https://goo.gl/LKBqPJ

petulant-moo

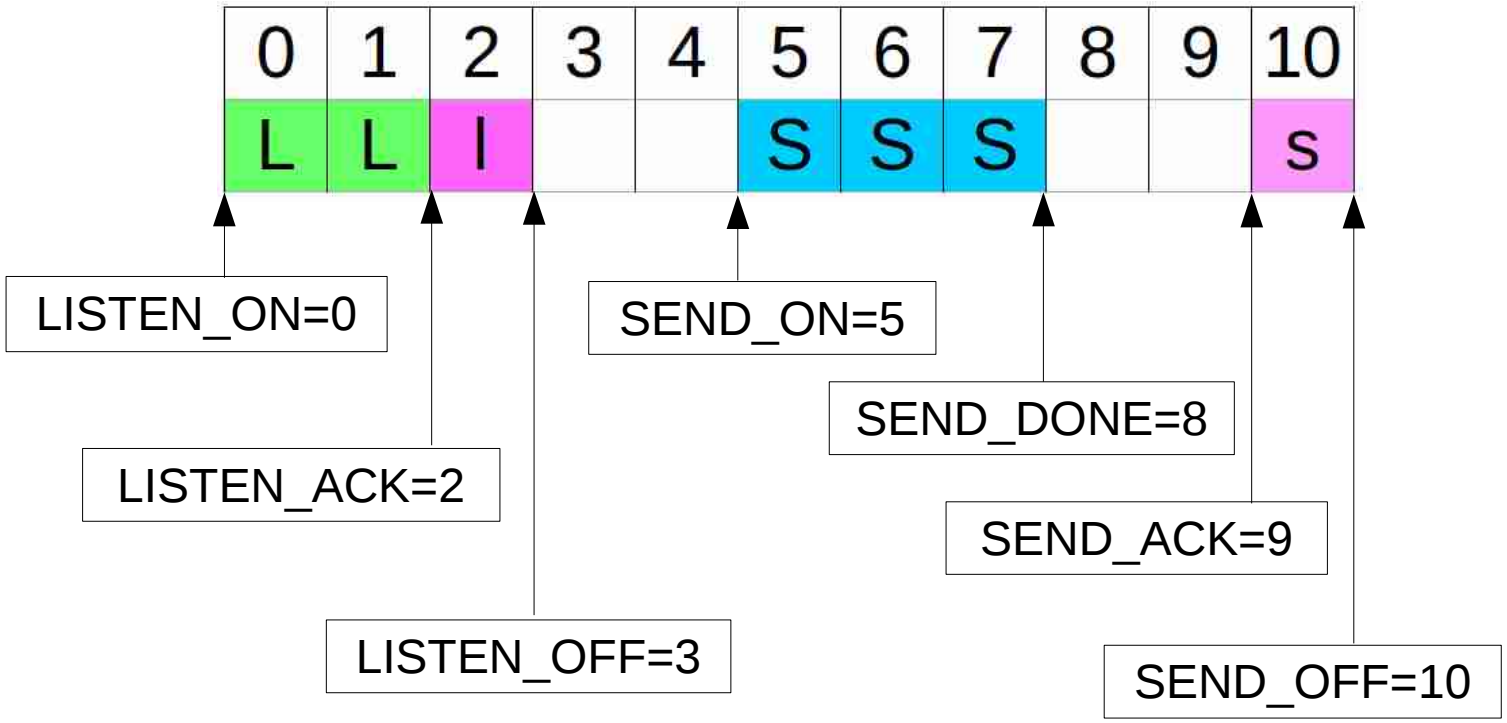
Pre-determined TDMA scheduling.

