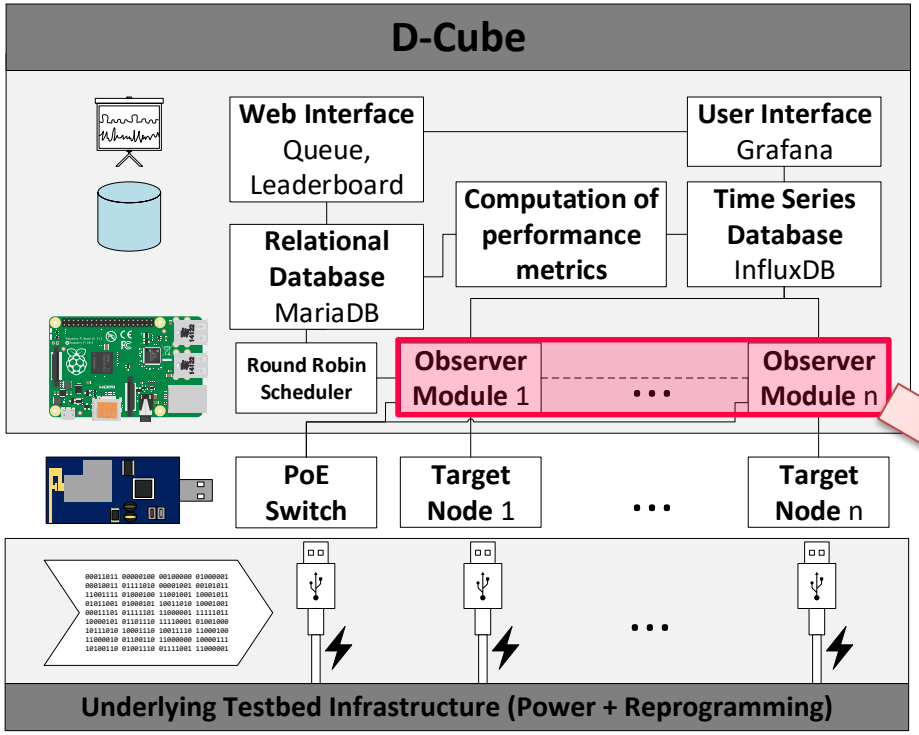
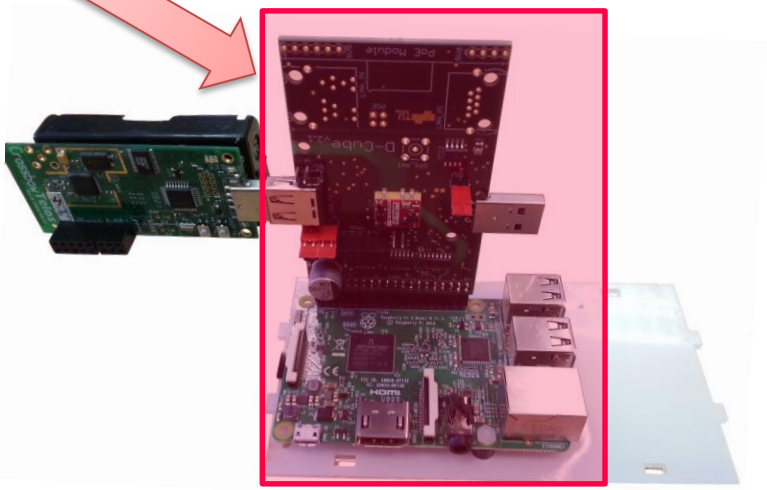
D-Cube:
Short Overview

More info:
http://iti.tugraz.at/d-cube

D-Cube: A Low-Cost Benchmarking Tool

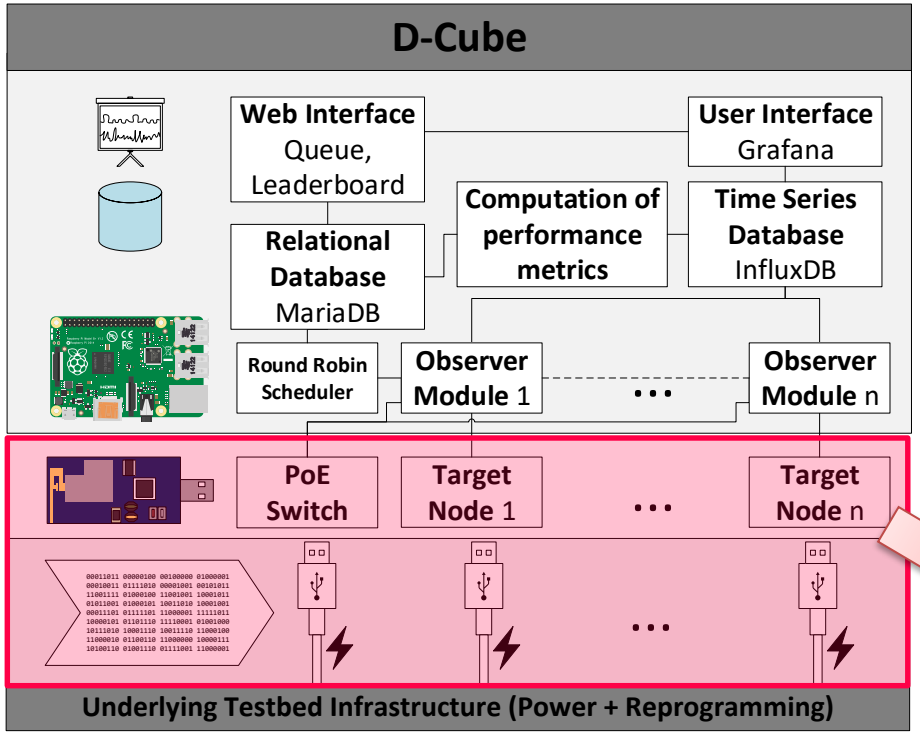


- Raspberry Pi3 as unobtrusive observers
  - Measuring in HW the reliability, latency, and energy efficiency
  - Open-source HW & SW

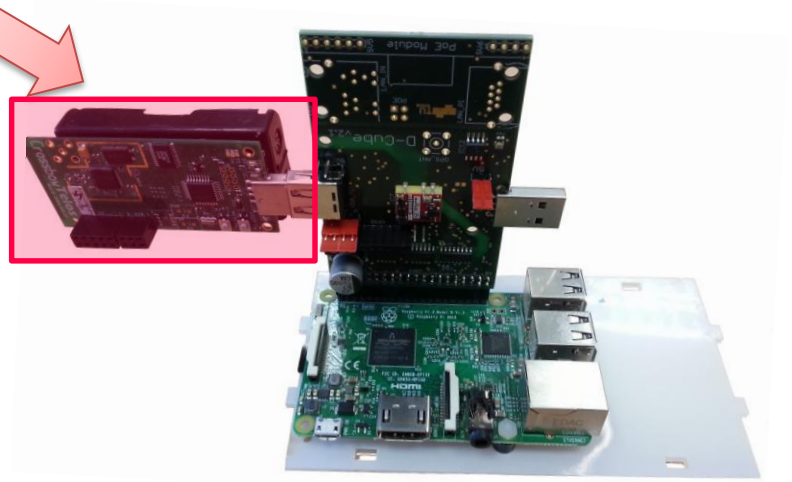


(enclosure and cabling removed)

