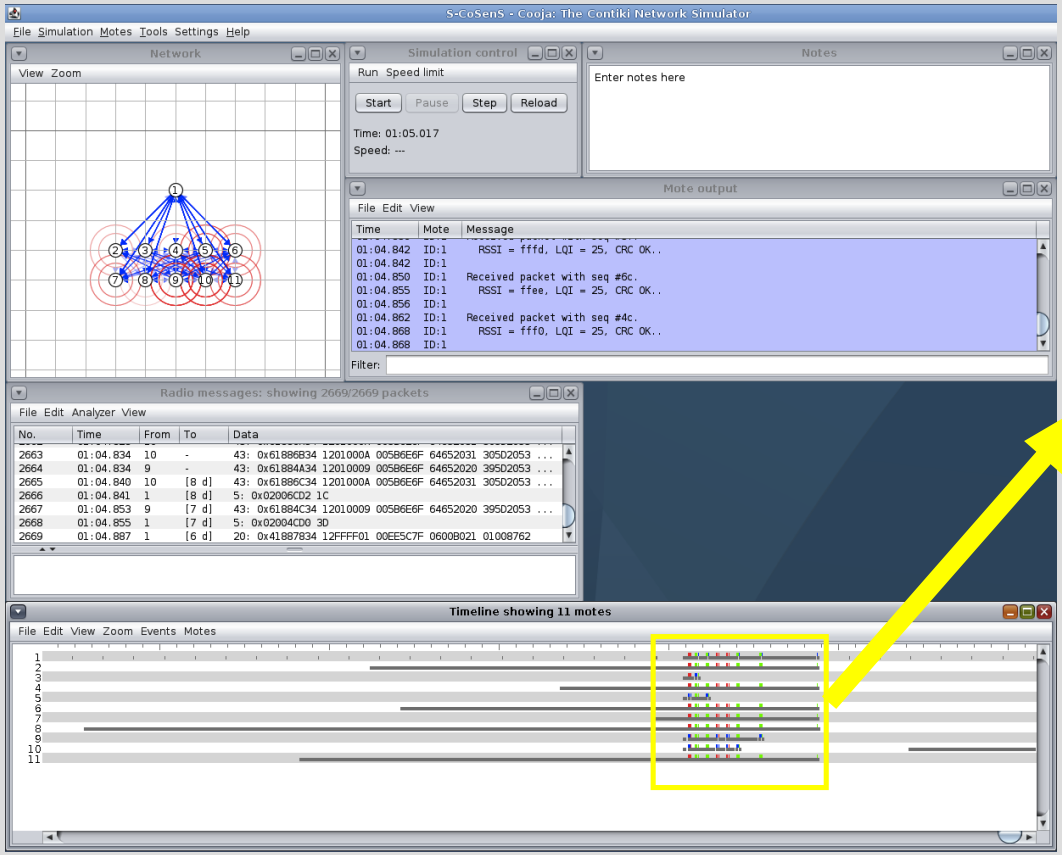
Beacon broadcast + SP + WP + TP = one duty cycle

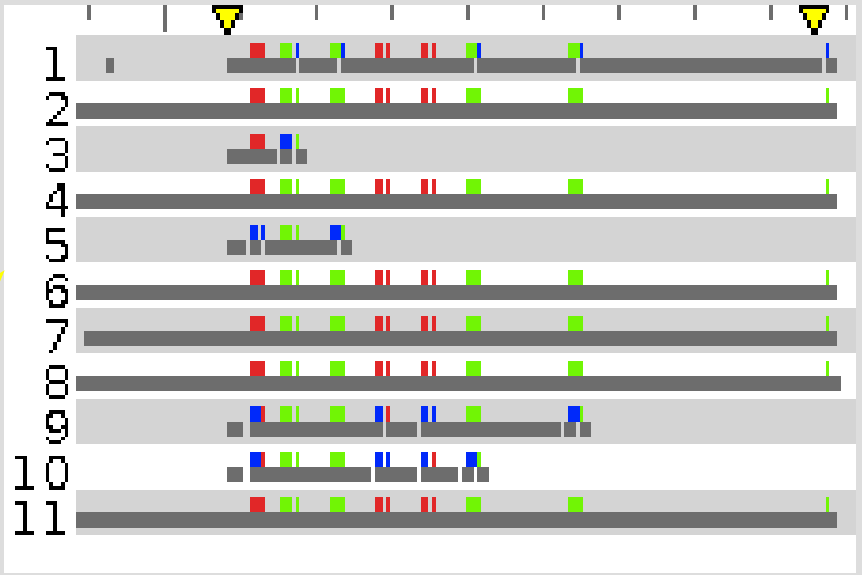
Our (virtual) test PAN setup using Cooja



Use of RIOT real-time features for precise synchronisation

- ...between nodes of a PAN
- Just impossible under Contiki (or Tiny OS)





Performance of S-CoSenS: preliminary QoS results

- We consider the rate of packets successfully transmitted to destination as the QoS marker
- S-CoSenS do better than ContikiMAC, using default parameters (duty cycle of 128ms) :

Protocol / %success	Moderate (6.7 pkt/s)	High (10 pkt/s)	Very high (20 pkt/s)	Extreme (100 pkt/s)
S-CoSenS	96.60 %	94.72 %	85.30 %	17.28 %
ContikiMAC	49.70 %	32.82 %	14.44 %	0.64 %

- More tests currently being made

Conclusion & Future Works

- We now have a functional and advanced software platform to develop our MAC/RDC protocols

  → **We aim to :**

- Develop new protocols as efficient as possible
- Provide RIOT OS the MAC/RDC layer that it network stack still lacks... and make it the best available !

Main References

- [RIOT] Hahm, O., Baccelli, E., Gu nes, M., Wa hlisch, M., and Schmidt, T. C. (2013). RIOT OS: Towards an OS for the Internet of Things. In *INFOCOM 2013, Poster Session*. `http://www.riot-os.org/`.
- [CoSenS] Nefzi, B. and Song, Y.-Q. (2010). CoSenS: a Collect- ing and Sending Burst Scheme for Performance Improvement of IEEE 802.15.4. In *LCN '10*, pages 172–175. IEEE Computer Society.