Framework for PHY-MAC layers Prototyping in Dense IoT Networks

INRIA / INSA-Lyon / CITI lab

Othmane Oubeja, Jean-Marie Gorce, Diane Duchemin, Léonardo Cardoso

(on the behalf of the FIT/CorteXlab team)

Project funding

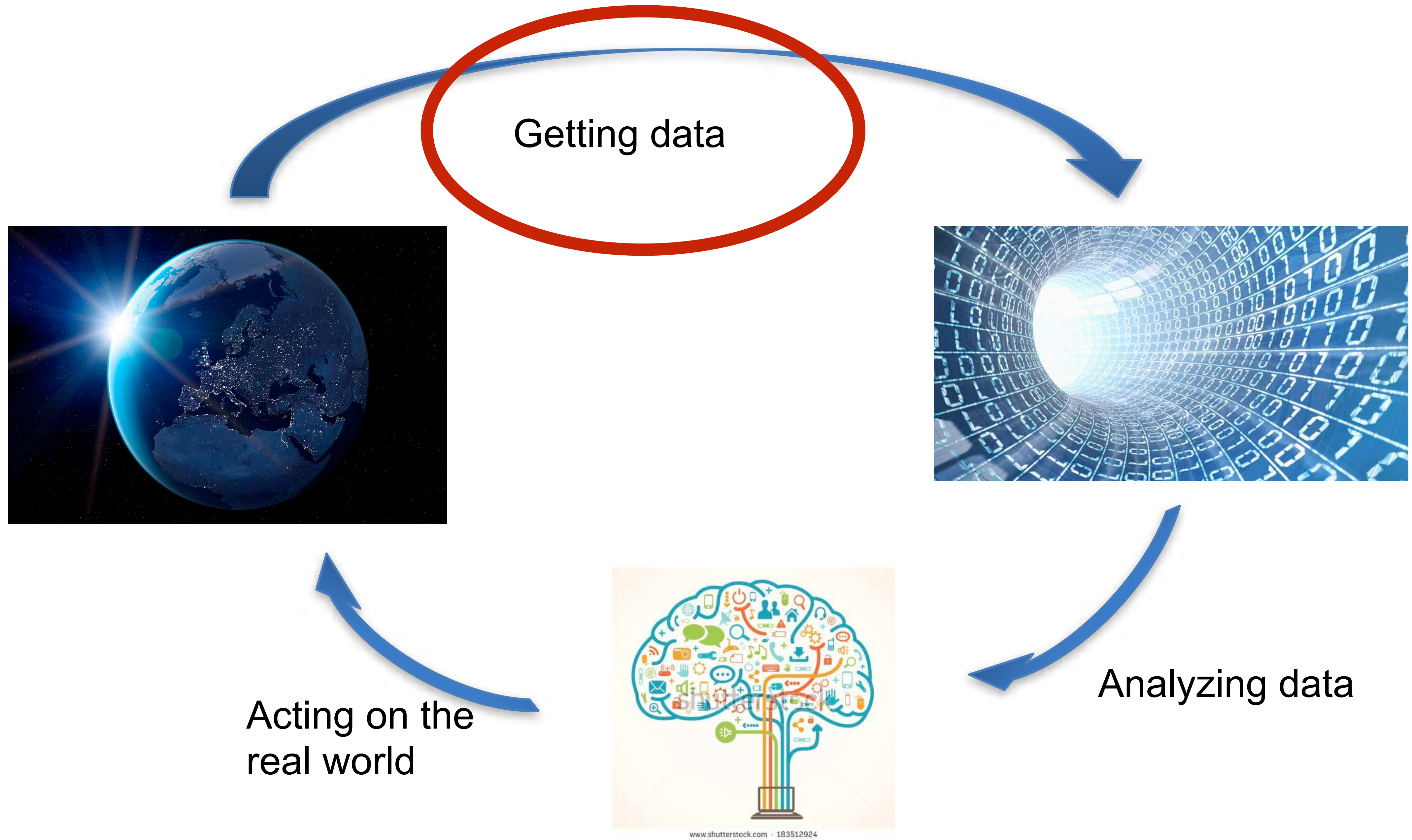
EquipEx 2009-2019 : for an experimental approach for networking

ANR 2016-2019 : new concepts for 5G and beyond

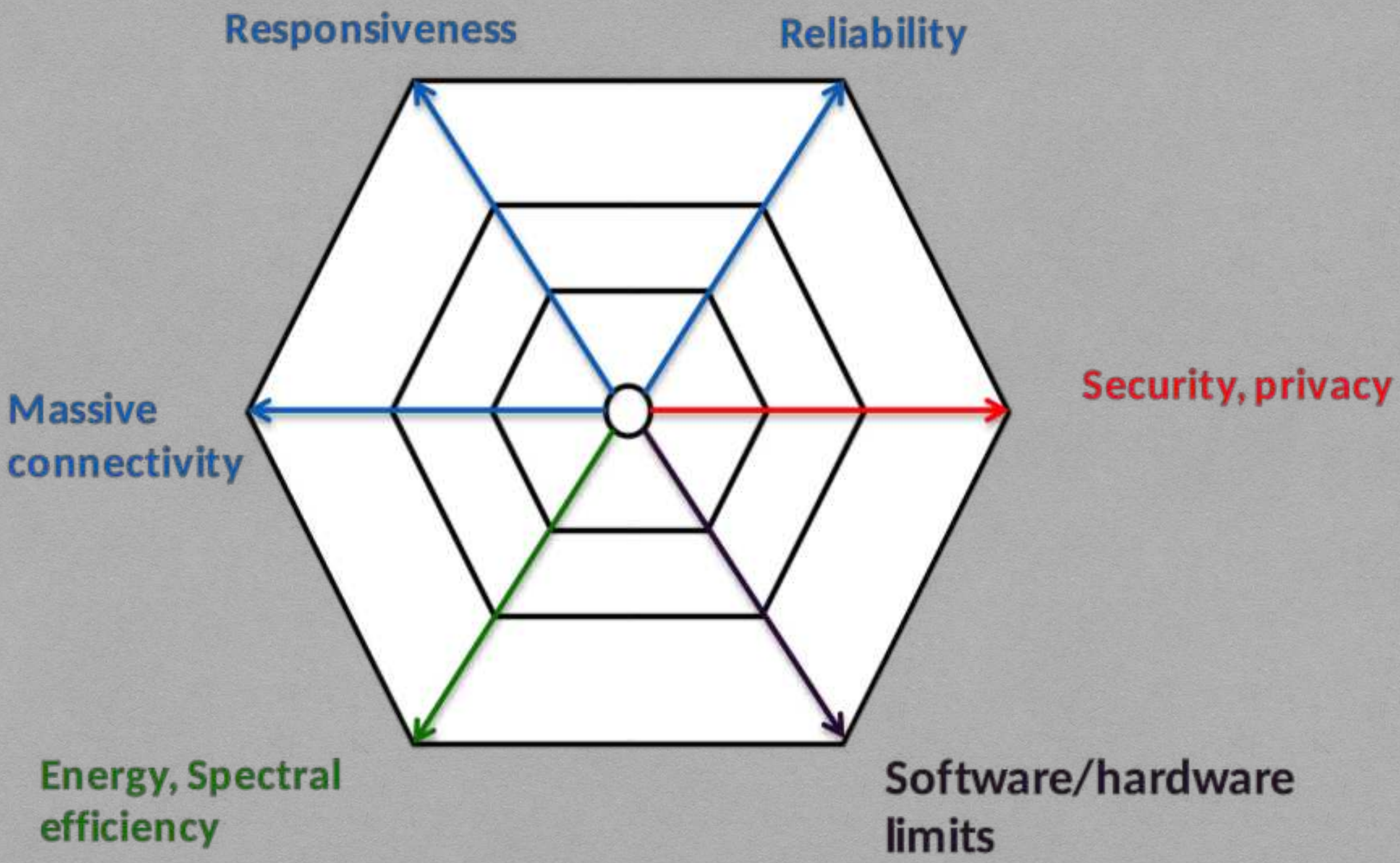
ANR 2016-2020 : a theoretical framework for IoT massive access

Massive Access in IoT : challenges

« Toward a reactive network »



