Kiel University
Christian-Albrechts-Universität zu Kiel

Distributed Systems
Research Group

IoT Lab 1

Your first wireless network

Overview

- Basic concept of this lab
- Your tasks
- Your deliverables

Kiel University
Christian-Albrechts-Universität zu Kiel

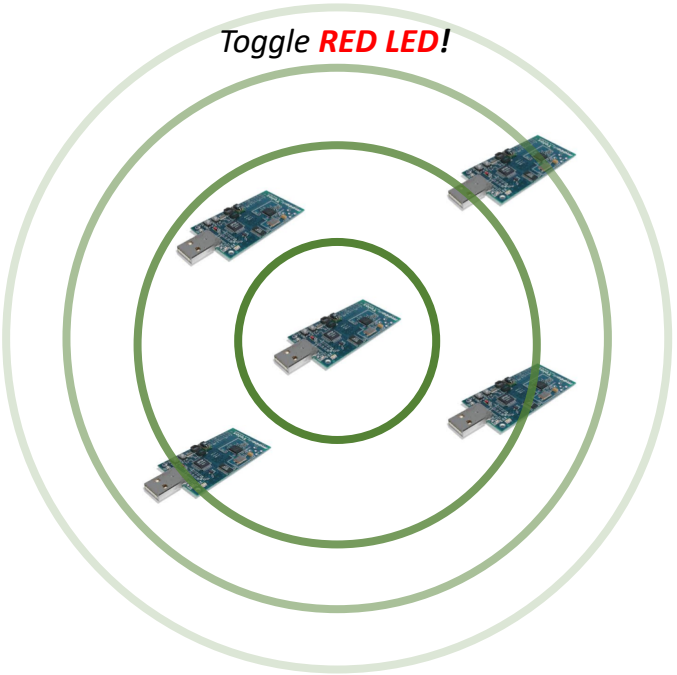
Distributed Systems
Research Group

Basic Concept

One-hop Dissemination & Collection

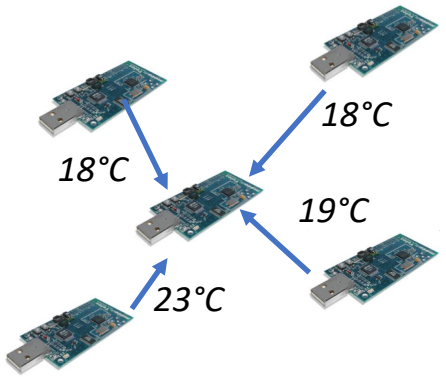
Main Idea

- We will study a single-hop wireless network
  - Any node can hear every other node



- Part 1: A central node forces all nodes to toggle a LED
  - Every second
  - Random LED

- Part 2: All nodes must share a sensor value with the central node
  - Every second
  - Random value



Setup

- Create a new Cooja simulation
  - Unit disk graph medium
  - 5 Sky motes
  - Random positioning (with a max of 50m in X and Y!)

- If you compile from Cooja, always remember to clean first before recompiling!