An efficient multihop routing algorithm for low power high data yield in static topologies.

https://github.com/gubser/petulant-moo

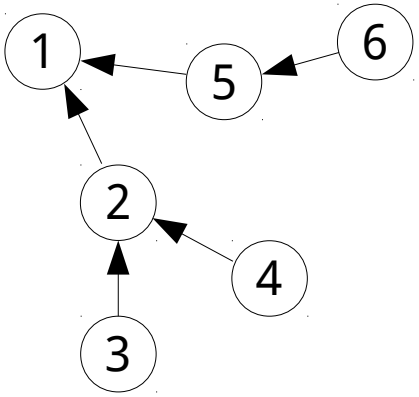
Scheduler

- Idea: Low duty cycle with pre-determined TDMA.
- State machine on every node executing its given schedule:



Scheduler

- Schedule for each node pre-computed based on selected topology:



t	0	1	2	3	4	5	6	7	8	9	10
node 3	S		s								
node 4		S	s								
node 2	L	L	I			S	S	S			s
node 6				S	s						
node 5				L	I				S	S	s
node 1						L	L	L	L	L	I

Python script:

```
mote2 = M(2, [M(3), M(4)])
mote5 = M(5, [M(6)])
sink = M(1, [mote2, mote5])

sink.calculate()
```

