

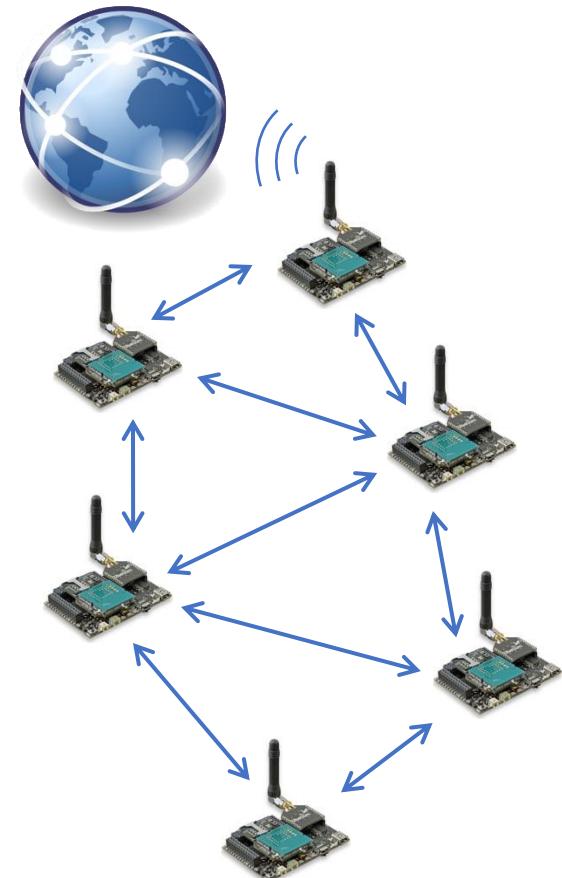
Low-power Wireless Networking for the Internet of Things

Lab5: The CLOVES testbed

Matteo Trobinger (matteo.trobinger@unitn.it)

Credits for some slides to:

Davide Vecchia, Enrico Soprana



CLOVES



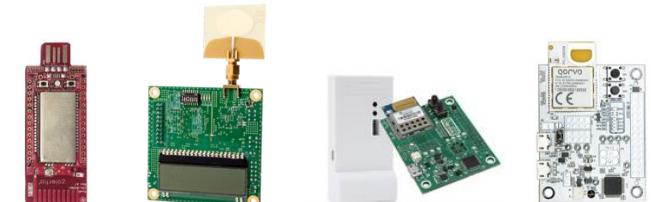
Communication and Localization testbed for Validation of Embedded Systems

Info at iottestbed.disi.unitn.it/cloves &

Poster abstract:
CLOVES: A Large-scale Ultra-wideband Testbed
Davide Molteni, Gian Pietro Pucco, Matteo Tröbinger, Davide Vecchia
{davide.molteni,gianpietro.pucco,matteo.troebinger,davide.vecchia}@unitn.it
University of Trento, Italy

ABSTRACT
Decentralized devices in low power wireless systems have greatly benefited from the availability of public testbeds. However, none is currently available for the increasingly popular ultra-wideband (UWB) radios enabling communication and localization. We present CLOVES, the first publicly available UWB testbed supporting UWB/UWB, IEEE 802.15.4 and WiFi. The testbed consists of 130 nodes, each with an STM32F103CBT6 (ARM Cortex M3 MCUs). However, the infrastructure supports multiple platforms at once. Several nodes host a ZedBoard, which is a Xilinx Zynq-7000 SoC board, and a BeagleBoard with an AM335x SoC; integrating an ARM Cortex M4 MCU with a WiFi radio. Finally, all nodes also host a Zelena Firefly equipped with a CC2538 SoC, integrating an ARM Cortex M3 MCUs and a 2.4 GHz IEEE 802.15.4 transceiver radio. Our platform is a popular choice

CCS CONCEPTS
• Networks • Network experimentation



In a nutshell:

- 130 nodes over 3 areas for a total of 7250 m²
- 344 devices
- The largest publicly-available UWB testbed
- One-of-a-kind opportunity for direct comparison of narrowband vs. UWB network stacks
- Remotely controlled, from the website or the Python client