D-Cube: A Low-Cost Benchmarking Tool



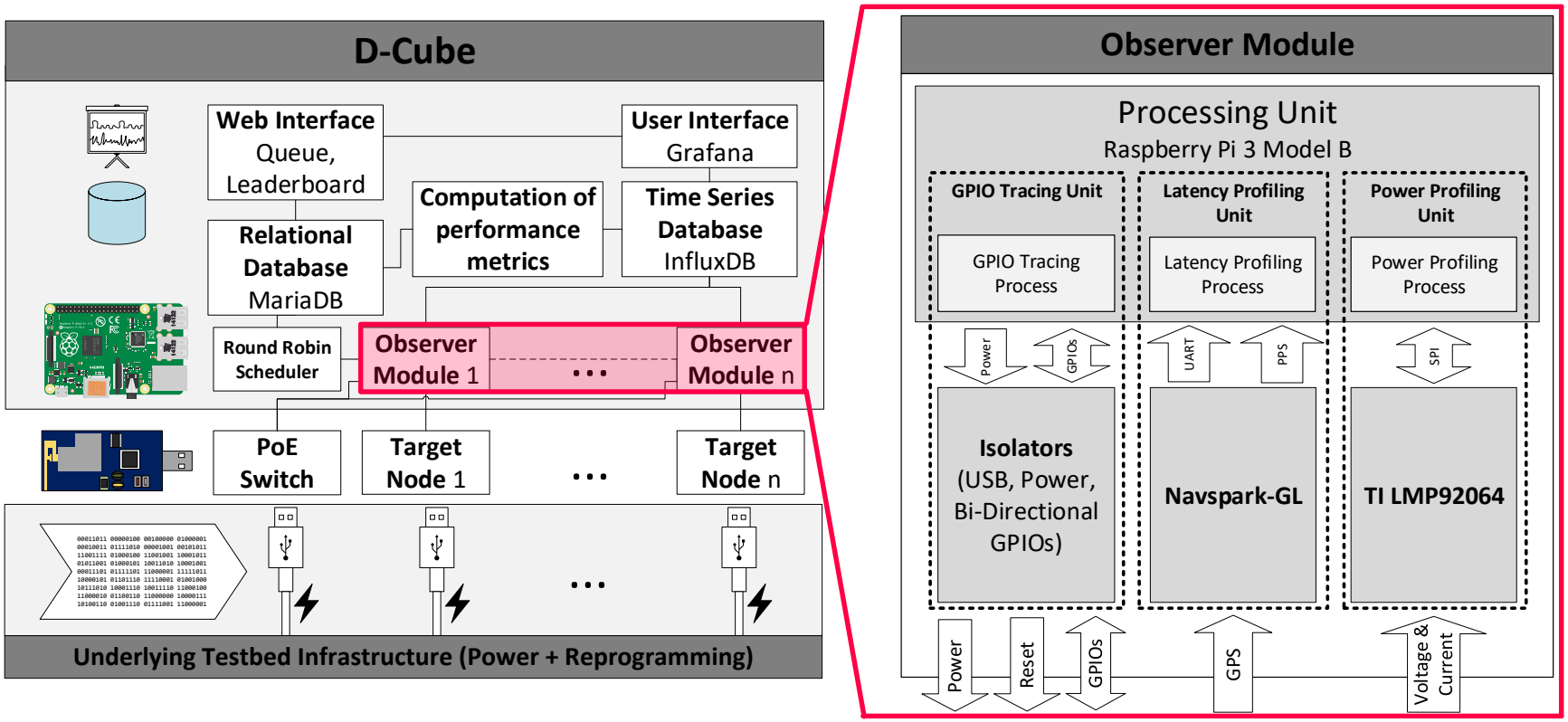
- Raspberry Pi3 as unobtrusive observers
  - Measuring in HW the reliability, latency, and energy efficiency
  - Open-source HW & SW



(enclosure and cabling removed)

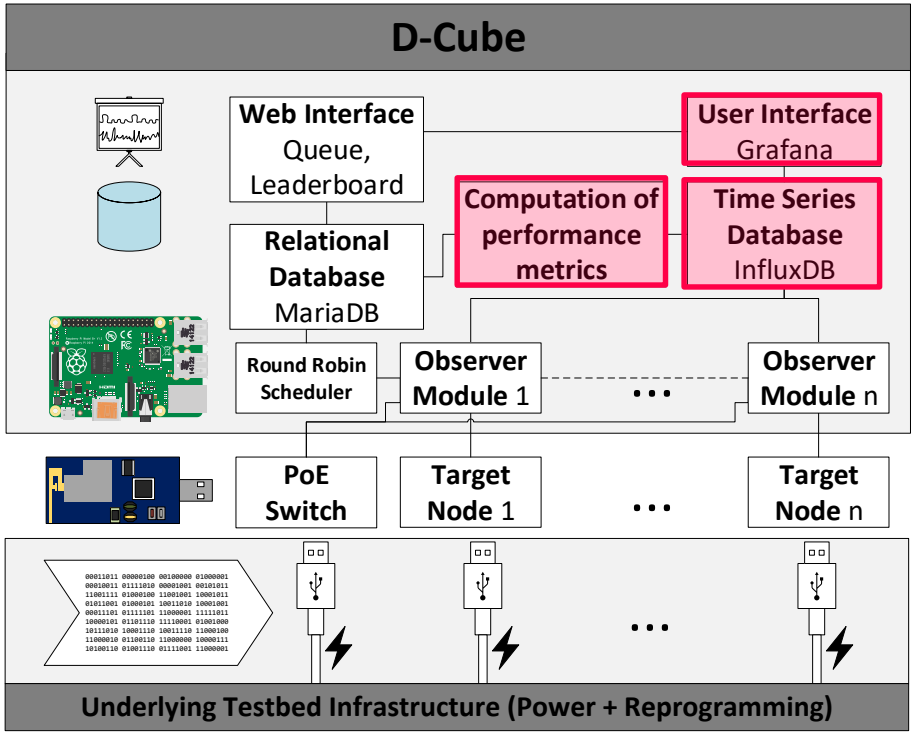
- Observers are agnostic to the target nodes
  - Any system under test
  - Powered / connected via USB

D-Cube: A Low-Cost Benchmarking Tool



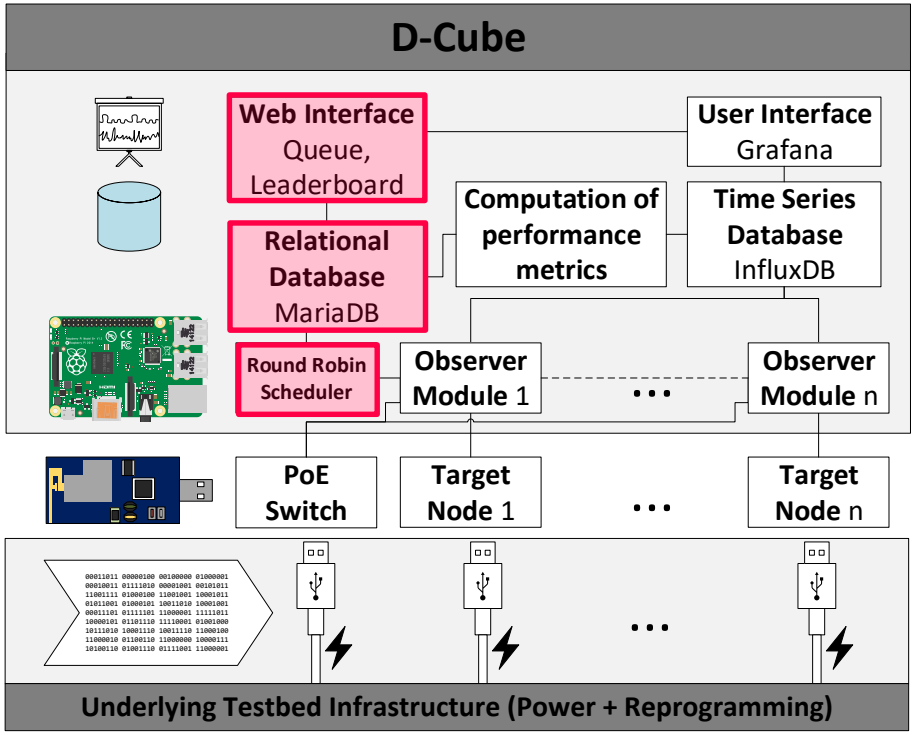
- Observer nodes feature:
  - → Power-profiling unit @125 kHz
  - → Latency-profiling unit for timestamping events
  - → GPIO tracing and actuation unit
    (to enable/disable nodes, vary network density, simulate battery failures)