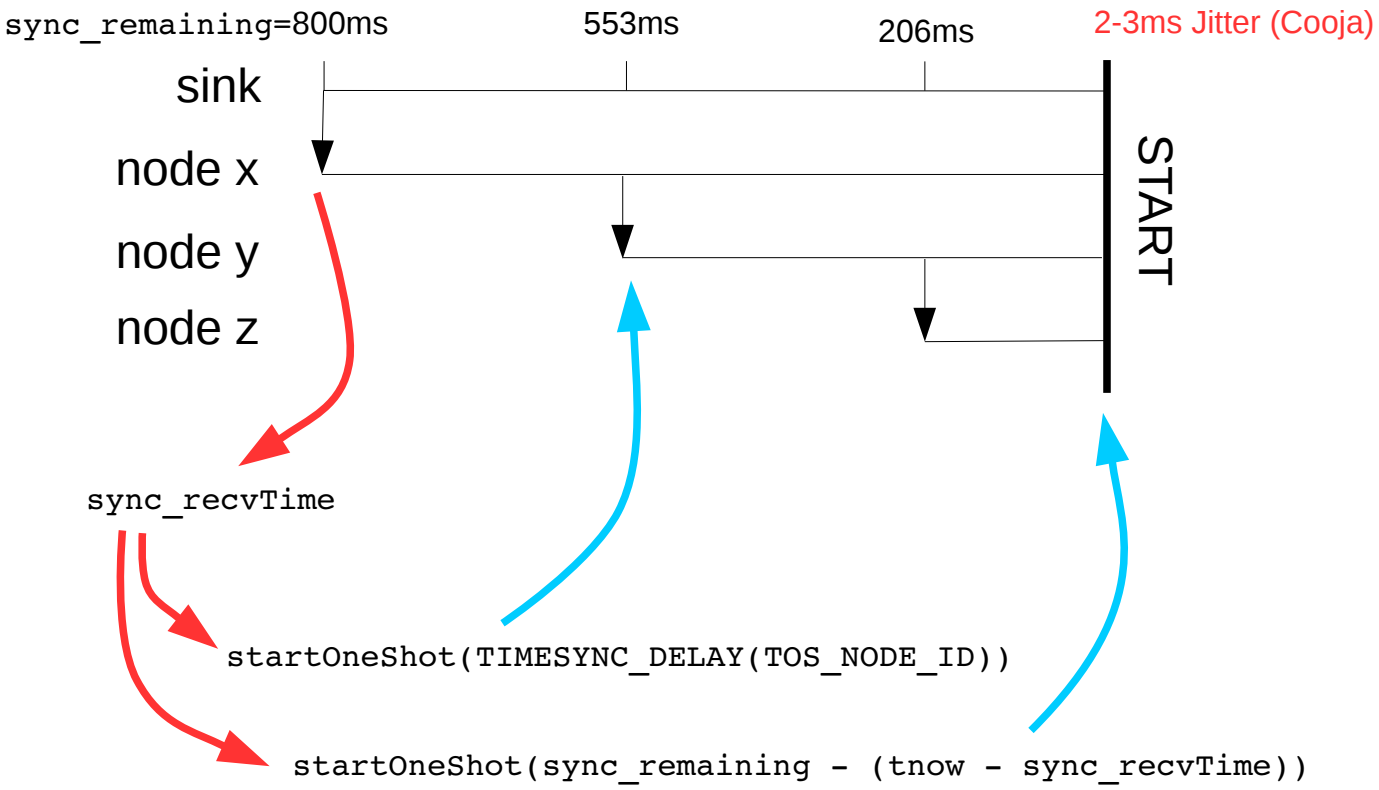
→ **C code output**

Legend

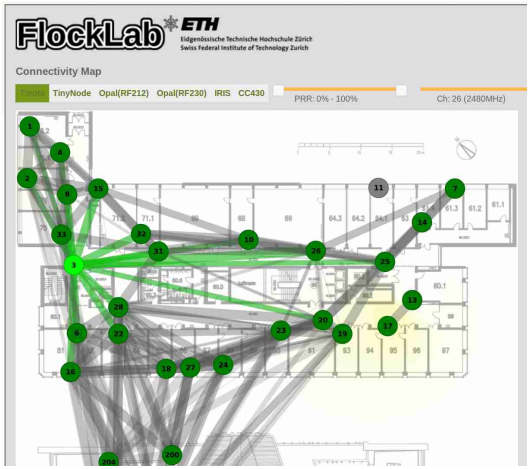
L	Listen	S	Send
I	Send ACK	s	Recv ACK

Connectivity

CC2420TimeSyncMessageC



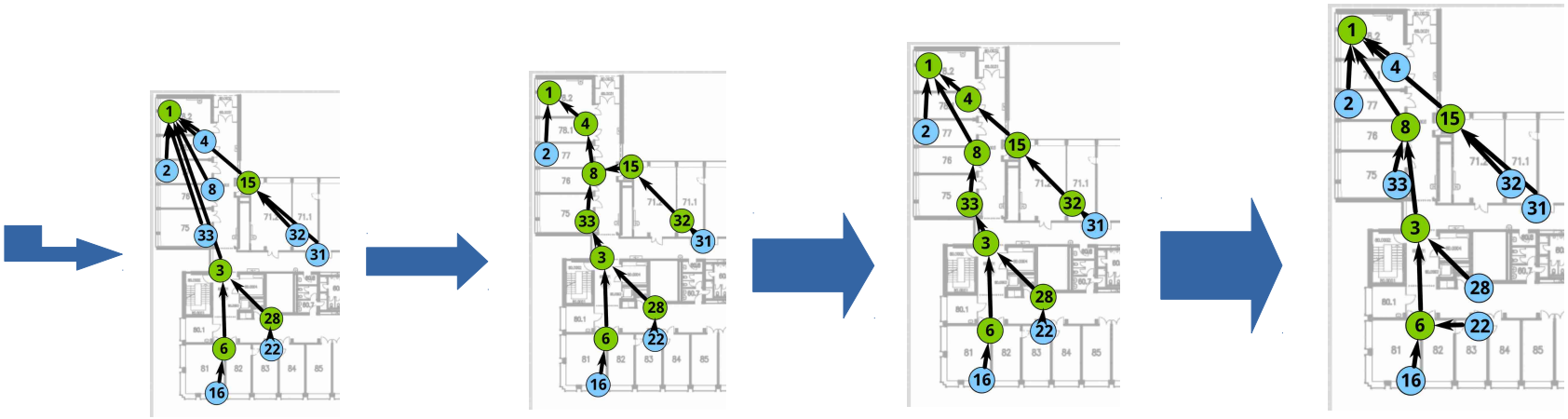
Connectivity



- Low current consumption thanks to scheduler.
- But not full data yield. How is the connectivity?