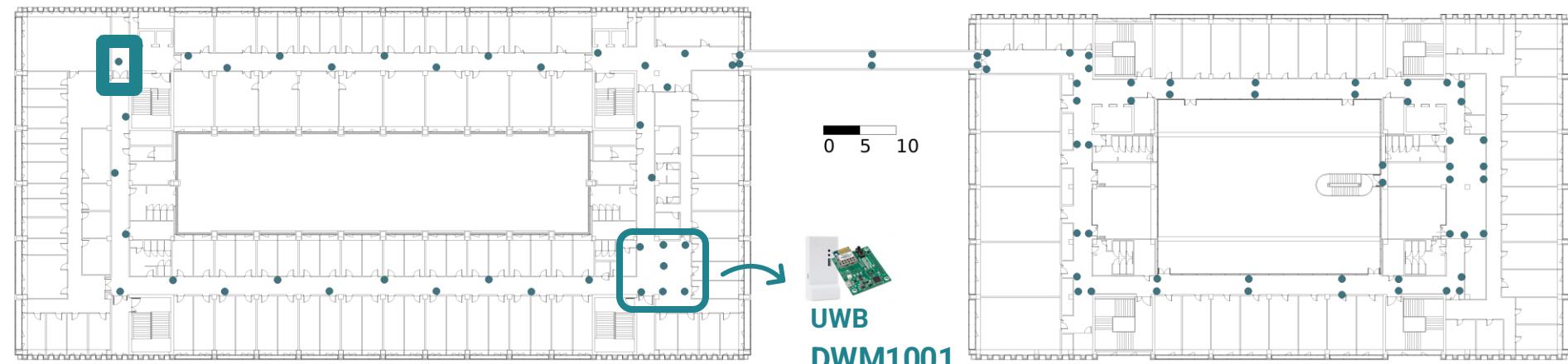
We will use the website: <https://research.iottestbed.disi.unitn.it/>

but you can find the instructions for the client here:

<https://github.com/d3s-trento/cloves-client>

DEPT

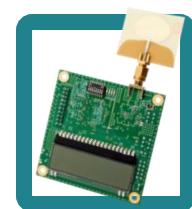
91 nodes (6382 m²)



In all nodes:



UWB
DW3001 CDK



UWB
EVB1000



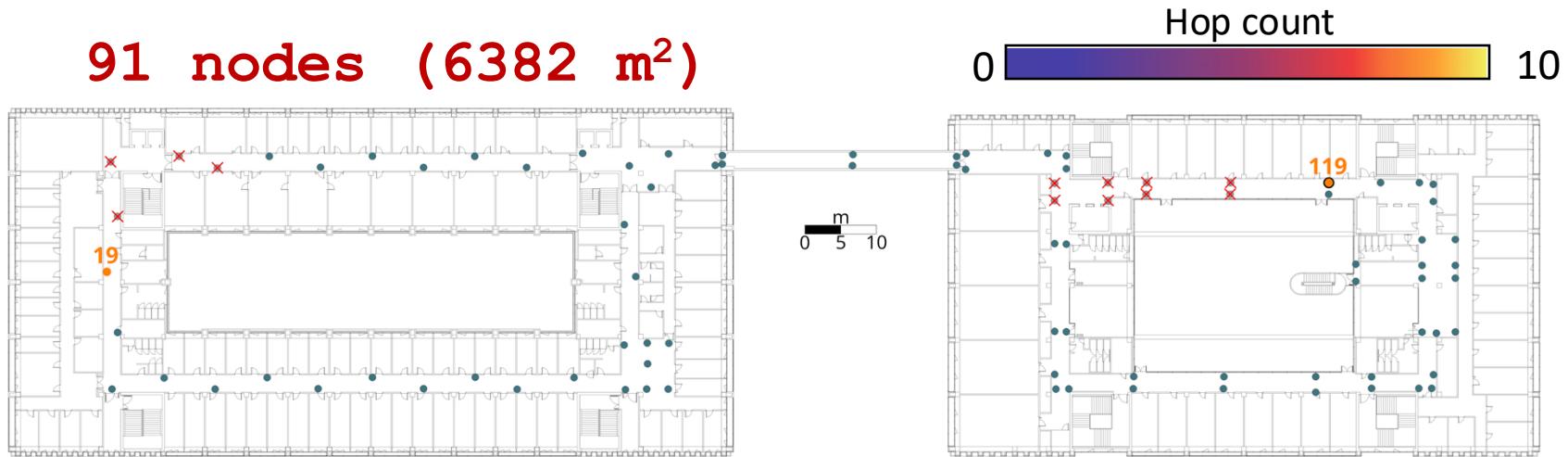
2.4GHz
Firefly

Main characteristics:

Long (up to 134 m) and narrow (1.7 – 3.2 m) corridors with rare wider areas.