Massive Access in IoT : figures of merit



Massive Access in IoT : challenges

i- Massive access : many nodes may transmit in a bursty manner

—> Spontaneous multiple access

ii- Information granularity : small packets

—> Zero-overhead transmissions

iii- Reliability - Latency tradeoff

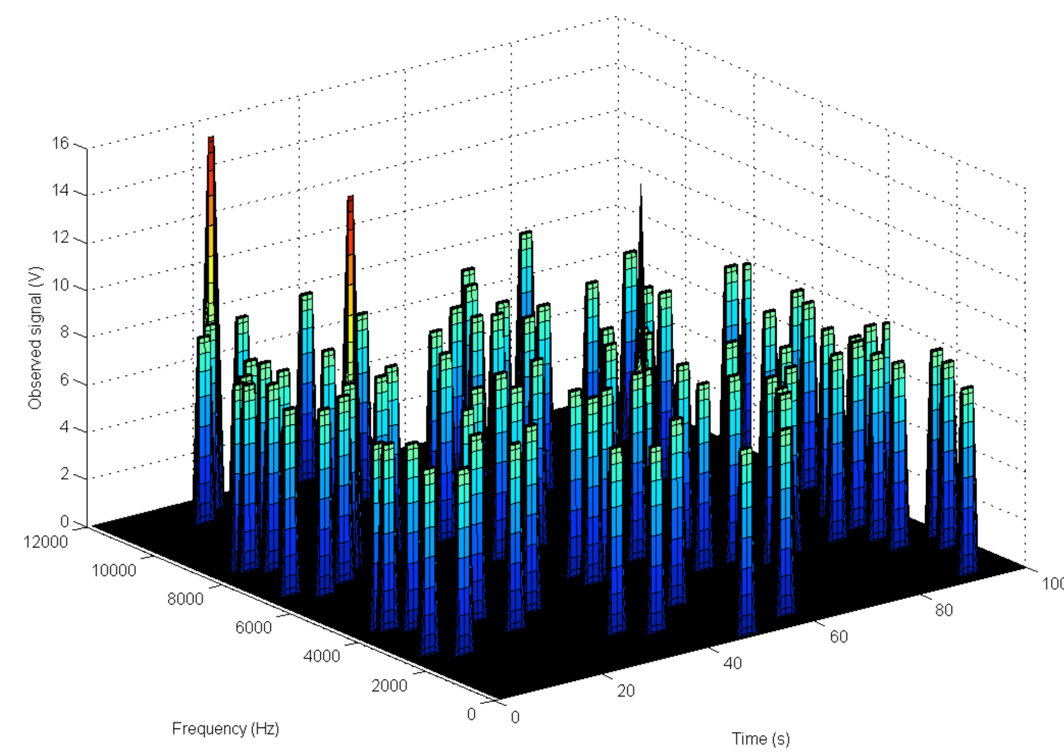
—> to have your cake and eat it too

iv- Non stationary process: load and channel conditions

—> learning, learning, learning - exploration vs exploitation

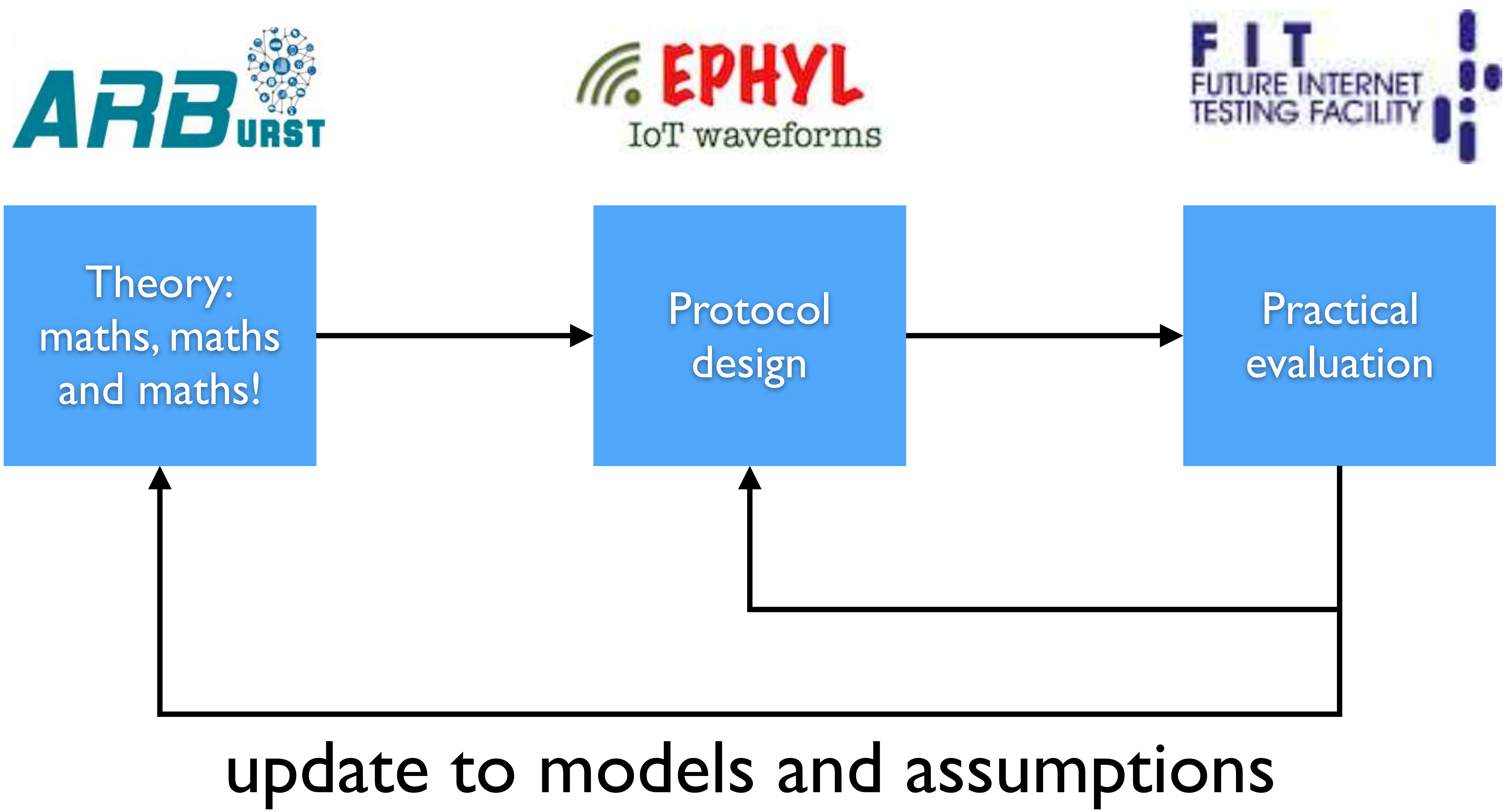
Some technologies

	SIGFOX	LoRa	clean slate cIoT	NB LTE-M Rel. 13	LTE-M Rel. 12/13	EC-GSM Rel. 13	5G (targets)
Range (outdoor) MCL	<13km 160 dB	<11km 157 dB	<15km 164 dB	<15km 164 dB	<11km 156 dB	<15km 164 dB	<15km 164 dB
Spectrum Bandwidth	Unlicensed 900MHz 100Hz	Unlicensed 900MHz <500kHz	Licensed 7-900MHz 200kHz or dedicated	Licensed 7-900MHz 200kHz or shared	Licensed 7-900MHz 1.4 MHz or shared	Licensed 8-900MHz 2.4 MHz or shared	Licensed 7-900MHz shared
Data rate	<100bps	<10 kbps	<50kbps	<150kbps	<1 Mbps	10kbps	<1 Mbps
Battery life	>10 years	>10 years	>10 years	>10 years	>10 years	>10 years	>10 years
Availability	Today	Today	2016	2016	2016	2016	beyond 2020



[Goursaud & Gorce, EAI Endorsed transactions 2016]

A virtuous approach



Framework designed in Ephyl project

EPHYL
IoT waveforms

Three key elements

Keystone elements for massive access

(star based architecture)

to be addressed jointly

- PHY layer : What is the best solution for non orthogonal bursty communications and MU receivers ?
- MAC layer : What are the best direct access mechanisms allowing to avoid any handshake mechanisms ?
- Link layer : What kind of learning can be implemented ? try and learn

