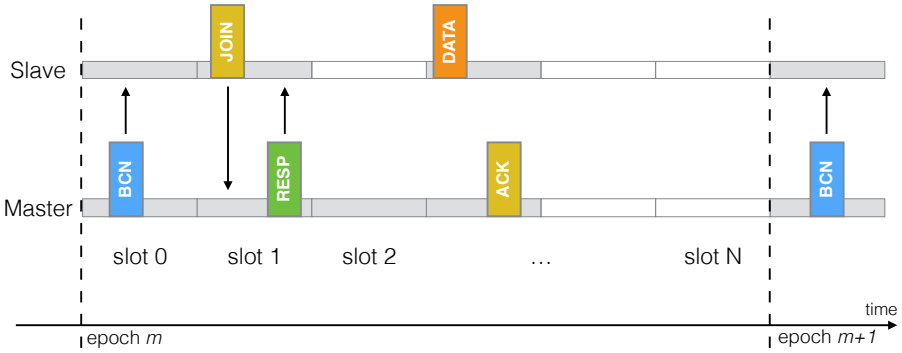
Course Project: Time Division Multiple Access Layer

Laboratory of Wireless Sensor Networks, 2015–2016

In this project, you will implement a simple power-saving time division multiple access (TDMA) channel access method for a single-hop wireless network with one *master* and multiple *slave* nodes. The master node is responsible for assigning non-overlapping time slots to the slaves that they can use to transmit packets to the master. This will prevent collisions of the transmitted data packets as soon as the schedule is established. Additionally, turning off the radios during unused slots will save energy.

Supported data flows. The system should support transmitting application data packets only in one direction: from the slave nodes to the master.

Periodic schedule. The time is split into epochs of a fixed duration T_{epoch}, and the epoch is further split into N time slots, T_{slot} long each ($T_{slot} = T_{epoch}/N$). In the beginning of each epoch, at the time slot *zero*, the master transmits a beacon packet that provides the reference point in time for the slaves. The slot *one* is reserved for control packets sent by the newly joining slaves to request a data time slot. The slots $[2; N]$ are data slots and can be assigned to the slaves, at most one slot per node. A slave device may use its data slot to transmit application data to the master. The slot duration should be long enough for the data packet to be acknowledged by the master within the same slot.



Joining the network. The master starts without any slots assigned to slaves. It just transmits the beacon packets with a fixed interval of T_{epoch}. A slave node starts with the radio always on until it receives a beacon. Having received a beacon, the node knows the epoch start time t_{ref} and can compute the beginning of every slot in the epoch. At the slot *one*, it should send a slot request to the master. The master should assign any of the available slots, memorize the assignment of the slot to the joining slave and reply with the assigned slot number within the slot *one*. Having received the slot number, the slave considers itself joined and should start following the assigned schedule in all future epochs. If the reply of the master was lost, the slave should repeat the joining attempts in the following epochs.

Following the schedule. As a general rule, the radio of a node that expects to receive a packet during a given slot should be on. During the slots that are not assigned to any node, all the radios should remain off. If a slot is assigned, but the corresponding sender node does not have data to send, the radio of the sender should remain off during that slot.

Lost synchronisation. When a slave node loses 5 beacons in a row, it turns into the *resynchronising* mode. It stops following the previously assigned slot schedule and stops transmitting any data packets, turns the radio on and waits for a beacon to arrive. Once a beacon arrives, the node goes back to the normal synchronised operation without issuing a new join request.

Application interface. The TDMA layer should provide the standard `AMSend` and `Receive` interfaces of TinyOS for sending and receiving data, as well as a custom interface for initialising the TDMA layer either in slave or master mode.

Underlying services. Use the `CC2420TimeSynchMessageC` for sending beacons with integrated reference time t_{ref} . For slot request and response as well as data packets, use the `AMSenderC` and `AMReceiverC` components. Use the `PacketLink` interface to set up acknowledged transmission of data packets. For all time operations with slots and epochs use the provided 32kHz `Timer32C` component. Do not enable LPL.

Rules of the game.

- The project is individual. You are strongly encouraged to deliver it before the beginning of the second semester, and in any case no later than September 2016.
- You should submit the code and a short description document and show that your project works (in the Cooja simulator).
- The code MUST be properly formatted. Follow style guidelines (e.g. this one).
- You MUST contact through e-mail the instructor (gianpietro.picco@unitn.it) AND the teaching assistants (piyare@fbk.eu, timofei.istomin@unitn.it), well in advance, i.e. a couple of weeks, before the presentation.
- Both the code and documentation must be submitted in electronic format via email at least one day before the meeting. The documentation must be a single self-contained pdf/txt. All code must be sent in a single tarball consisting of a single folder (named after your surname) containing all your source files.
- The project will be evaluated for its technical content (algorithm correctness). *Do not* spend time implementing unrequested features — focus on doing the core functionality, and doing it well.
- The project is demonstrated in front of the instructor and/or assistant.

Plagiarism is not tolerated. Do not hesitate to ask questions if you run into troubles with your implementation. We are here to help.