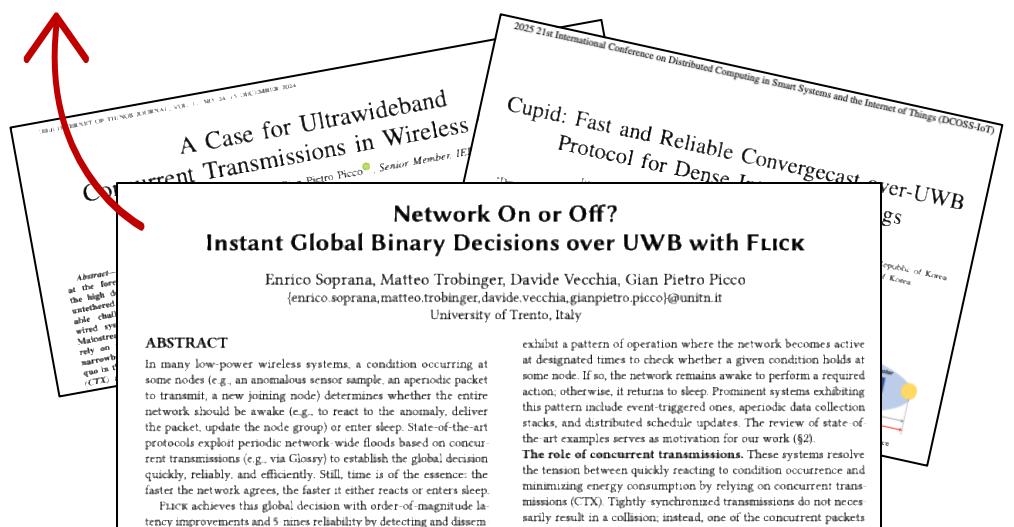
» High likelihood of strong signal reflection and RF interference (especially for )