The approach

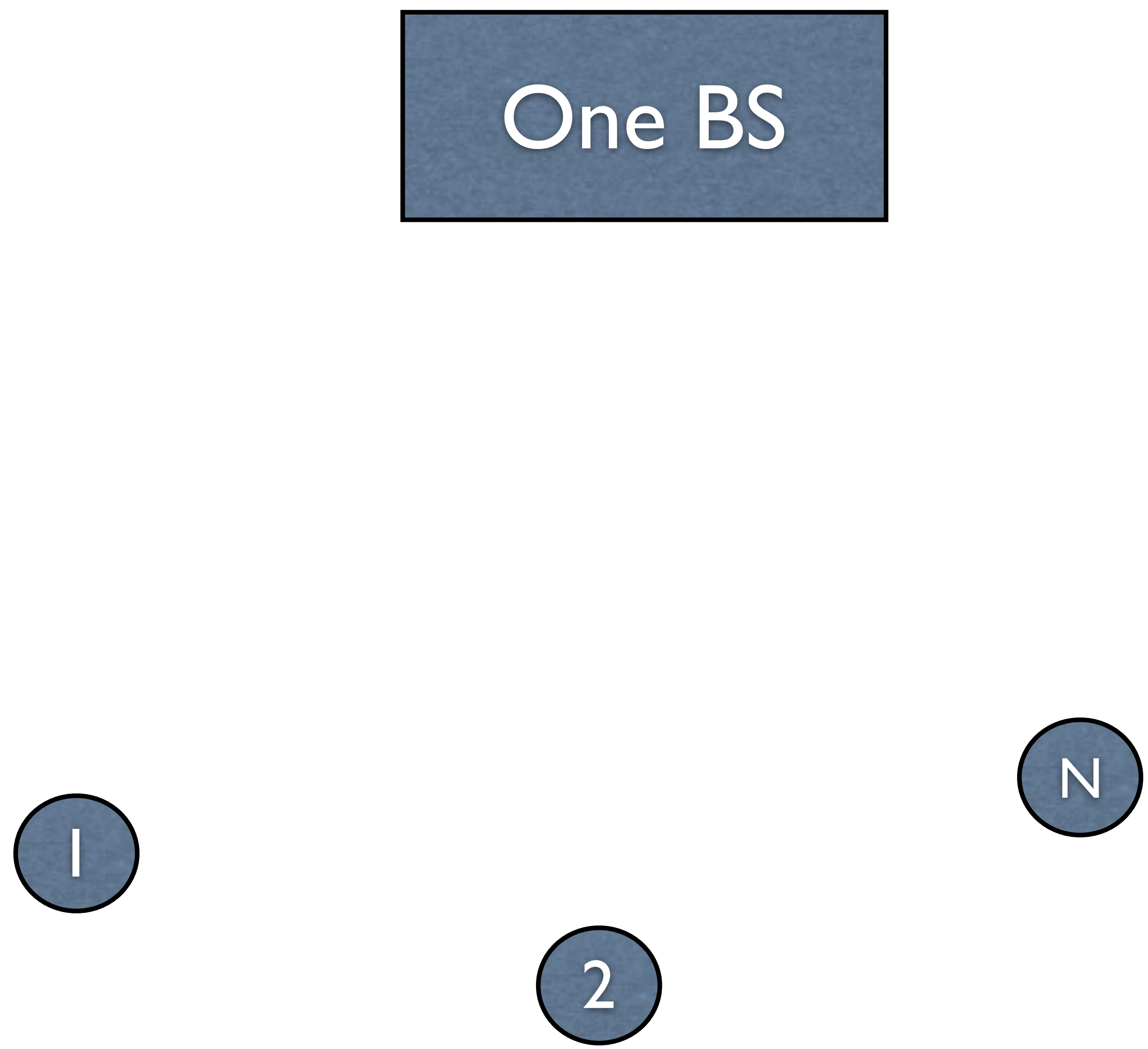
Experimental based assessment of a complete system with

- Fast Test and Try environment —> **remote access**
- Reproducible environment —> **interference free**
- Adaptable to almost any PHY layer —> **software radio**

Desired Architecture

Features

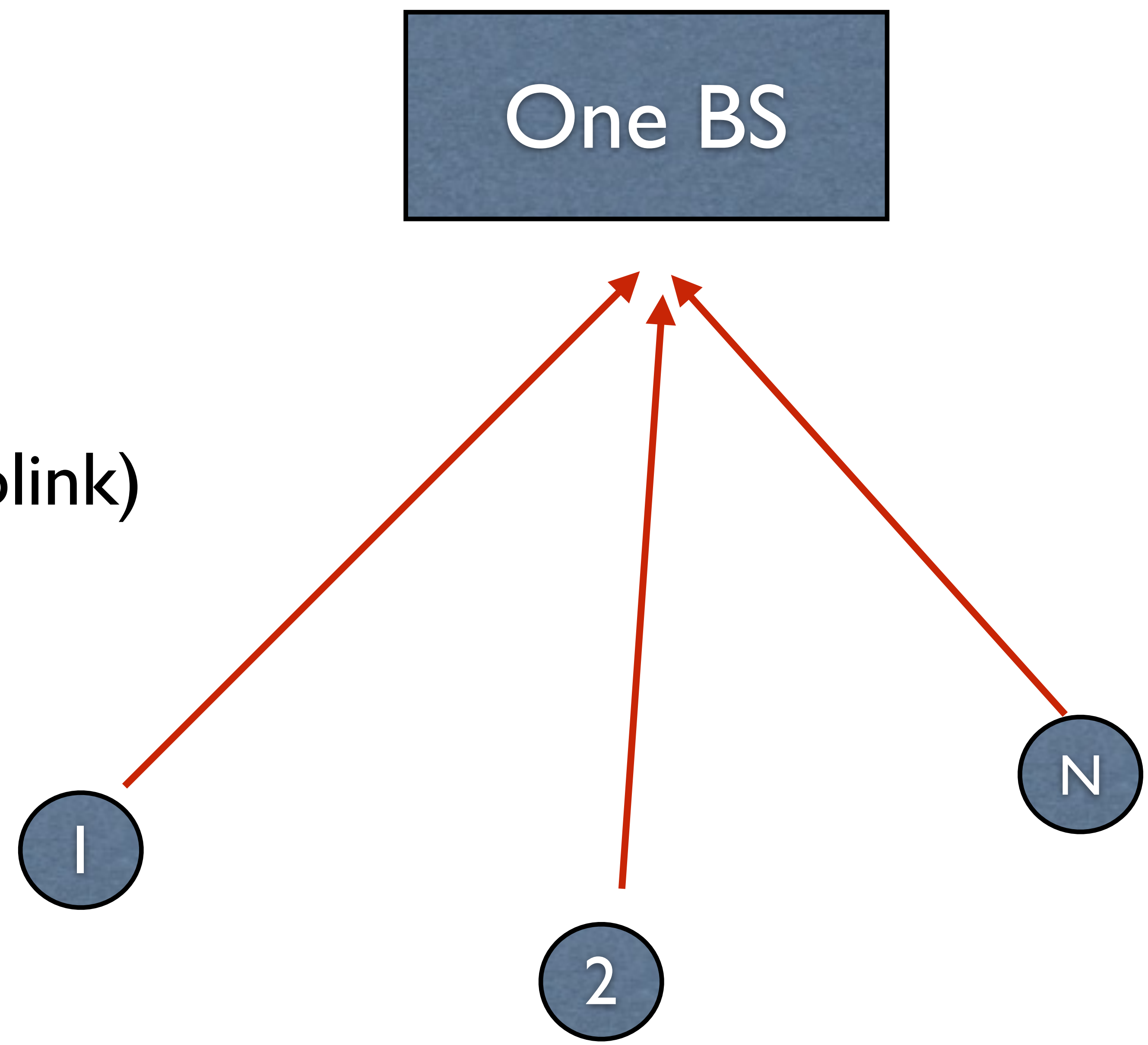
- 1 BS and N nodes (software radio based)