➡ Topology improvement by:
- connectivity map
- test runs

Example 10Pkt/s:



DY: 72.69%
I_avg: 0.55mA (0.98)
KPI: 85%

DY: 87.65%
I_avg: 0.87mA (0.97)
KPI: 92%

DY: 100.00%
I_avg: 0.76mA (0.97)
KPI: 98%

DY: 99.62%
I_avg: 0.61mA (0.98)
KPI: 99%

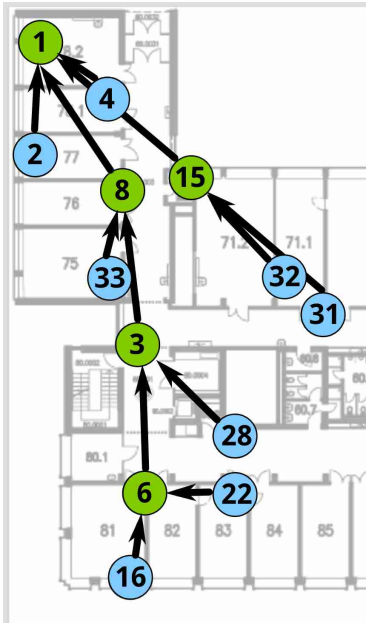
(Simple) ACK

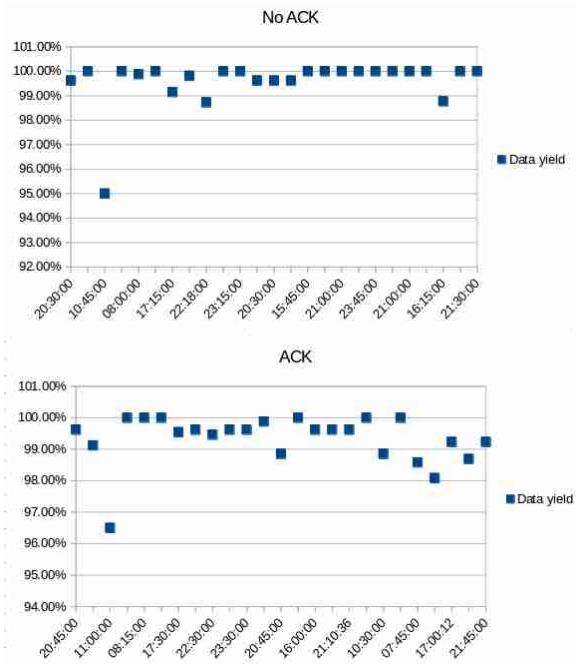
- Can ACK improve the results?

  Can better data yield compensate worser power performance?

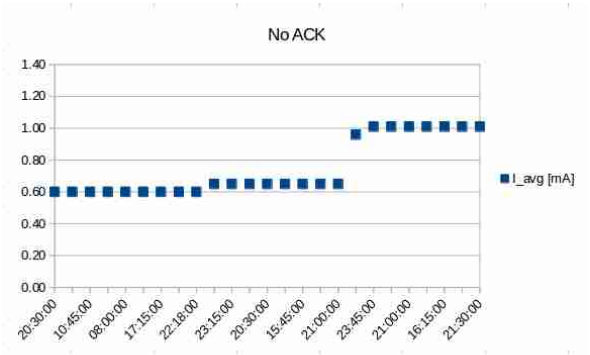
- Idea: Is any data received?
  - Yes: Clear sent packets in buffer
  - No: Keep packets in buffer

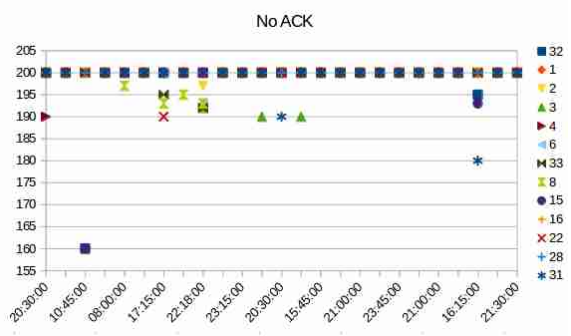
- Problem:
  - Jitter: TDMA not working properly
  - More packets losses in good environment