DEPT: Testing multi-hop communication

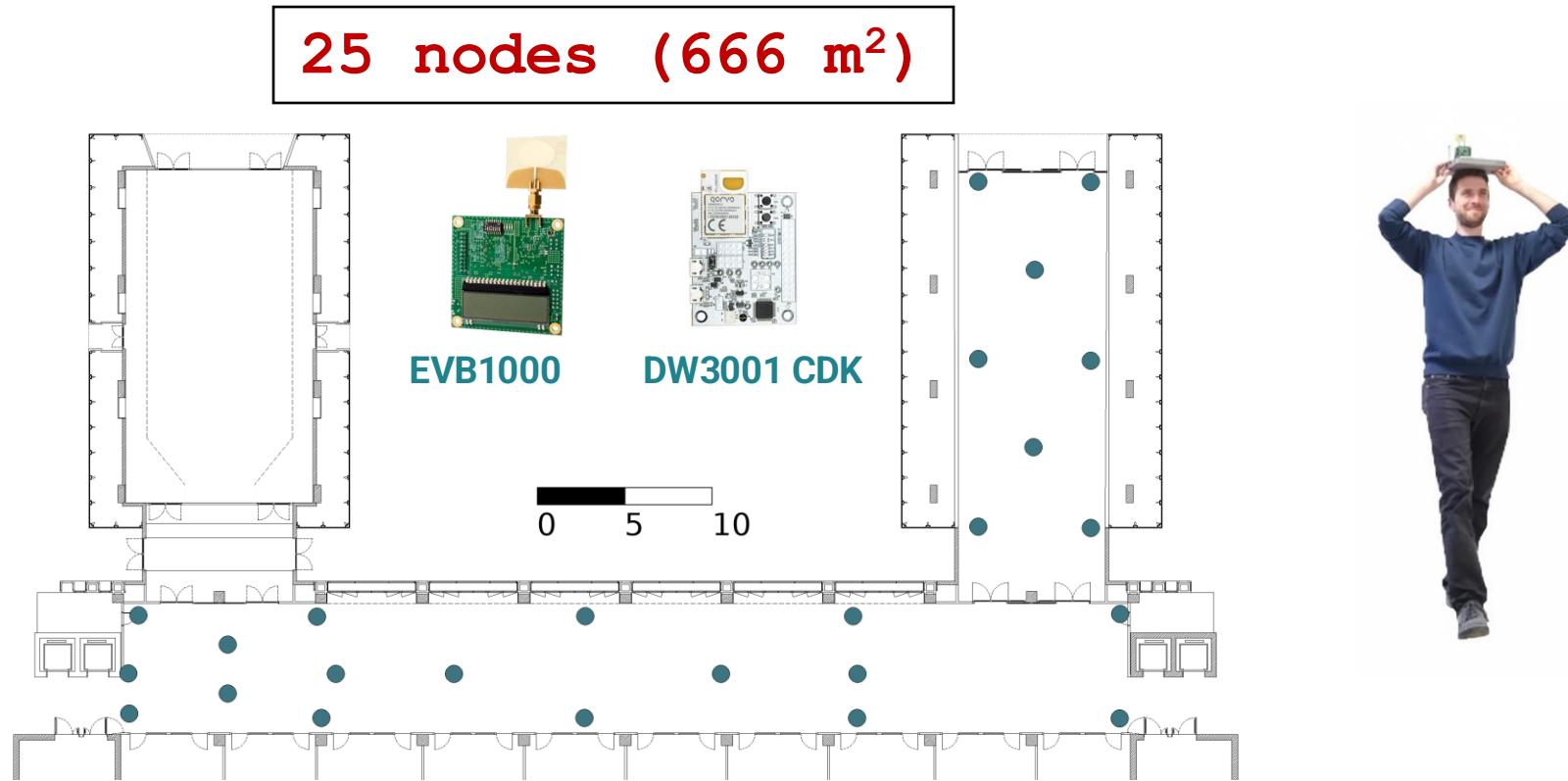


FLICK disseminates a binary ON-OFF value over 10 hops in < 500 µs

Multi-hop network with
10+ hops
1410 links
>15 neighbors per node

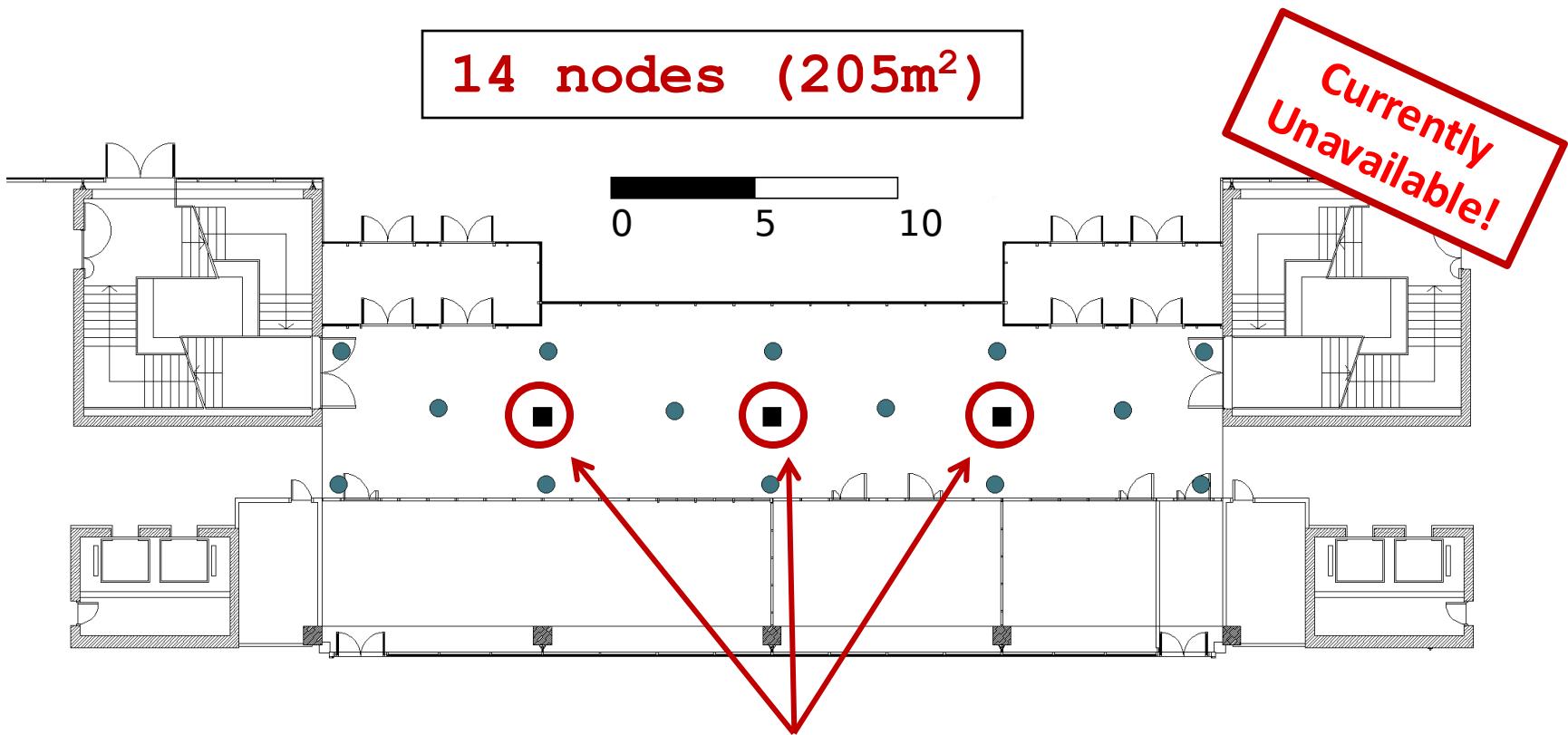


HALL-A



- Could be used for communication, but its geometry is *particularly suited* to study UWB ranging and localization schemes
- Useful to evaluate ranging and localization schemes in line-of-sight (LOS) and non-line-of-sight (NLOS) conditions