D-Cube: A Low-Cost Benchmarking Tool



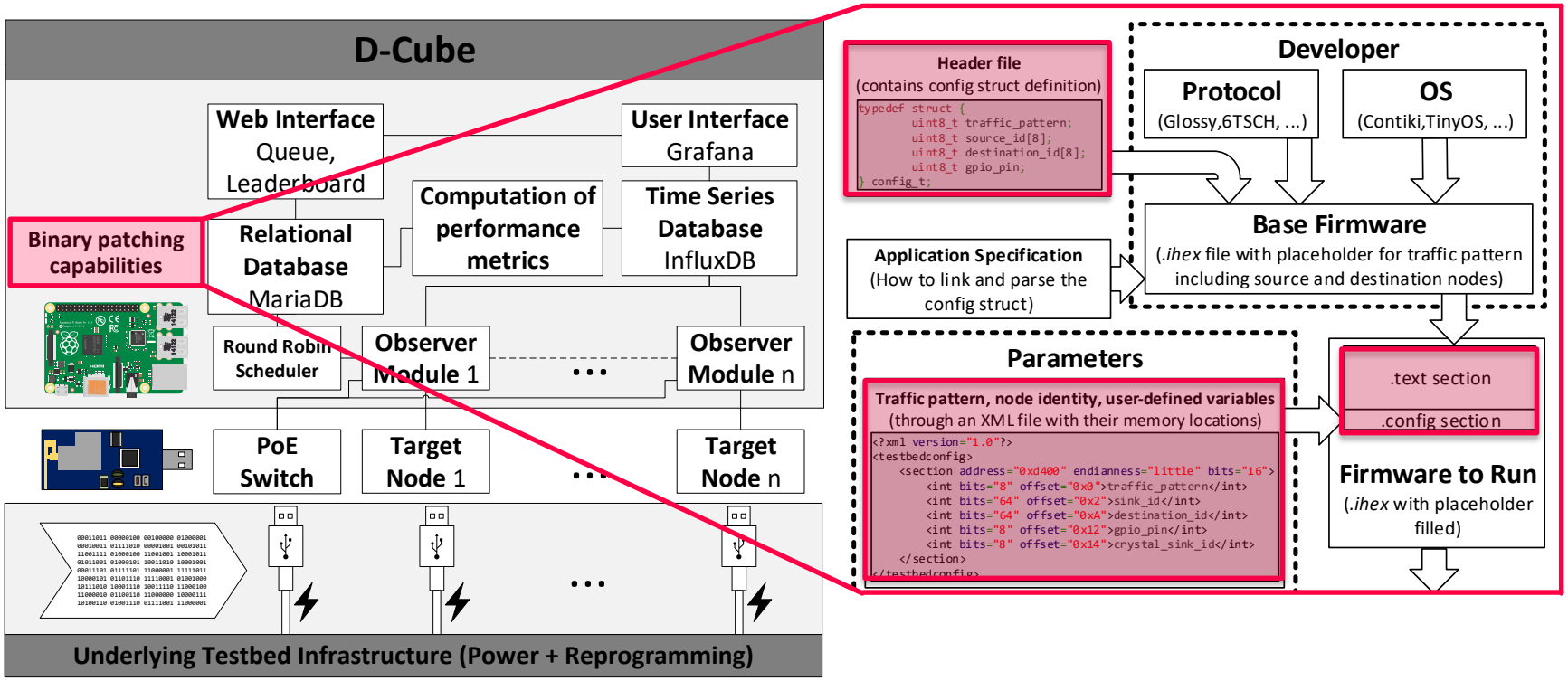
- Time Series database
  - → Collects and persistently stores the data from all observers
  - → InfluxDB (open-source)
  - → Nanosecond precision timestamps
- User Interface
  - → Acts as proxy to the database and gives real-time feedback

D-Cube: A Low-Cost Benchmarking Tool



- Time Series database
  - → Collects and persistently stores the data from all observers
  - → InfluxDB (open-source)
  - → Nanosecond precision timestamps
- User Interface
  - → Acts as proxy to the database and gives real-time feedback
- Management Web interface & scheduler
  - ReST API for automation
- Description of performance metrics