Tasks

What you must do

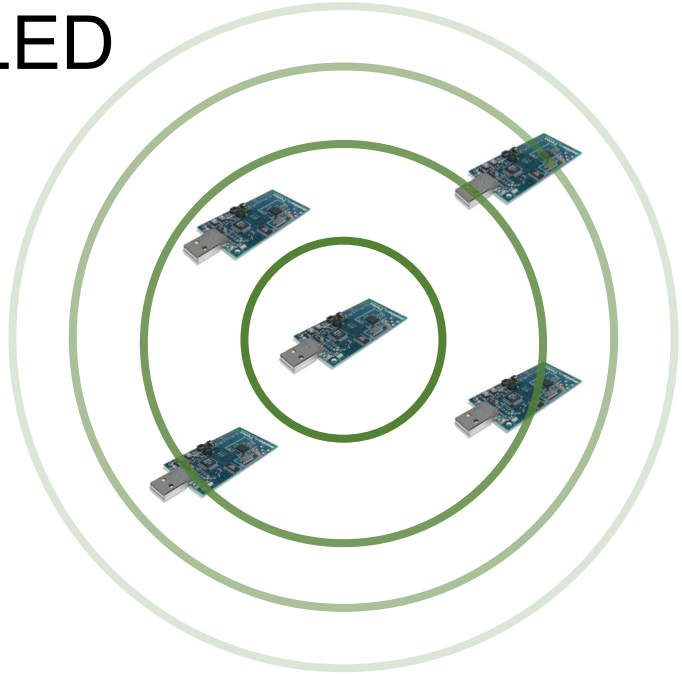
Part 1 – Command dissemination

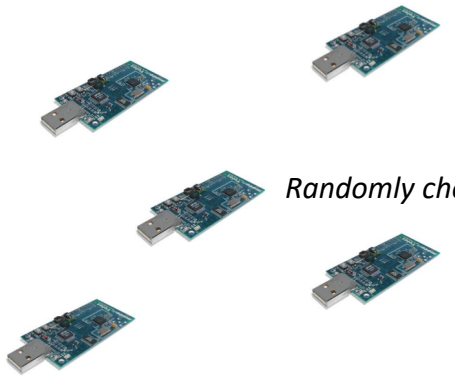
5 nodes: one central node (ID 1), four ”slave” nodes (ID 2 to 5)

Every second:

- The central node selects a random LED
- Sends a ”Toggle LED” command to the network
- Upon reception, a slave toggles the associated LED

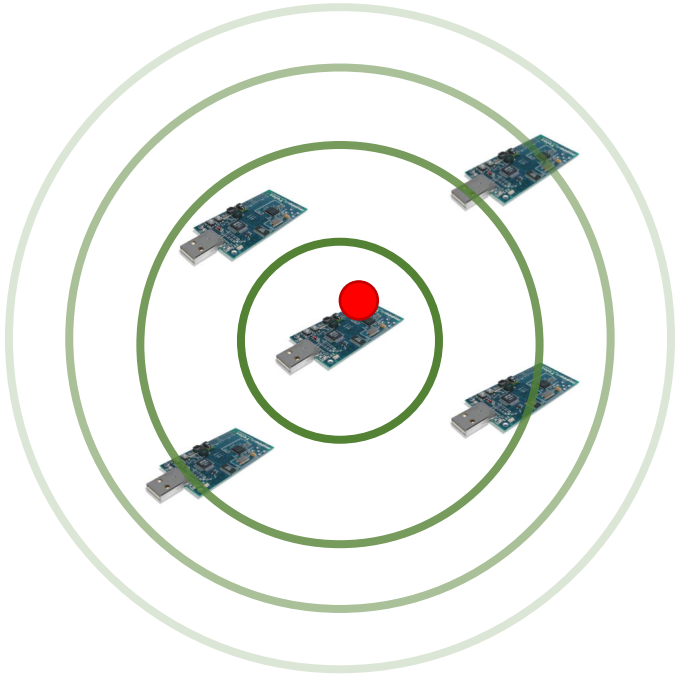
*Toggle **RED LED**!*

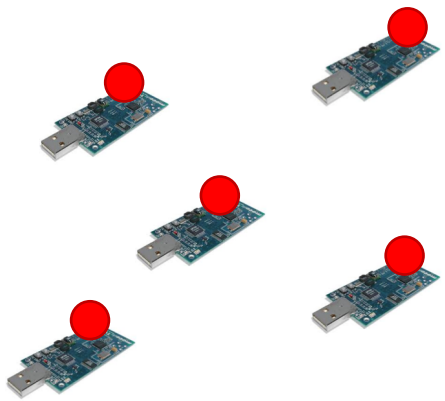


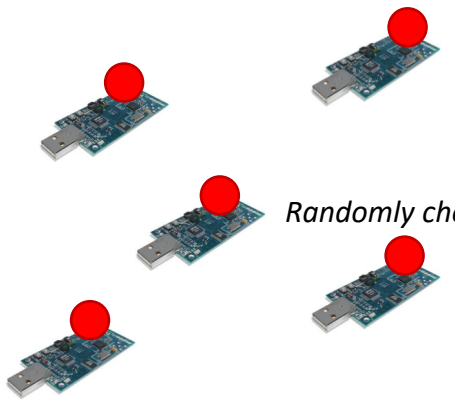


*Randomly chosen LED: **RED LED***

*Toggle **RED LED**!*







*Randomly chosen LED: **GREEN LED***

Some hints

- How to use LEDs

  #include "dev/leds.h”

  Three leds: *LEDS_RED, LEDS_GREEN, LEDS_YELLOW* (yellow will be blue in cooja)

  leds_on(LEDS_RED); turns on the red led

  leds_off(LEDS_GREEN); turns off the green led

  leds_toggle(LEDS_YELLOW); toggles the yellow/blue led (toggle: turn off if on, turn on if off)

- How to use random numbers:

  #include <random.h>

  unsigned short rand = random_rand()%100; gives a short between 0 and 99.