HALL-B



Node mapping: From CLOVES IDs to IEEE addresses

<https://research.iottestbed.disi.unitn.it/idMapping/>

Node Id	Area	Coordinates	Dwm1001 address 	Evb1000 address 	Firefly address 
1	DEPT	(76, 3.97)	01:3a:61:02:c3:f4:1a:01	04:32:51:02:01:64:13:9a	00:12:4B:00:18:D6:F7:9C
2	DEPT	(72.74, 6.6)	01:3a:61:02:c3:f4:80:89	10:20:5f:13:10:00:19:15	00:12:4B:00:14:B5:D9:76
3	DEPT	(75.97, 6.86)	01:3a:61:02:c3:f4:c2:36	01:3a:61:02:c4:c2:11:0c	00:12:4B:00:18:D6:F3:84
4	DEPT	(78.98, 6.85)	01:3a:61:02:c3:f4:8f:2a	04:32:51:02:41:64:12:8a	00:12:4B:00:18:D6:F3:EE
5	DEPT	(78.89, 0.44)	01:3a:61:02:c4:c2:89:00	04:32:51:02:41:64:11:a3	00:12:4B:00:18:D6:F7:92
6	DEPT	(75.89, 0.43)	05:c3:22:08:4a:d4:17:83	04:32:51:02:01:64:10:9b	00:12:4B:00:18:D6:F3:9A
7	DEPT	(72.91, 0.37)	01:3a:61:02:c4:40:d7:9f	10:20:5f:13:10:00:18:33	00:12:4B:00:14:B5:DE:21
8	DEPT	(65.73, 2.05)	None	04:32:51:02:41:64:10:89	00:12:4B:00:18:D6:F2:A1
9	DEPT	(57.79, 0.42)	None	10:20:5e:fe:10:00:11:4a	00:12:4B:00:14:B5:D8:B5
10	DEPT	(51.93, 2.09)	None	05:c3:22:08:4b:94:c8:0c	00:12:4B:00:18:D6:1E
11	DEPT	(44.7, 0.49)	None	10:20:5f:13:10:00:18:32	00:12:4B:00:18:D6:5F
12	DEPT	(37.56, 2.06)	None	04:32:51:02:41:64:17:93	00:12:4B:00:18:D6:33

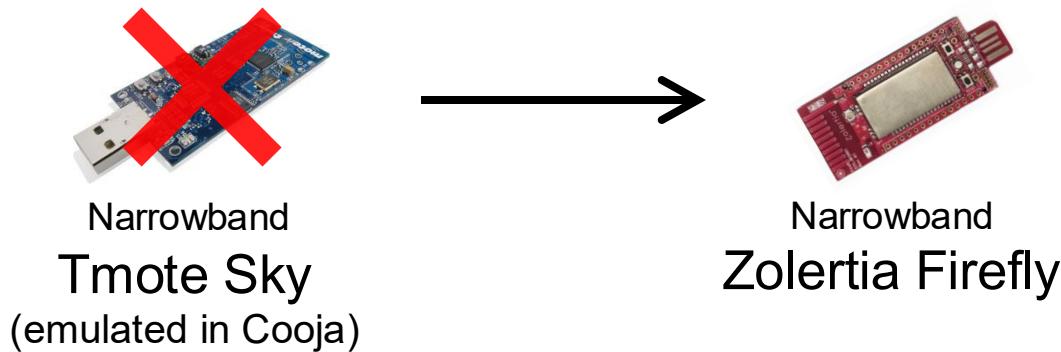
Each node hosts one or more platforms
(DWM1001, EVB1000, Firefly).

The 802.15.4 short address of the node
(Contiki linkaddr_node_addr)

Running a test with the CLOVES website

1. Compile your code targeting a platform available in CLOVES.

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.



Compile for Zolertia Firefly



1. Provide platform-specific configuration in **project-conf.h**.

Compare today's `project-conf.h` file with those used in previous exercises.

2. Specify the correct target and board in the **Makefile**

```
TARGET ?= zoul  
BOARD ?= firefly  
LDFLAGS += -specs=nosys.specs
```



3. Compile with **make**

Since we are programming on top of Contiki, you can run all previous exercises on a different platform **with almost no changes to the application code!**

Running a test with the CLOVES website

1. Compile your code targeting a platform available in CLOVES.

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.