D-Cube: A Low-Cost Benchmarking Tool



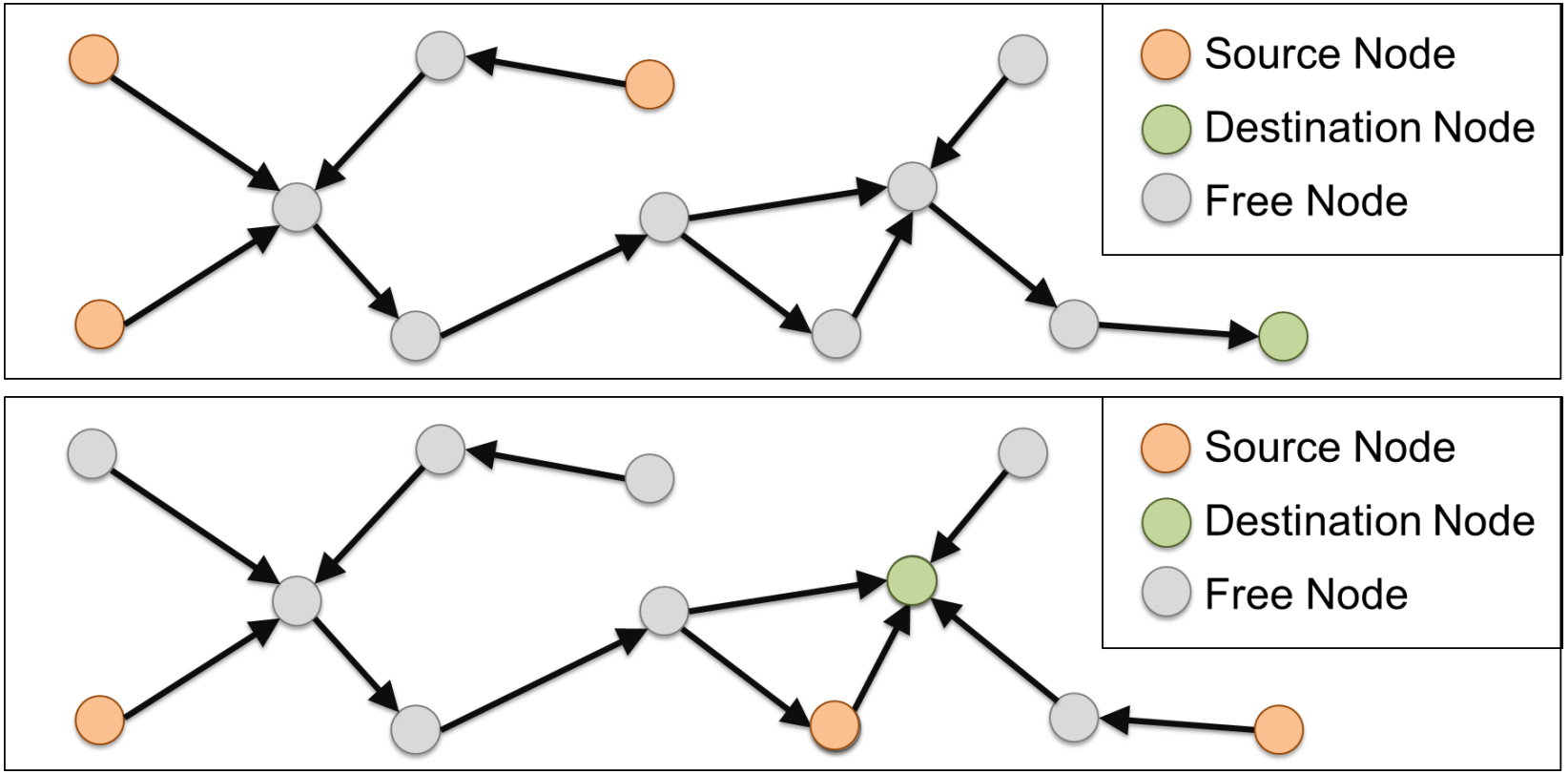
- Binary patching unit

  → The testbed can automatically path the firmware under test and insert values for pre-defined fields (e.g., node ID, payload length, ...)

  - Well-defined configuration `struct` provided as header file
  - Testbeds builds a `config` section with custom values

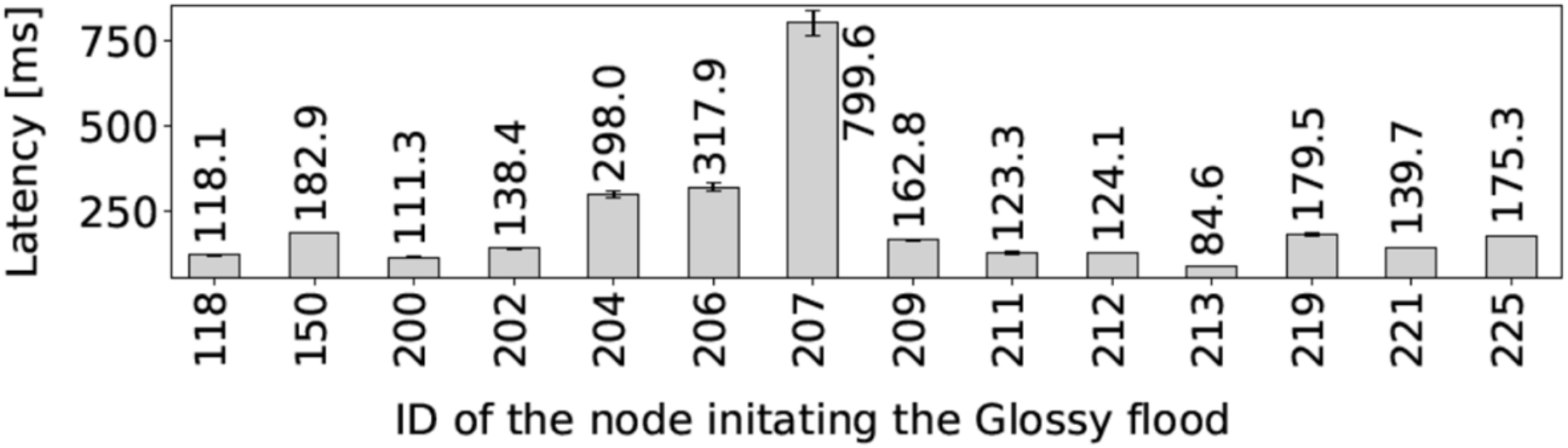
D-Cube: A Low-Cost Benchmarking Tool

- Why is binary patching useful?
  - Automatically testing different scenarios
  - Example: different source and destination nodes



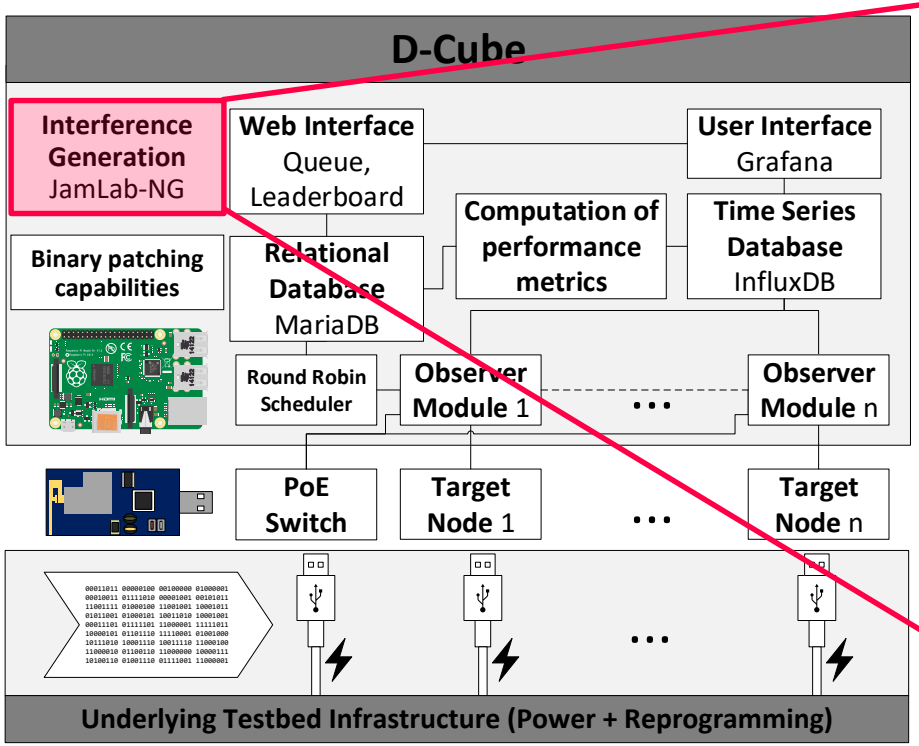
D-Cube: A Low-Cost Benchmarking Tool

- Why is binary patching useful?
  - Automatically testing different parameters
  - Example: testing the impact of a user-defined variable (`crystal_sink_id`)

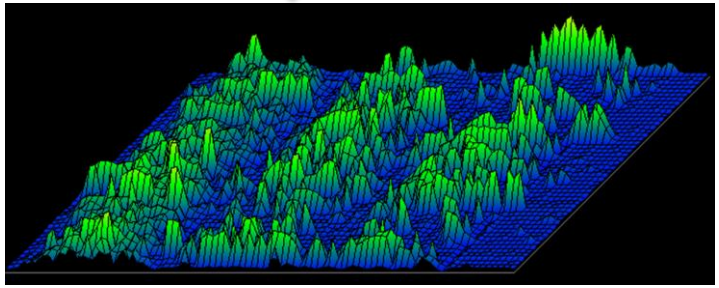


More info: M. Schuß, C.A. Boano, and K. Römer. **Moving Beyond Competitions: Extending D-Cube to Seamlessly Benchmark Low-Power Wireless Systems.** In Proc. of the 1st CPSBench Workshop. Porto, Portugal. April 2018.