Desired Architecture

Features

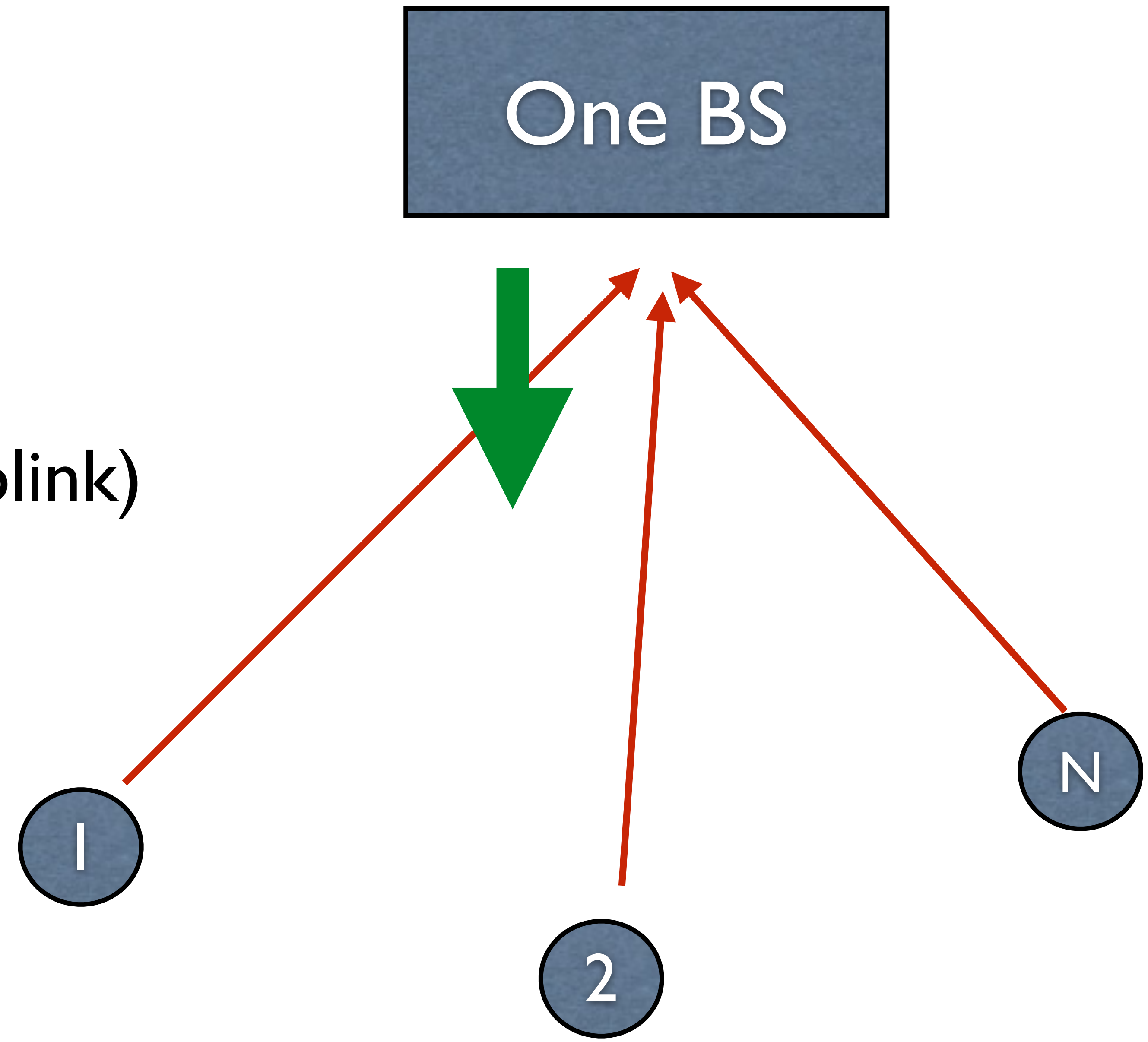
- 1 BS and N nodes (software radio based)
- A multi-bands multi-slots multiple access channel (uplink)



Desired Architecture

Features

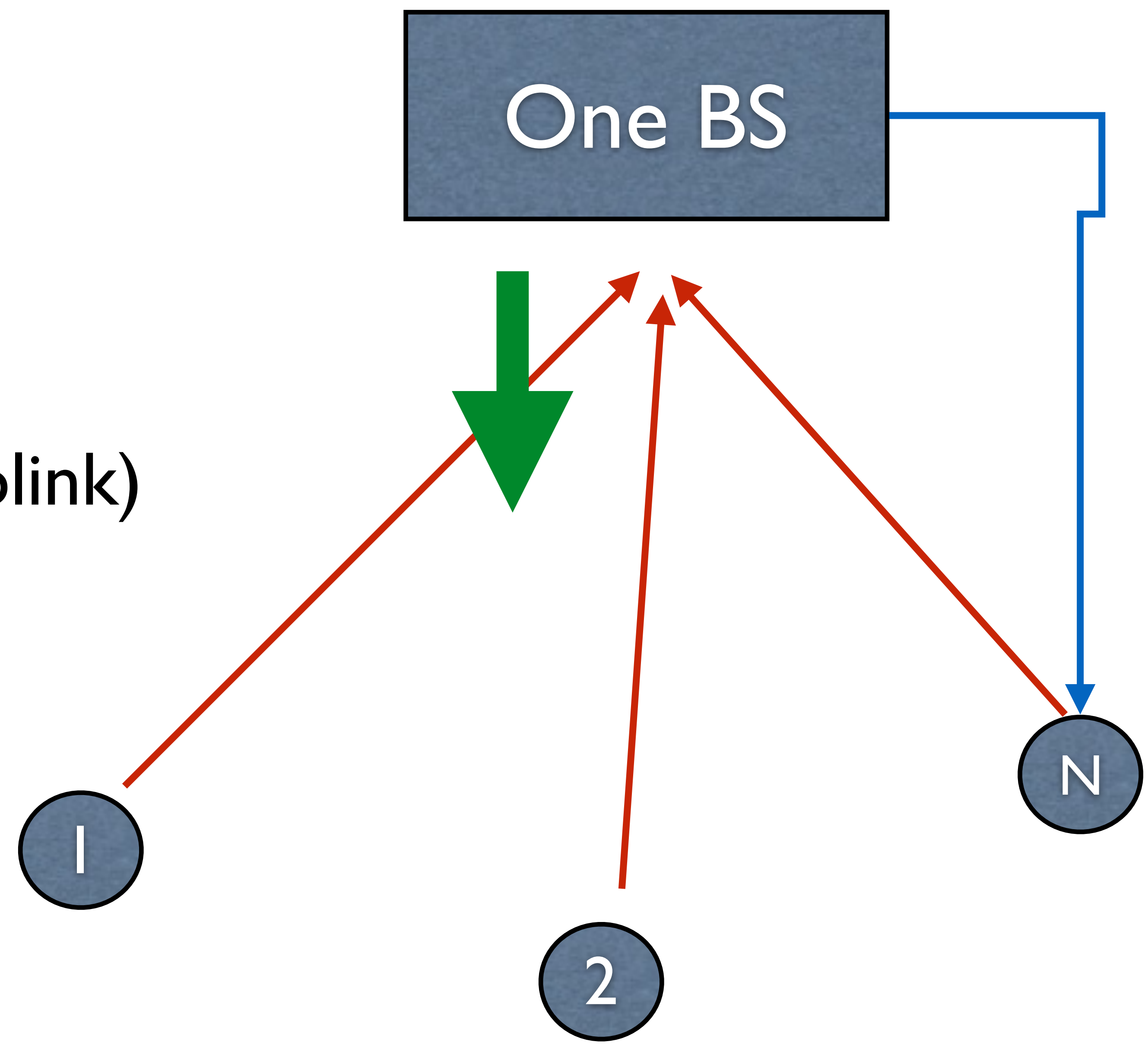
- 1 BS and N nodes (software radio based)
- A multi-bands multi-slots multiple access channel (uplink)
- A broadcast channel for Synch and beaconing