2. Log into the CLOVES web interface.

Visit <https://research.iottestbed.disi.unitn.it/>

Documentation

- User manual (client)
- Job File
- Snapshot

Log in

Log in with your testbed credentials

Private area

- Profile
- Create Job
- Create Reservation
- Download Jobs

Click on "Create Job"

Running a test with the CLOVES website

1. **Compile your code targeting a platform available in CLOVES.**

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.

2. **Log into the CLOVES web interface.**

Start at research.iottestbed.disi.unitn.it, and log in with your credentials. Then navigate to “Create job”.

3. **Insert scheduling information.**

Area of the test, start time, duration.

Insert scheduling information [STEP 3]

CLOVES synonym
for “Area”

“DEPT” or “HALL-A”. If you use
Firefly nodes “DEPT” is the only option

Time info

Island DEPT

Start time ASAP

Start time date 01/11/2023 22:15

Duration 60

Options: “Now”, “ASAP”, and “Date”

In seconds

Only available with “Date” or “ASAP”.
With “Date”, job starts at the specified date
and time, if possible. With “ASAP”, it starts at
the first available timeslot after the specified
date and time

Running a test with the CLOVES website

1. **Compile your code targeting a platform available in CLOVES.**

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.

2. **Log into the CLOVES interface.**

Start at research.iottestbed.disi.unitn.it, and log in with your credentials. Then navigate to “Create job”.

3. **Insert scheduling information.**

Area of the test, start time, duration.

4. **Upload the binary, specifying the target platform.**

Binary upload [STEP 4]

Select “Firefly” or “eVB1000” (Lab 9)

Upload the .bin file obtained from step 1

Binary file 1

Upload file

Hardware firefly

Bin file connectivity.bin

Targets all

Programaddress 0x00200000

The screenshot shows a software interface for binary upload. At the top, there's a callout pointing to the 'Hardware' dropdown which contains 'firefly'. Another callout points to the 'Bin file' input field containing 'connectivity.bin'. The interface includes fields for 'Targets' (set to 'all'), 'Programaddress' (set to '0x00200000'), and an 'Upload file' checkbox.

You have to specify this field
ONLY IF you work with Firefly.
Always use: **0x00200000**

On which nodes you want to upload the firmware.
Just write **all**, or **comma-separated IDs**
(e.g., 1,3,4,12,85) or, in DEPT, **disi_povo1**,
disi_povo2, **disi_bridge**.

Running a test with the CLOVES website

1. **Compile your code targeting a platform available in CLOVES.**

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.

2. **Log into the CLOVES interface.**

Start at research.iottestbed.disi.unitn.it, and log in with your credentials. Then navigate to “Create job”.

3. **Insert scheduling information.**

Area of the test, start time, duration.

4. **Upload the binary, specifying the target platform.**

Most of the time, you will only need one binary file. Choose the platform (DWM1001, EVB1000, Firefly) and specify (*comma-separated*) the node IDs you want to activate. You can put “all” here to select all nodes for the specific platform in the desired testbed area.

5. **Submit the job**

Submit job

If everything goes well, you will be redirected to a webpage like this one, where you can check your configuration again after submitting the job

Job has been successfully created!

Job details:

ID: 43831

Start date: 27 October 2025 at 17:38:00 (+0100)

End date: 27 October 2025 at 17:39:40 (+0100)

Generated configuration:

```
{  
  "island": "DEPT",  
  "start_time": "asap 2025-10-27 15:51",  
  "duration": 60,  
  "binaries": [  
    {  
      "hardware": "firefly",  
      "bin_file": "connectivity.bin",  
      "targets": [  
        "all"  
      ],  
      "programAddress": "0x00200000"  
    }  
  ]  
}
```



IF you decide to use the Python client, this is precisely the JSON file that you need to create and pass as argument to `iot_testbed_client.py`

 Go to dashboard

 Schedule another job

Running a test with the CLOVES website

1. Compile your code targeting a platform available in CLOVES.

In the previous labs, we used the Tmote Sky for Cooja simulations. To run an experiment on the testbed, the code must target one of its platforms instead.

2. Log into the CLOVES interface.

Start at research.iottestbed.disi.unitn.it, and log in with your credentials. Then navigate to “Create job”.

3. Insert scheduling information.

Area of the test, start time, duration.

4. Upload the binary, specifying the target platform.

Most of the time, you will only need one binary file. Choose the platform (DWM1001, EVB1000, Firefly) and specify (*comma-separated*) the node IDs you want to activate. You can put “all” here to select all nodes for the specific platform in the desired testbed area.