D-Cube: A Low-Cost Benchmarking Tool



Reproducible & repeatable Wi-Fi interference



- JamLab-NG: generation of repeatable Wi-Fi interference
  - → Transmission of packets triggered within the radio module and full control of interference properties

More info:
http://www.iti.tugraz.at/JamLab-NG

JamLab-NG: Design Walkthrough

- Traditional packet generation tools (e.g., `iperf`) are unable to precisely control the transmission of packets

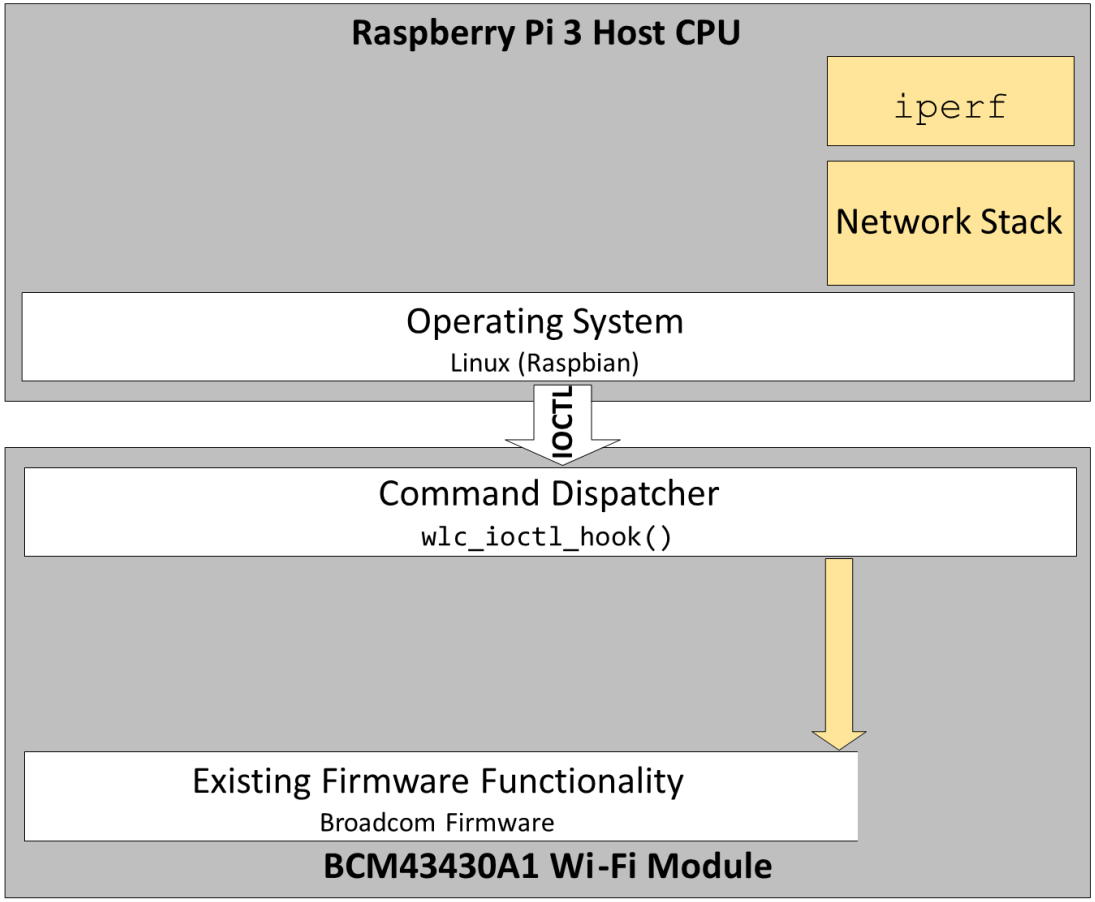
Send arbitrary raw frames ✗

No need of established connection to access point ✗

Configuration of PHY settings such as channel and transmission rate ✗

Disabling the CCA ✗

Controlling the delays introduced by OS and radio firmware ✗



JamLab-NG: Design Walkthrough

- Triggering the transmission of packets within the radio module and controlling the interference properties