Desired Architecture

Features

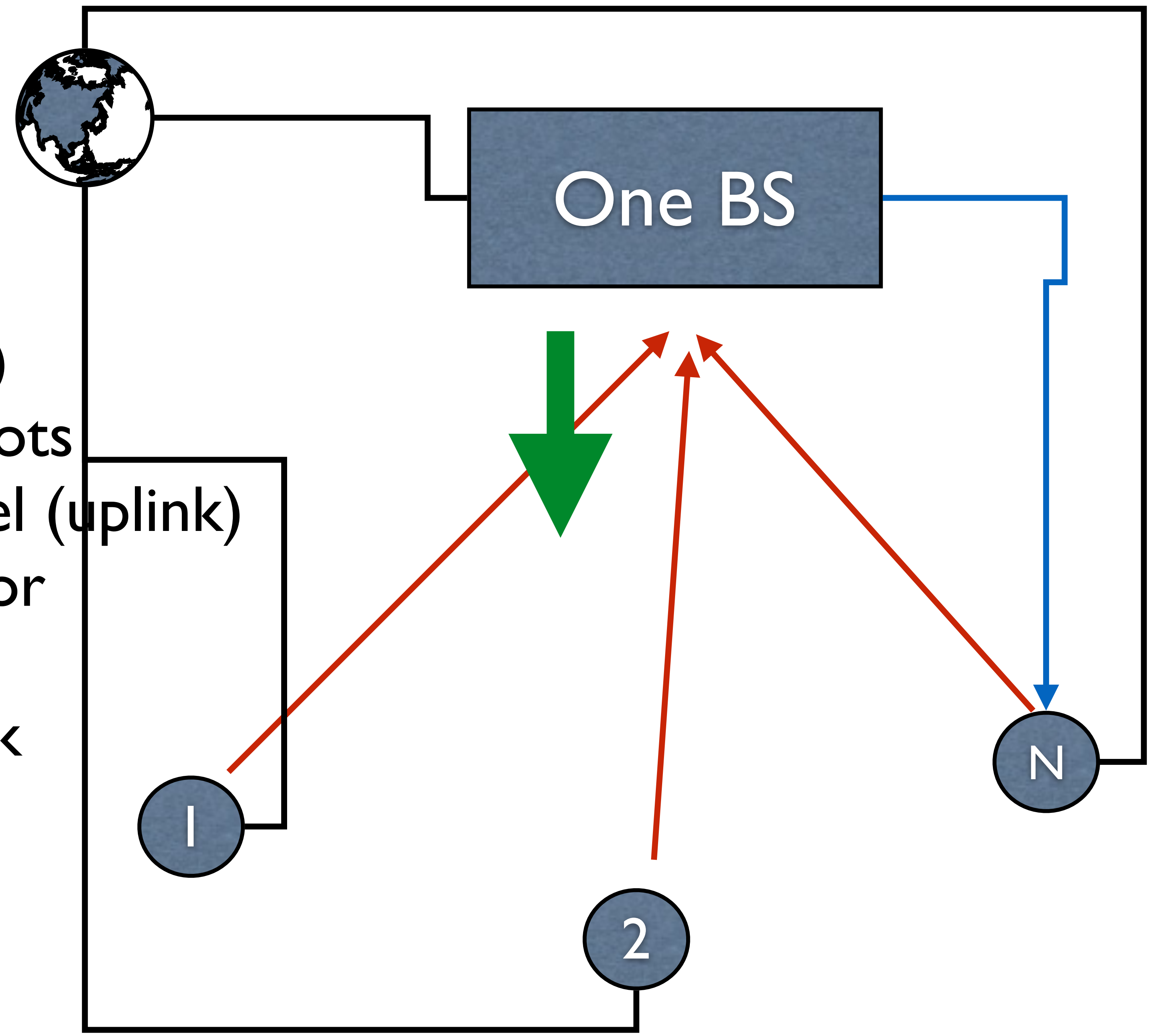
- 1 BS and N nodes (software radio based)
- A multi-bands multi-slots multiple access channel (uplink)
- A broadcast channel for Synch and beaconing
- An error free feedback channel for learning



Desired Architecture

Features

- 1 BS and N nodes (software radio based)
- A multi-bands multi-slots multiple access channel (uplink)
- A broadcast channel for Synch and beaconing
- An error free feedback channel for learning
- A remote access for programming

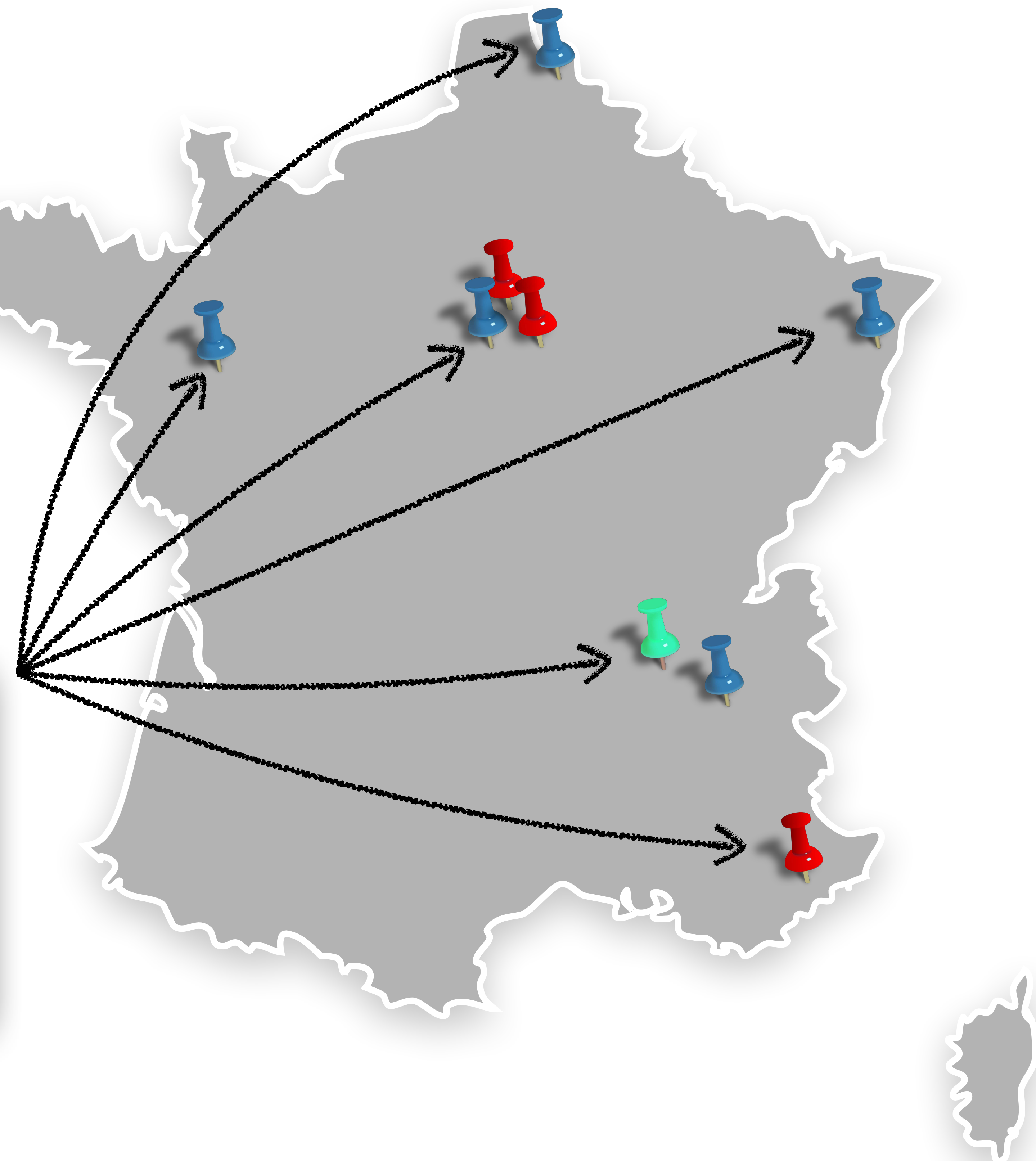


Few words about FIT/CorteXlab

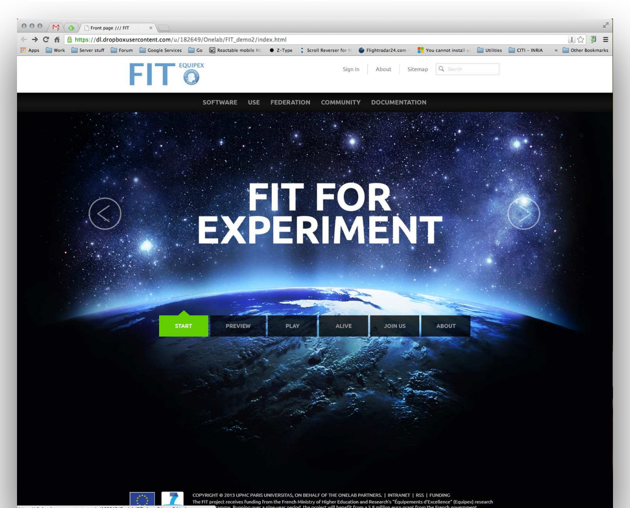
What is FIT/CorteXlab?