5. Submit the job.

6. Wait for the test to complete.

7. Download the output.

Download [STEP 7]

Private area

- [Profile](#)
- [Create Job](#)
- [Create Reservation](#)
- [Download Jobs](#)

If it is too early and the job is still running...

No completed jobs found

Otherwise ...

Jobs download

Job id	Begin	End	Area	
10223	10:16 p.m.	Nov. 1, 2023, 10:17 p.m.	DEPT	Download Delete

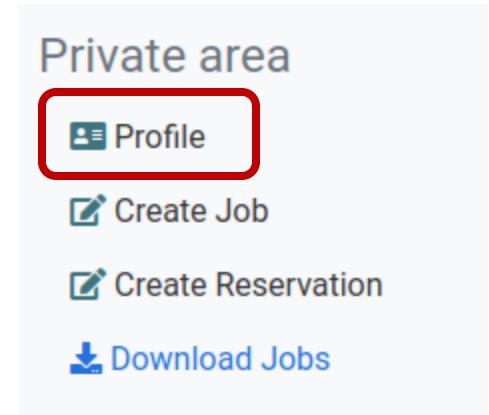
Finally get the logs!!!

Time budget

- In CLOVES, users belong to groups with different authorizations.
- Students can schedule jobs so that their total duration does not exceed **4h/week**. This should suffice to validate your implementation (already working in Cooja).
- You can see your time budget in your profile.
If you *really* need more time, just write to me!

Things to keep in mind:

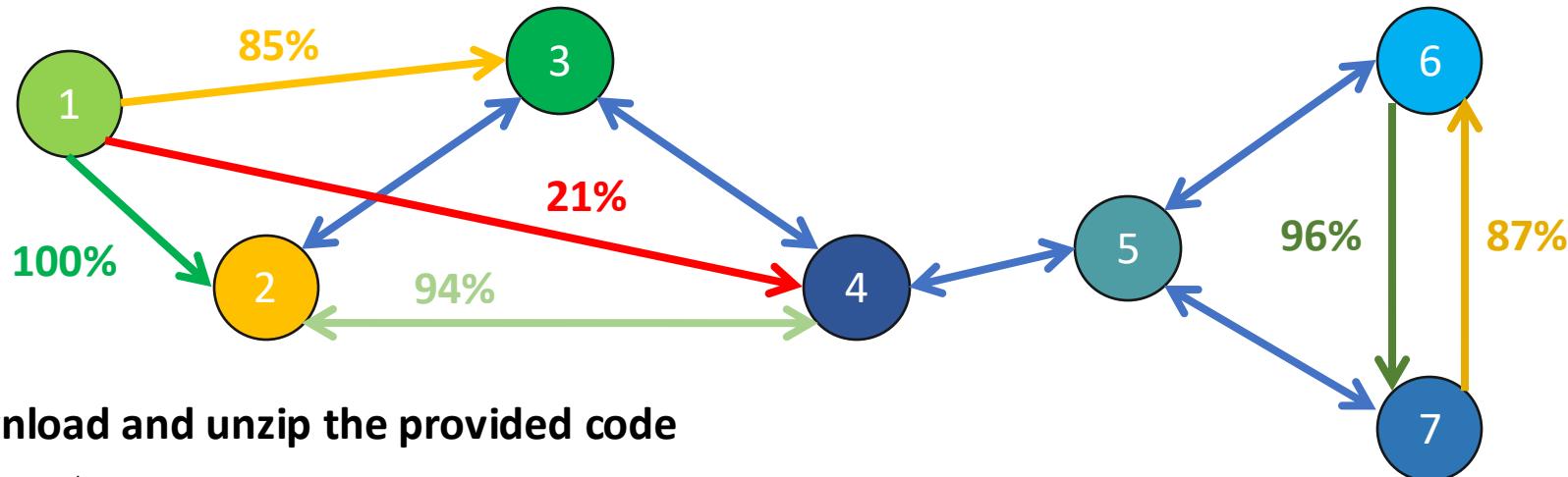
- The testbed might be reserved for research purposes by other users. Occasionally, it might be under maintenance ...
- **In other words: DO NOT WAIT till the last minute to run your testbed experiments!**



Exercise 1: Connectivity experiment

Goal: Discover all available links in the testbed and estimate their reliability

Motivation: Initial (crucial) step to characterize the testbed. It provides a baseline of sort for the expected packet reception rate (PRR) of all your future experiments.



Download and unzip the provided code

- `$ unzip connectivity-template.zip`

Go to the code directory

- `$ cd connectivity-template`

Compile:

- `$ make` (firmware, project-conf.h and Makefile are ready to use)

Upload and run in CLOVES - DEPT, 2-minute experiment:

- *If you are using the client*, experiment.json is already provided in the code directory.