  (random_rand()%50)+50; gives a short between 50 and 99

Some hints

- How to send to more than one node? Unicast or broadcast?

  You can check examples/nullnet/nullnet-broadcast.c

  Or examples/nullnet/nullnet-unicast.c

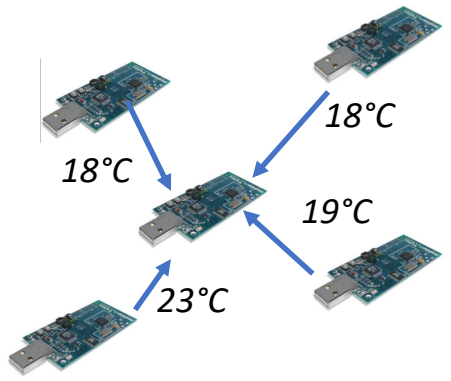
Please note, in nullnet-broadcast.c, that line 100 is superfluous (the memcpy copies from the count address memory to nullnet_buf, which itself points to count. This behavior should be avoided)

Part 2 – Data collection

5 nodes: one central node (ID 1), four ”slave” nodes (ID 2 to 5)

Every second:

- Each slave node generates a random value between -100 and +100
  - This is our ”temperature” value
- Sends the temperature to the central node
- Once all values are received, the central node prints each value and the average temperature



Temperature: 20°C

Temperature: 30°C

Temperature: -20°C

Temperature: -50°C

Temperature: 20°C

Temperature: 30°C

Temperature: -20°C

Temperature: -50°C

Temperature: 20°C

Temperature: 30°C

Temperature: -20°C

Temperature: -50°C

Temperature: 20°C

Temperature: 30°C

Temperature: -20°C

Temperature: -50°C

Temperature: 20°C

Temperature: 30°C

Temperature: -20°C

Temperature: -50°C

Temperature: 20°C

Temperature: 30°C

Temperatures: 20, 30, -50, -20
Average: -5

Temperature: -20°C

Temperature: -50°C

Some hints

- How to send to a specific node? You can use unicast

  see examples/nullnet/nullnet-unicast.c

  The address is printed at the beginning of the simulation

  ID 1: linkaddr_t dest_addr = {{ 0x01, 0x01, 0x01, 0x00, 0x01, 0x74, 0x12, 0x00 }};

- How to send complex data?

  Use a struct!

  You can cast it to uint8_t *, when you point nullnet_buf

More hints

- In Cooja:
  - To display the LEDs: in Timeline, check Events->LEDs
  - If you compiled your code within Cooja, you can reload Cooja (Ctrl+R) to recompile the code
    - Otherwise, you can use make TARGET=sky clean && make TARGET=sky nullnet-unicast (if your file is called nullnet-unicast.c) in your terminal
  - You can filter in the Mote output:
    - *ID:1* will only display messages from node 1
  - You can zoom in and out in the timeline!
  - In the timeline, green = reception (Rx), blue = transmission (Tx), red = corrupted Rx/Tx

Your Deliverables

What you must submit

You must submit

- A **video** presenting your solutions:
  - 5 to 8 min (we remove points if the video is longer than 8 min!)
  - The video must contain:
    - A demo of your solution in cooja
    - An explanation of:
      - your design choices
      - your code
      - possible corner cases (what could break your solution)?
- Your **code**
  - Any file you might have modified

How to submit?

- Upload your source code and video as an archive (zip, rar) on **iLearn**
  - iLearn limits the file size to ~20Mb, if your file is too large, upload the source code only on iLearn, and the video on a dropbox link given in iLearn
  - Please note that the dropbox repository is shared with everybody, so you must limit your video size to **50Mb!!!**
- Like the prelab, you will work as a group (2 students)

Good luck!

Deadline: Monday, 27th April

To go further

- K. Romer and F. Mattern, "The design space of wireless sensor networks," in *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54-61, Dec. 2004.

- A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol", SICS Technical report

- Ad-hoc On-demand Distance Vector Routing (AODV), https://www.cs.jhu.edu/~cs647/aodv.pdf