- **F**uture **I**nternet of **T**hings (Equipex)
- **Co**gnitive **r**adio **t**estbed and **eX**perimentation **lab**
- A facility for state-of-the-art PHY layer (but not restricted to) research and test
- Remotely accessible to the global community of wireless experts
- Open source community and code sharing (FPGA and GNU Radio)

FIT
FUTURE INTERNET
TESTING FACILITY



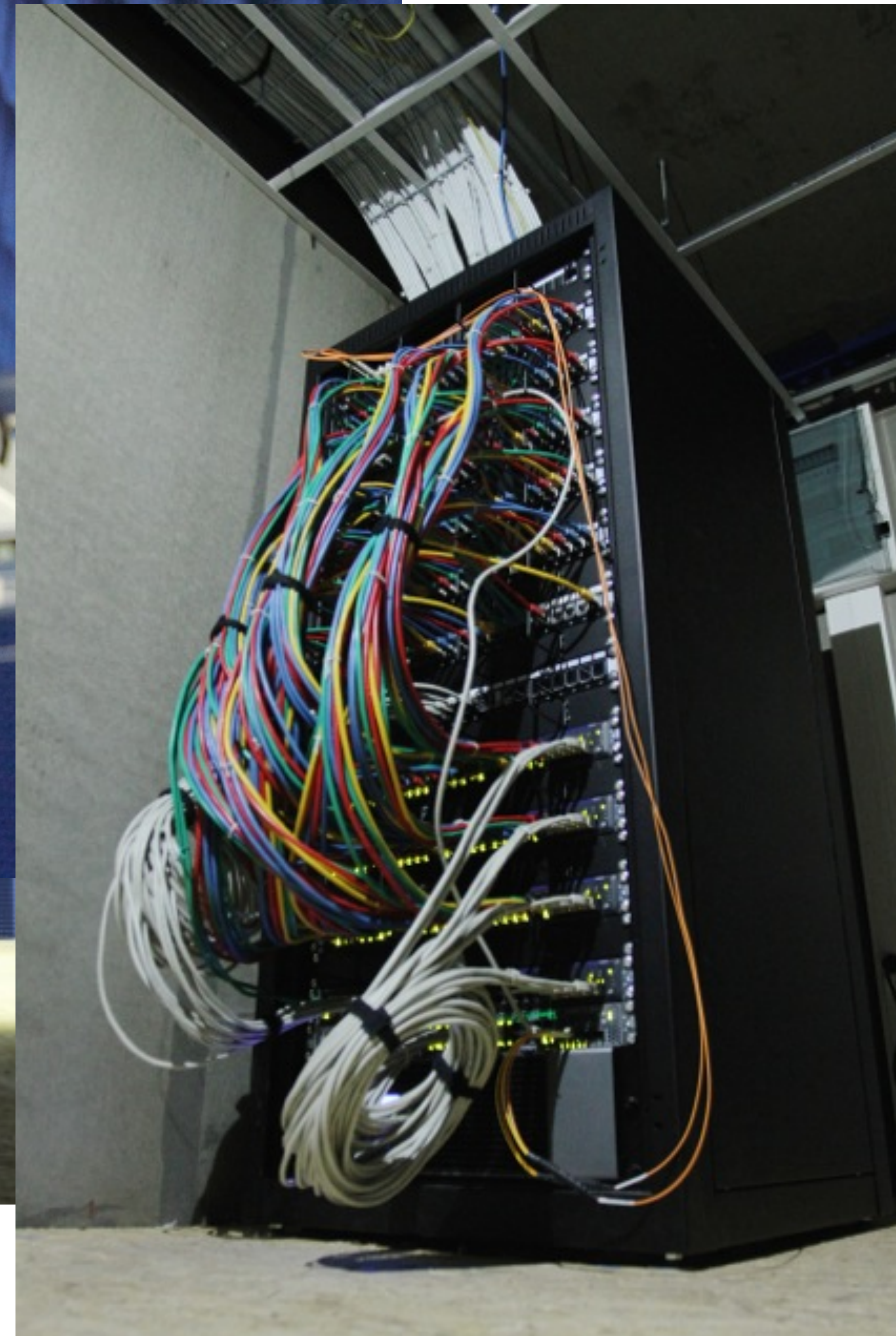
OneLab
FUTURE INTERNET TESTBEDS



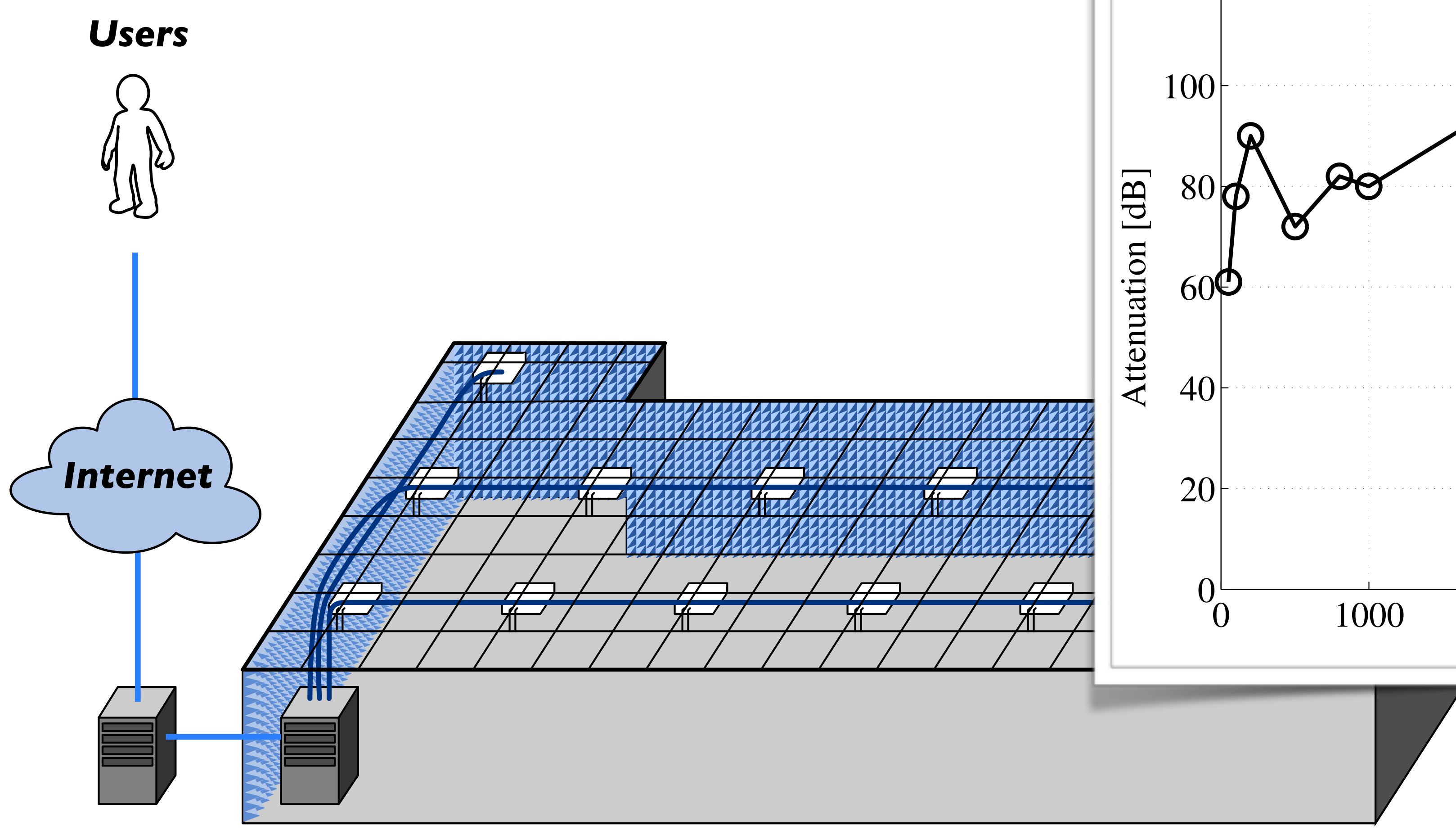
Radio Platforms and Equipment

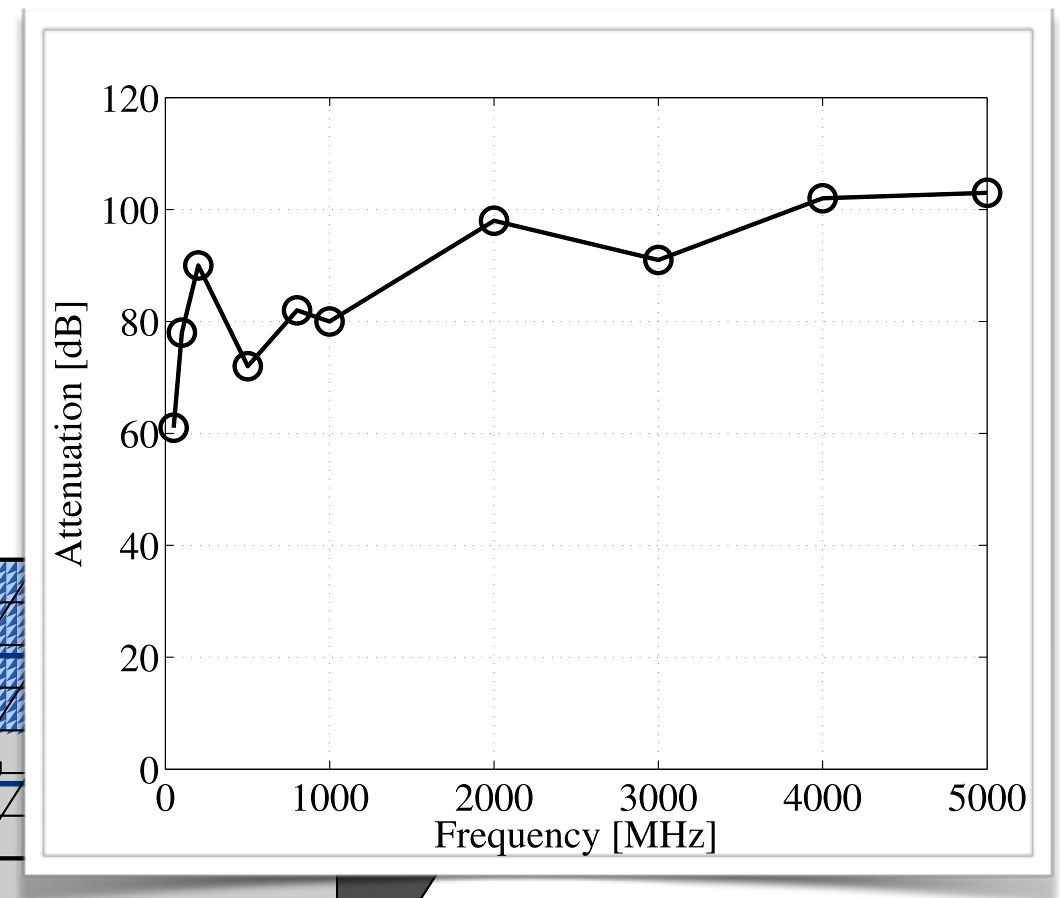
- A total of 38 (+ 4) nodes in three kinds:
  - 22 - SDR SISO - NI USRPs 2932 (400 MHz - 4 GHz @ 20 MHz)