Analysis of testbed logs

- Wait until your job is completed. Then download the archive and extract it!
- What we are interested in today is the **job.log** file
→ It contains the logs of all active nodes

While you wait for your job.log, you can find an example in the code folder

```
[2025-10-24 17:57:07,210] INFO:firefly.103: 103.firefly < b'TX 14:c3'  
[2025-10-24 17:57:07,211] INFO:firefly.105: 105.firefly < b'RX 14:c3->15:f3, RSSI = -100dBm'  
[2025-10-24 17:57:07,211] INFO:firefly.102: 102.firefly < b'RX 14:c3->16:5b, RSSI = -70dBm'  
[2025-10-24 17:57:07,211] INFO:firefly.100: 100.firefly < b'RX 14:c3->15:db, RSSI = -98dBm'  
[2025-10-24 17:57:07,211] INFO:firefly.32: 32.firefly < b'RX 14:c3->f2:91, RSSI = -96dBm'  
[2025-10-24 17:57:07,212] INFO:firefly.31: 31.firefly < b'RX 14:c3->de:af, RSSI = -94dBm'  
[2025-10-24 17:57:07,244] INFO:firefly.23: 23.firefly < b'TX f7:9a'  
[2025-10-24 17:57:07,246] INFO:firefly.22: 22.firefly < b'RX f7:9a->f3:88, RSSI = -85dBm'  
[2025-10-24 17:57:07,246] INFO:firefly.24: 24.firefly < b'RX f7:9a->f7:e7, RSSI = -87dBm'  
[2025-10-24 17:57:07,247] INFO:firefly.31: 31.firefly < b'RX f7:9a->de:af, RSSI = -83dBm'  
[2025-10-24 17:57:07,247] INFO:firefly.21: 21.firefly < b'RX f7:9a->de:e4, RSSI = -88dBm'  
[2025-10-24 17:57:07,314] INFO:firefly.132: 132.firefly < b'TX 15:b0'  
[2025-10-24 17:57:07,316] INFO:firefly.131: 131.firefly < b'RX 15:b0->15:87, RSSI = -91dBm'  
[2025-10-24 17:57:07,316] INFO:firefly.125: 125.firefly < b'RX 15:b0->15:bc, RSSI = -98dBm'  
[2025-10-24 17:57:07,316] INFO:firefly.127: 127.firefly < b'RX 15:b0->16:fe, RSSI = -86dBm'  
[2025-10-24 17:57:07,316] INFO:firefly.128: 128.firefly < b'RX 15:b0->15:f2, RSSI = -93dBm'
```

No GUI available ...
We need to **parse** and **process** the logs ourselves!

TODO: Assess the PRR between pairs of nodes and the average RSSI.

→ **PRR:** For each existing link, count **how many packets a node transmitted in broadcast and how many times they were received** by the other node.
E.g., Node A TX 10 times, node B RX 8 packets from A. PRR (A->B)% = 80%

Analysis of testbed logs

IF you have experience in data analysis, **try solving the exercise yourself** by analyzing the logs in `job.log`!

OTHERWISE, you can start from the `connectivity.py` script provided in the code folder. It

- takes the log file (`job.log`) as input,
- tries to match the expected TX and RX strings, and
- outputs the number of transmissions and receptions for each pair of nodes.



TODOs: Extend the `connectivity.py` script to show for each link

- the PRR % (instead of the raw number of successful transmissions and receptions);
- The average RSSI;

What is the relationship between PRR and RSSI?

Exercise 2: Run Lab 4 code in CLOVES

Adapt the code: You can't press buttons on the testbed. Therefore, let a single node (e.g., node 1 - F7:9C) initiate the chain of (unicast) messages by sending a message to a random neighbour every X seconds (modify the `button_process`).

Test your new code in Cooja (compile for Tmote Sky).

When you are confident about your solution:

1. Change Makefile and `project-conf.h`
2. Re-compile for the Zolertia platform (remember to make clean first)
3. [Client only] Create an appropriate `experiment.json`
4. Schedule a job on the testbed (e.g., in DEPT – `disi_povo2`)
5. [Optional] Parse and process the logs, e.g., try to understand if and when the chain of messages is interrupted (what is the average length?)

Hints:

- Use `linkaddr_cmp()` to determine whether a node should initiate the chain of unicast messages (only node 1 should)
- Check the code of Lab 3 to understand how to handle testbed and Cooja addresses in the same file:

```
static linkaddr_t unicast_chain_initiator = {{0xF7, 0x9C}};  
#if CONTIKI_TARGET_SKY  
// redefine unicast_chain_initiator ...
```