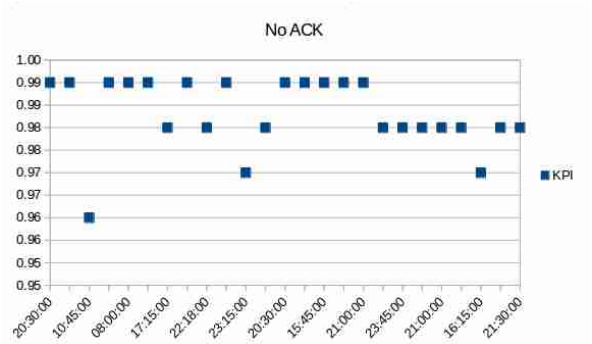


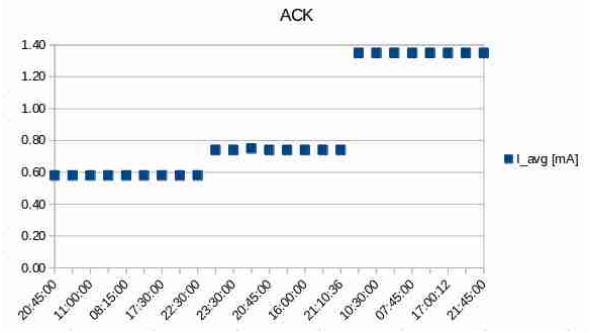


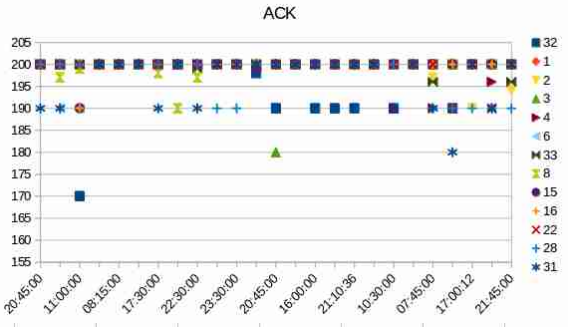
Results

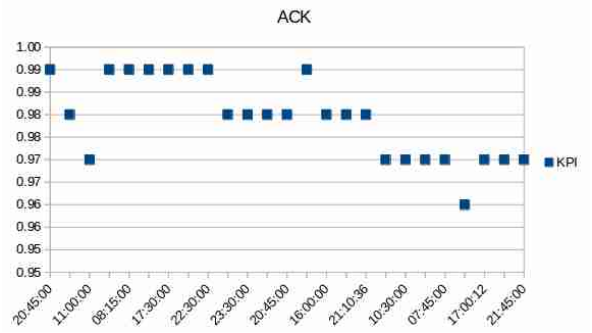












- 50 test runs from 2nd December to 5th December

NO ACK:
Data yield: 99.59%
I_avg: 0.75mA
KPI: 98%

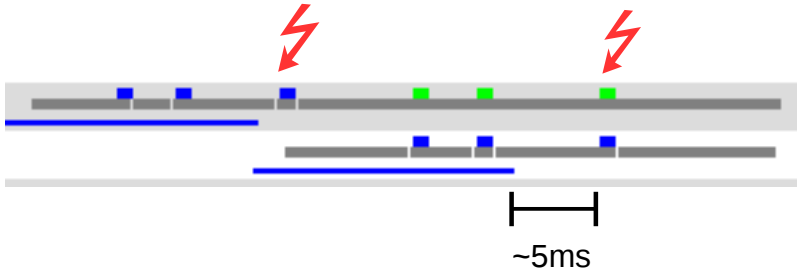
ACK:
Data yield: 99.33%
I_avg: 0.88mA
KPI: 98%

Challenges

- Reduce number of packets by bundling 10 data packets into one:

```
typedef nx_struct group_bulk_msg {
  nx_am_addr_t source;
  nx_uint8_t seq_no;
  nx_uint16_t data[BULK_SIZE];
} group_bulk_msg_t;
```

- CC2420 TinyOS implementation not suitable for TDMA. **Jitter → Packet Loss**
  "Volunteers are needed to redo this architecture […] to build other types of channel sharing mechanisms (i.e. TDMA)." [1]
- Due to very short channel occupancy, we are lucky most of the time.



[1] http://tinyos.stanford.edu/tinyos-wiki/index.php/CC2420_Layer_Descriptions

Challenges

- Intensively testing