  - 12 - SDR MIMO 2x2 - PicoSDR (380 MHz - 3.3 GHz @ 28 MHz)
  - 4 - SDR MIMO 4x4 - PicoSDR (380 MHz - 3.3 GHz @ 28 MHz)
  - To be deployed soon: 4 NI USRPs 2944R (6 MHz - 6 GHz @ 160 MHz)

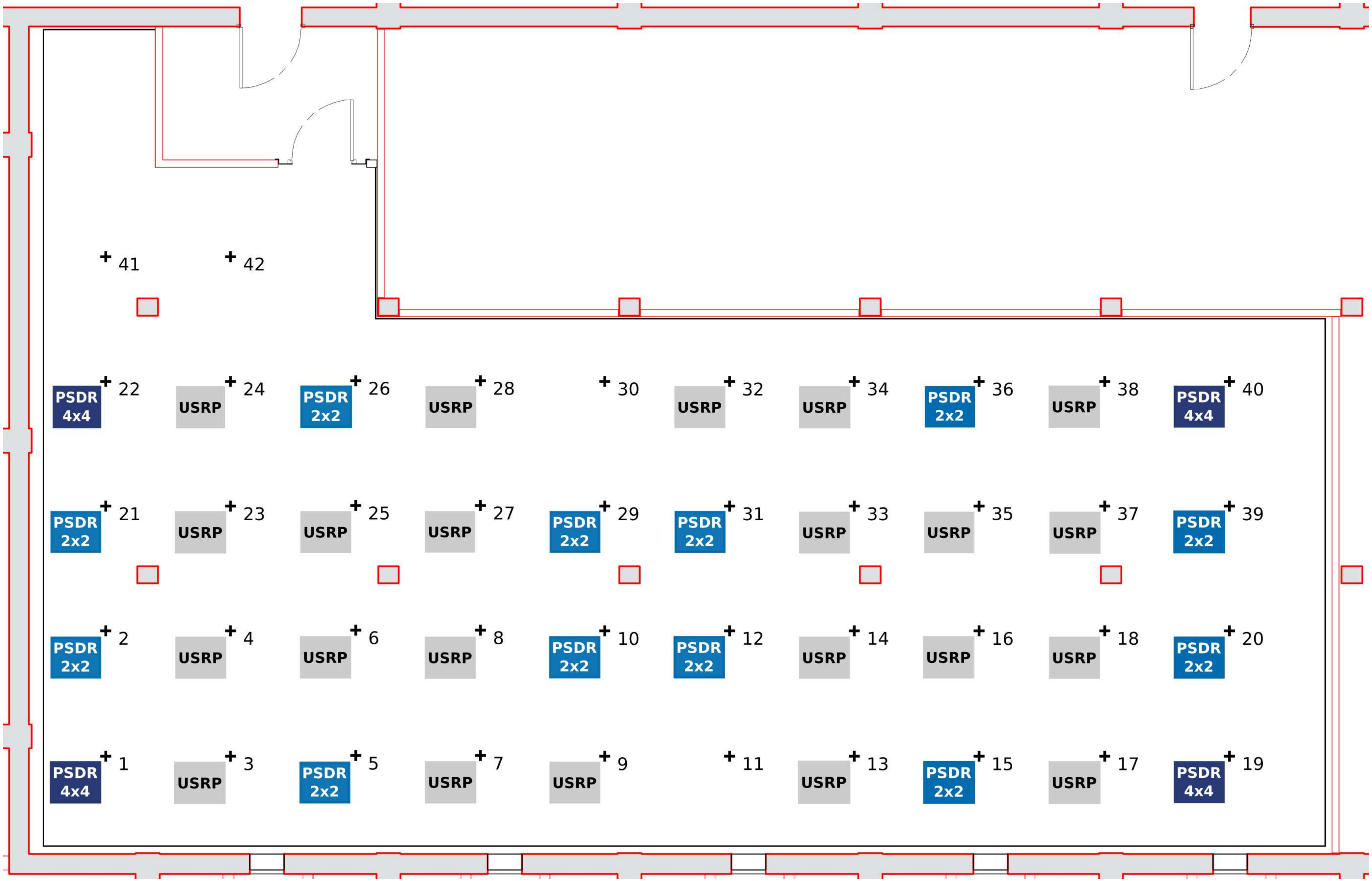

OmniLOG® 70600



Experimentation Room







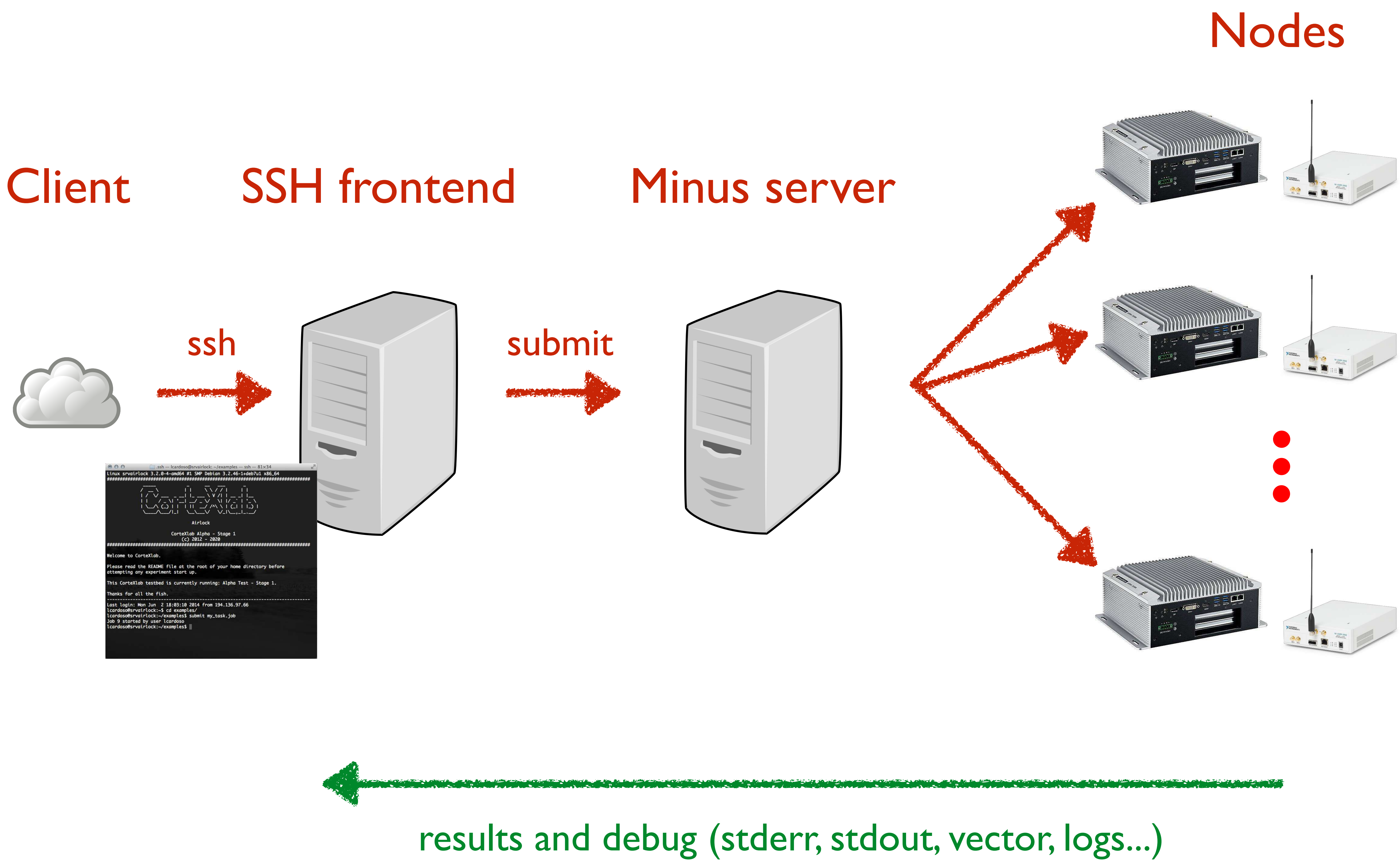
41
42
PSDR 4x4
22
USRP
24
PSDR 2x2
26
USRP
28
30
USRP
32
USRP
34
PSDR 2x2
36
USRP
38
PSDR 4x4
40
PSDR 2x2
21
USRP
23
USRP
25
USRP
27
PSDR 2x2
29
PSDR 2x2
31
USRP
33
USRP
35
USRP
37
PSDR 2x2
39
PSDR 2x2
2
USRP
4
USRP
6
USRP
8
PSDR 2x2
10
PSDR 2x2
12
USRP
14
USRP
16
USRP
18
PSDR 2x2
20
PSDR 4x4
1
USRP
3
PSDR 2x2
5
USRP
7
USRP
9
11
USRP
13
PSDR 2x2
15
USRP
17
PSDR 4x4
19