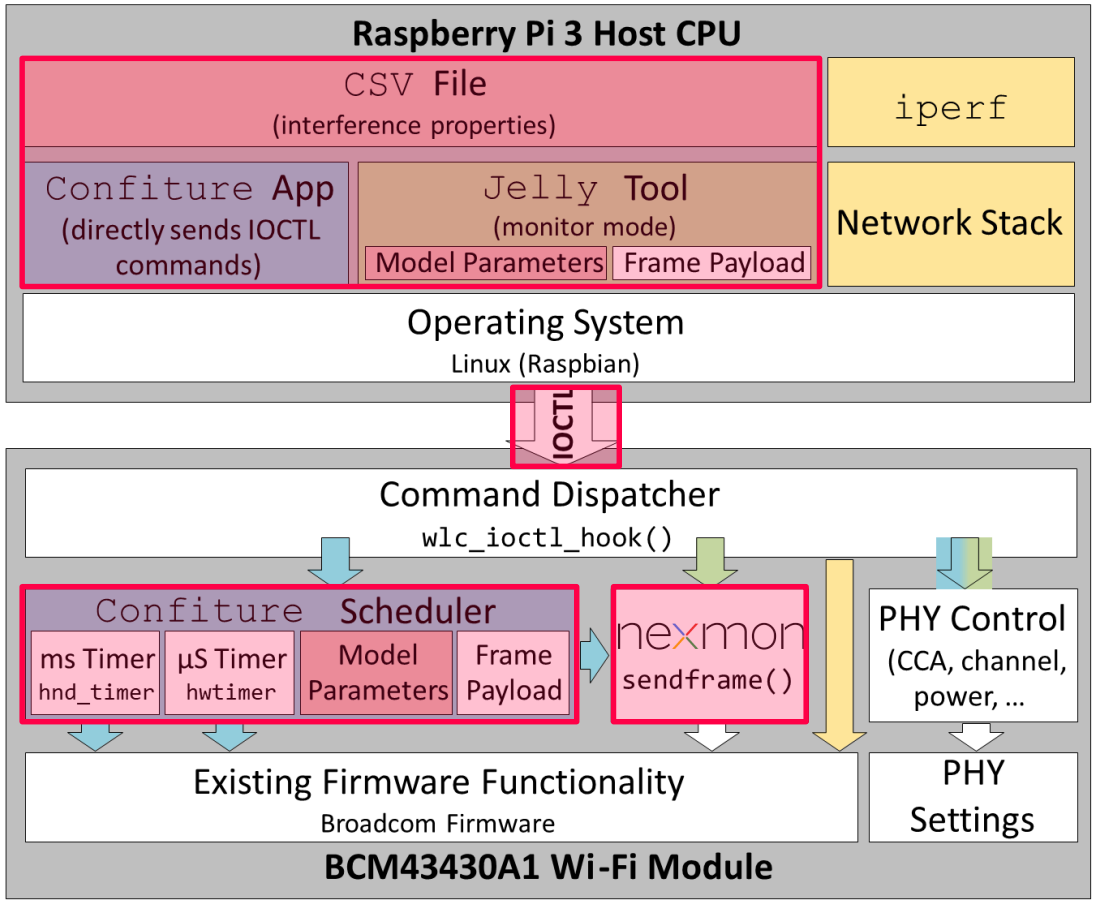
Send arbitrary raw frames ✓

No need of established connection to access point ✓

Configuration of PHY settings such as channel and transmission rate ✓

Disabling the CCA ✓

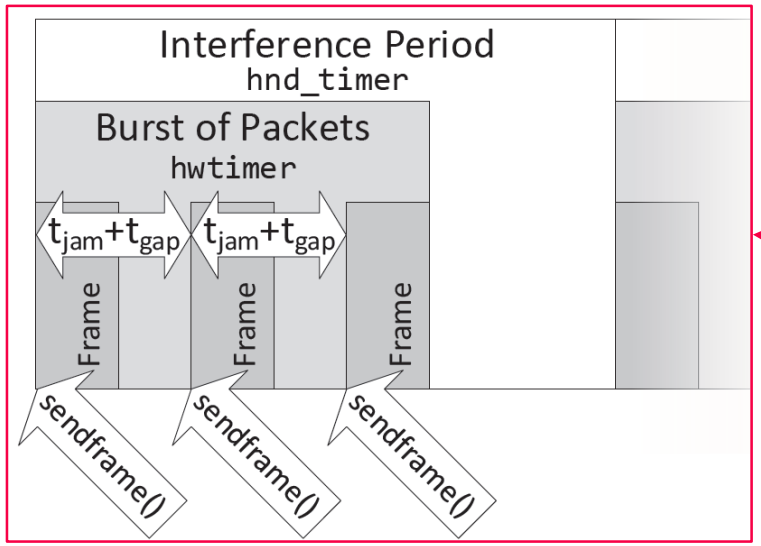
Controlling the delays introduced by OS and radio firmware ✓



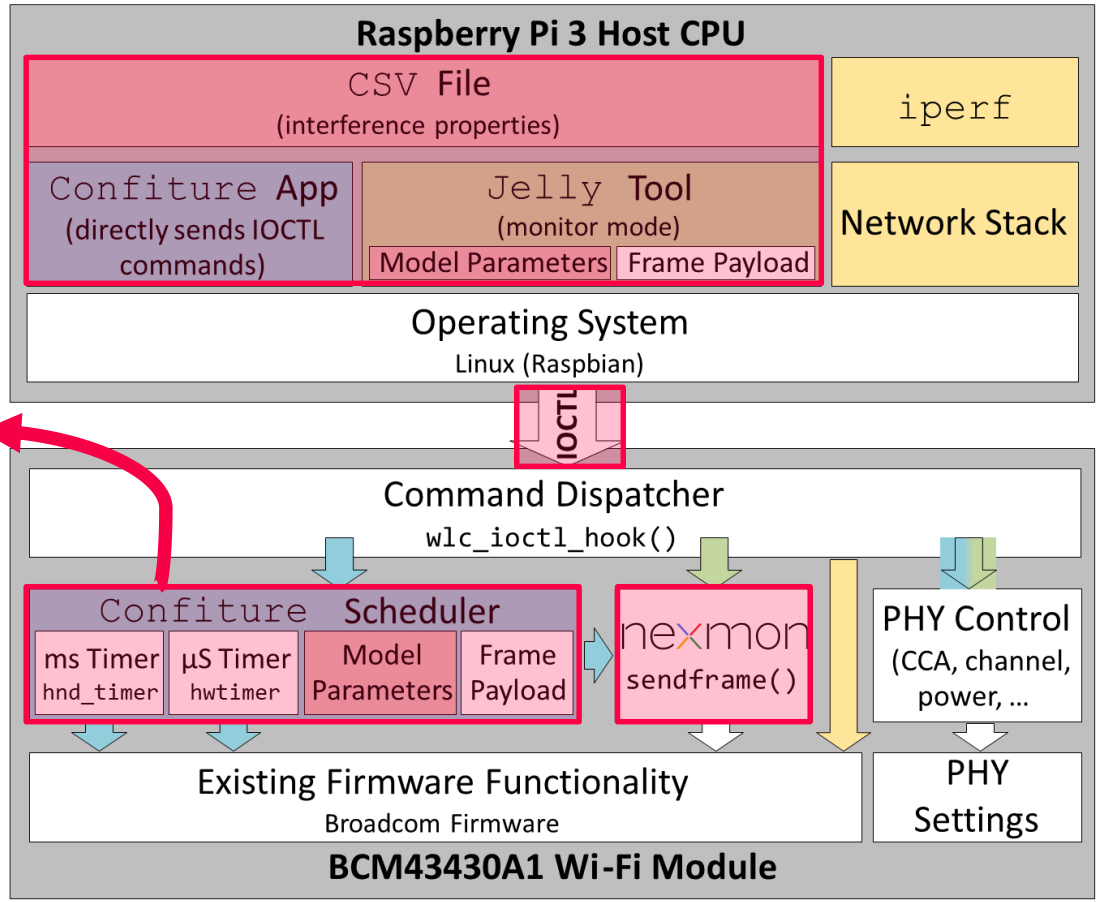
JamLab-NG: Design Walkthrough

- Triggering the transmission of packets within the radio module and controlling the interference properties

Building upon the nexmon tool
developed by SEEMOO @ TU Darmstadt

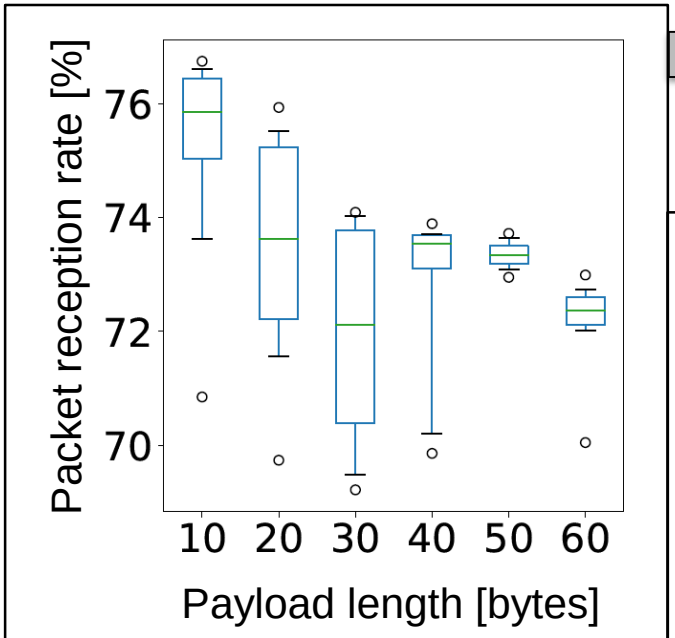


Controlling the delays introduced by OS and radio firmware



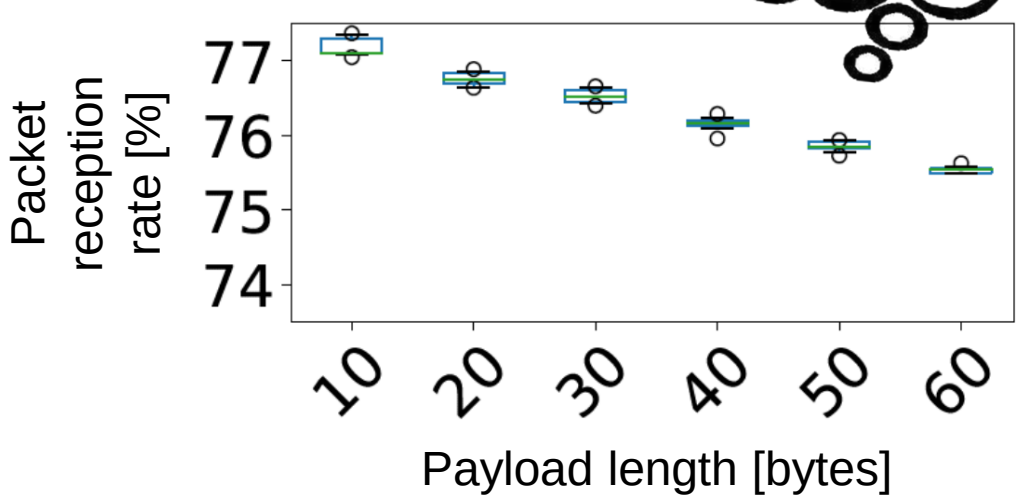
JamLab-NG: Aftermath

- Comparison to standard approaches
  (rate-limited TCP connection)



Standard approach (`iperf`)

Much better repeatability!
(± 0.2%)

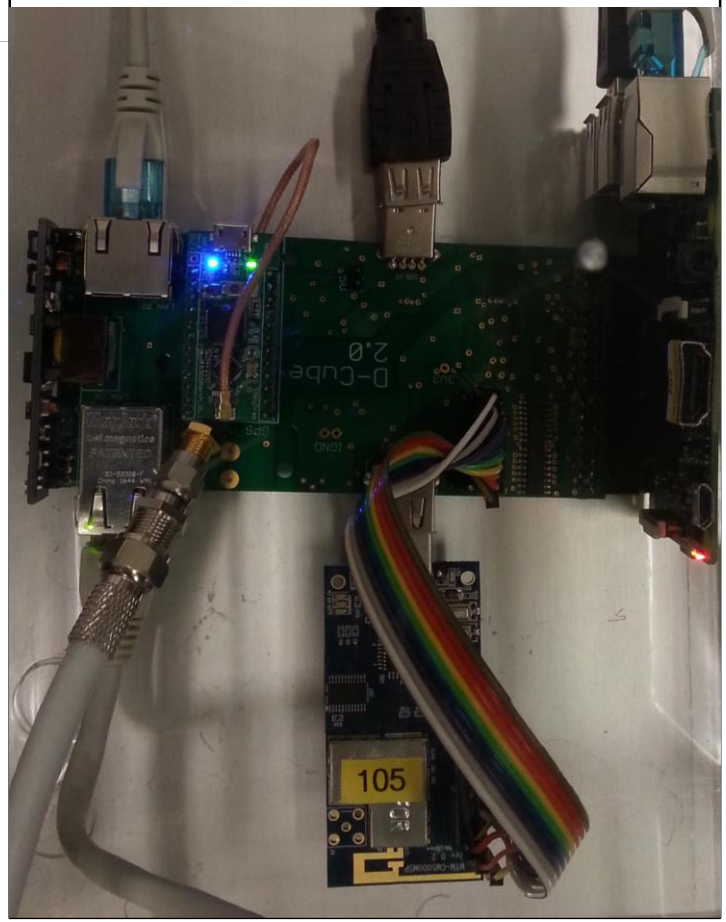


JamLab-NG

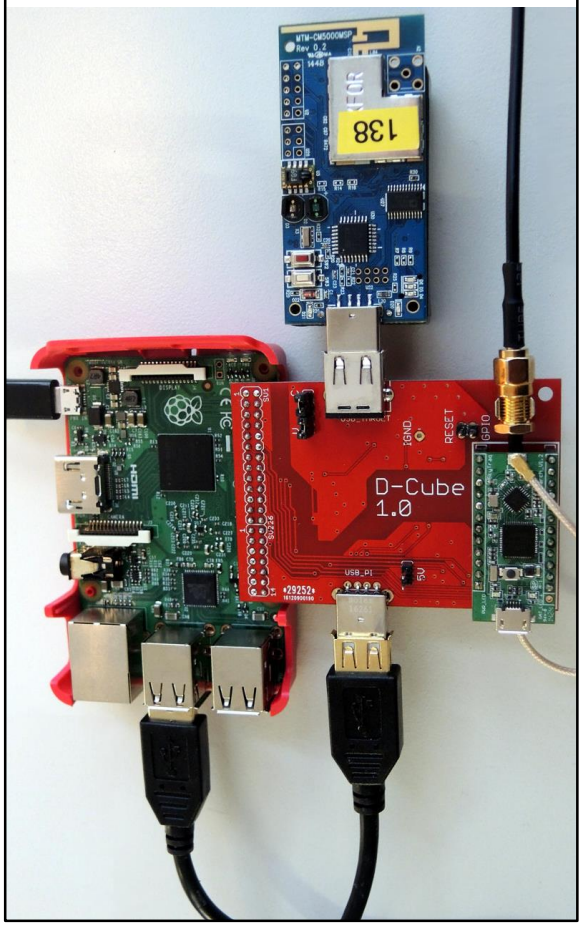
D-Cube: Hardware

- Continuously evolving

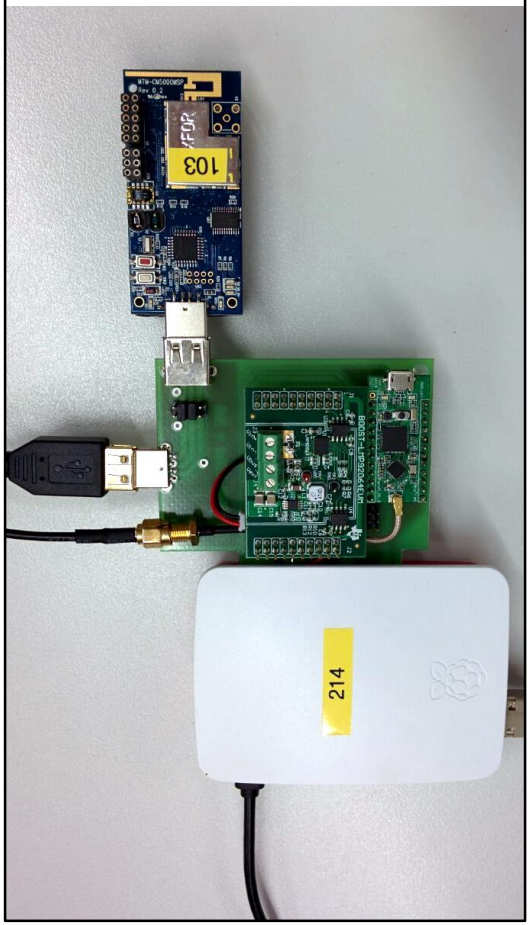
2018



2017



2016

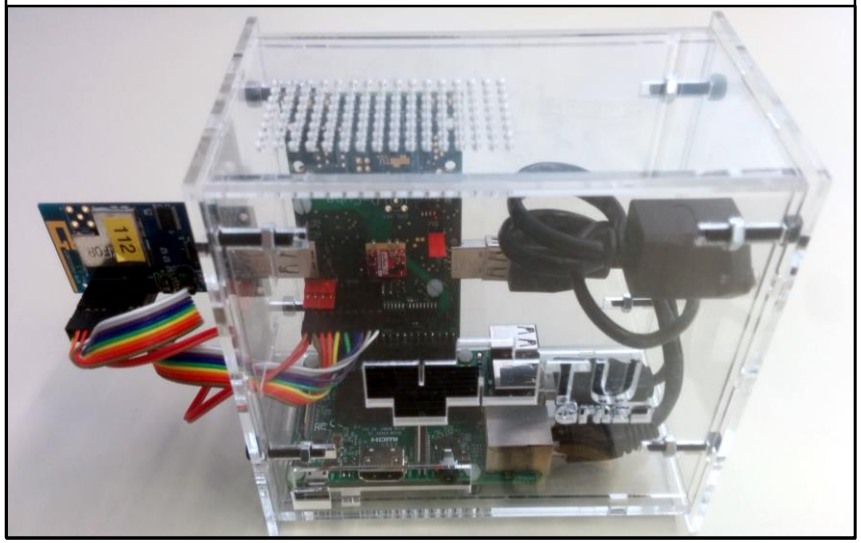


D-Cube: Hardware

- Continuously evolving

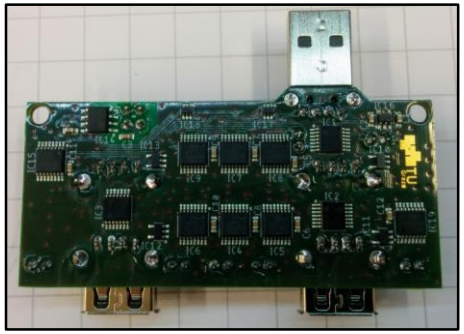
2019: D-Cube v2.1

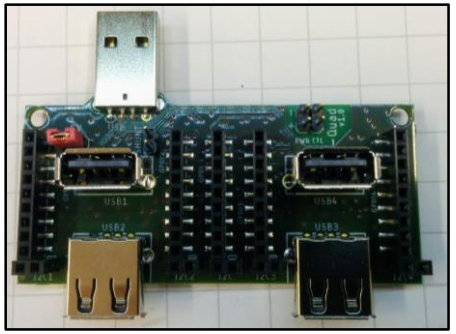
(now also available in Indriya2)



Latest extensions

Ability to control more target nodes at once



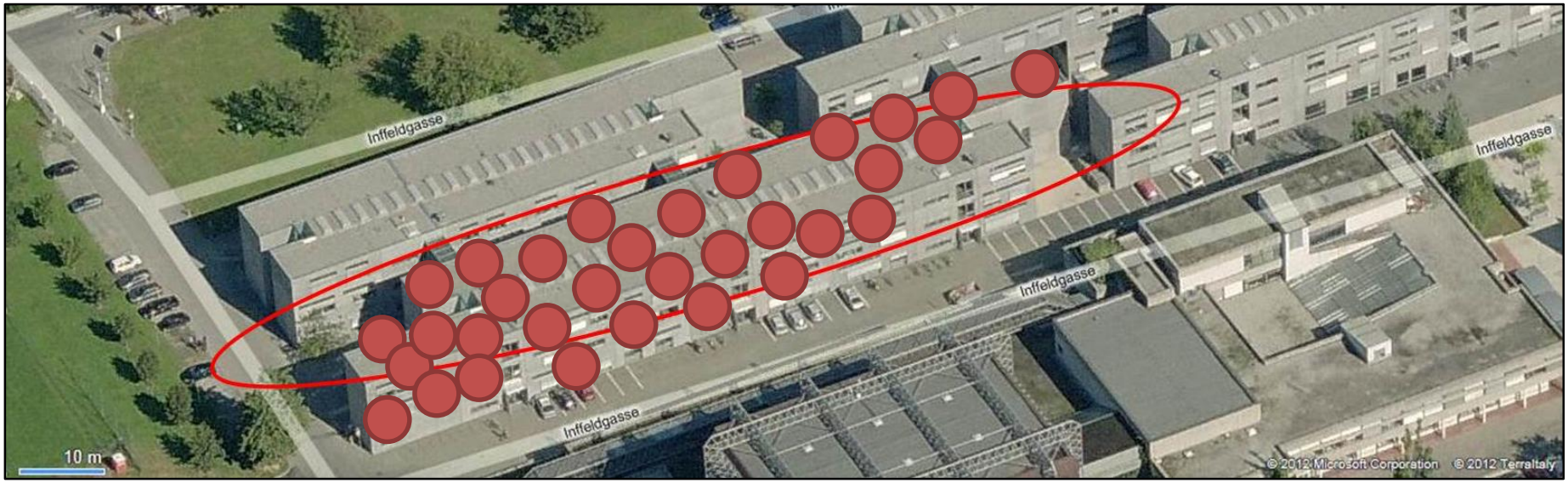


nRF52840-DK as new target nodes

BLE + IEEE 802.15.4

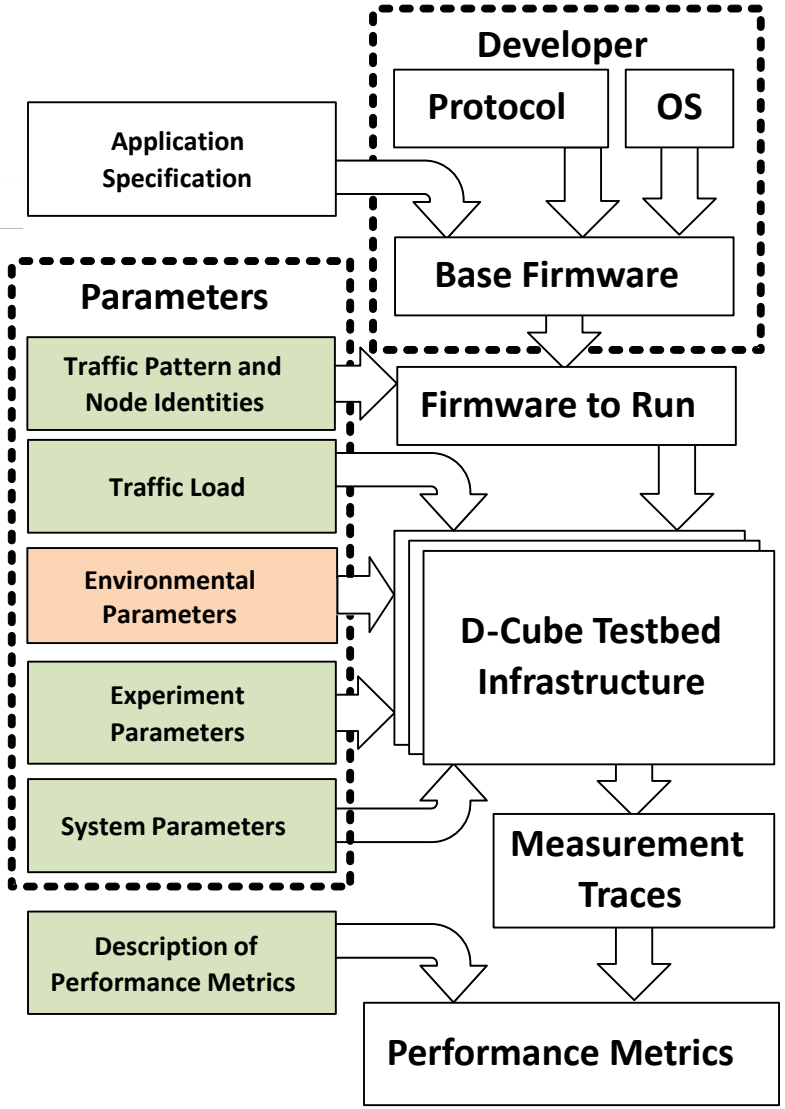
D-Cube: Hardware

- ~ 60 nodes deployed at our Institute (TU Graz, Austria)
  - Density of nodes varies across the building
  - Two floors, university offices, seminar rooms, and labs



More info: M. Schuß, C.A. Boano, M. Weber, and K. Römer. **A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge**. In Proceedings of the 14th EWSN Conference. 2017.

How does D-Cube help Benchmarking?



- Automated, seamless execution of experiments
- Configurable performance metrics
  - Packet reception rate
  - End-to-end latency (in hardware)
  - Energy consumption (in hardware)
- Configurable input parameters
  - Traffic parameters: load, pattern source and destination nodes
  - System parameters: network density
  - Experiment parameters: configurable number of repetitions
- Controllable environment parameters
  - Example: generation of radio interference (JamLab-NG)