Experimentation Software

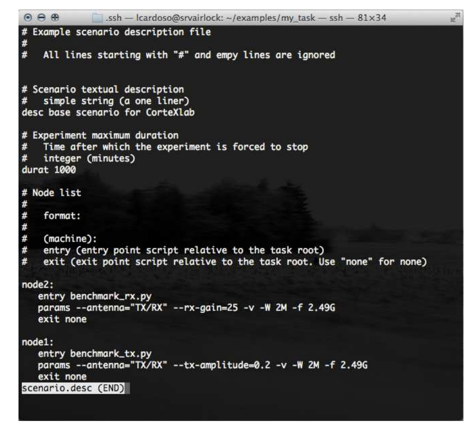
Testbed Usage Overview

- **oar** suite of tools allow to manage reservations on FIT/CorteXlab
- **minus** controls all aspects of the experiment execution
- Results are automatically loaded into the user home folder in the ssh frontend
- FFT-Web and log files allows for real-time monitoring of the experiment

Running an Experiment



Experiment Task



```
.ssh — lcardoso@srvairlock: ~/examples/my_task — ssh — 81×34
# Example scenario description file
#
#   All lines starting with "#" and empy lines are ignored


# Scenario textual description
#   simple string (a one liner)
desc base scenario for CorteXlab

# Experiment maximum duration
#   Time after which the experiment is forced to stop
#   integer (minutes)
durat 1000

# Node list
#
#   format:
#
#   (machine):
#   entry (entry point script relative to the task root)
#   exit (exit point script relative to the task root. Use "none" for none)

node2:
   entry benchmark_rx.py
   params --antenna="TX/RX" --rx-gain=25 -v -W 2M -f 2.49G
   exit none

node1:
   entry benchmark_tx.py
   params --antenna="TX/RX" --tx-amplitude=0.2 -v -W 2M -f 2.49G
   exit none
scenario.desc (END)
```

scenario.yaml

GNU Radio (Python)

GNU Radio (OOT libs)

C/C++

⋮

LabView executable?

⊕ → Minus task

Pedagogic Utilities

- Wiki page contains usage instructions
- Working tutorials on our wiki:
  - GNU Radio benchmark, our "Hello world"
  - From GNU Radio to CorteXlab (using USRPs)
  - From GNU Radio to CorteXlab (using PicoSDRs)
  - Custom bitstream with PicoSDR
  - Spectrum Analyzer example (FFT-Web)

http://wiki.cortexlab.fr

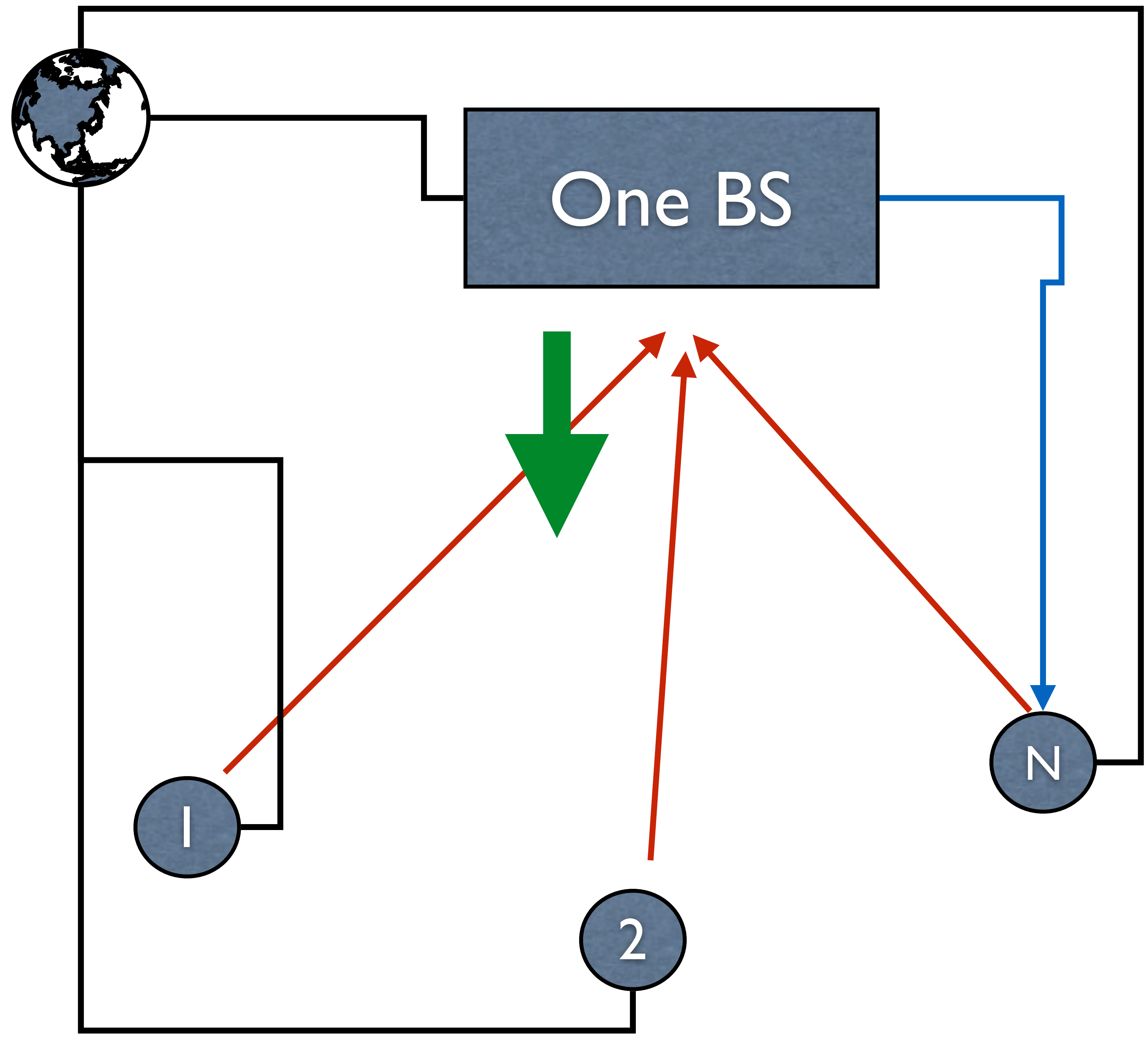
GitHub

- We also have a GitHub account with many utilities:
  - examples: some examples for the tutorials
  - cxlb-multitask-generator: automated generation of tasks
  - cxlb-link-profiler: link profiling code
  - gr-cortexlab: OOT blocks for 802.15.4 code
  - fft-web: fat-web utility and GNU Radio block
  - demos: fully running demos, including Interference Alignment
  - psdr-interface: a PicoSDR wrapper for GNU Radio
  - tool: some userspace tools to help the testbed usage

http://github.com/CorteXlab

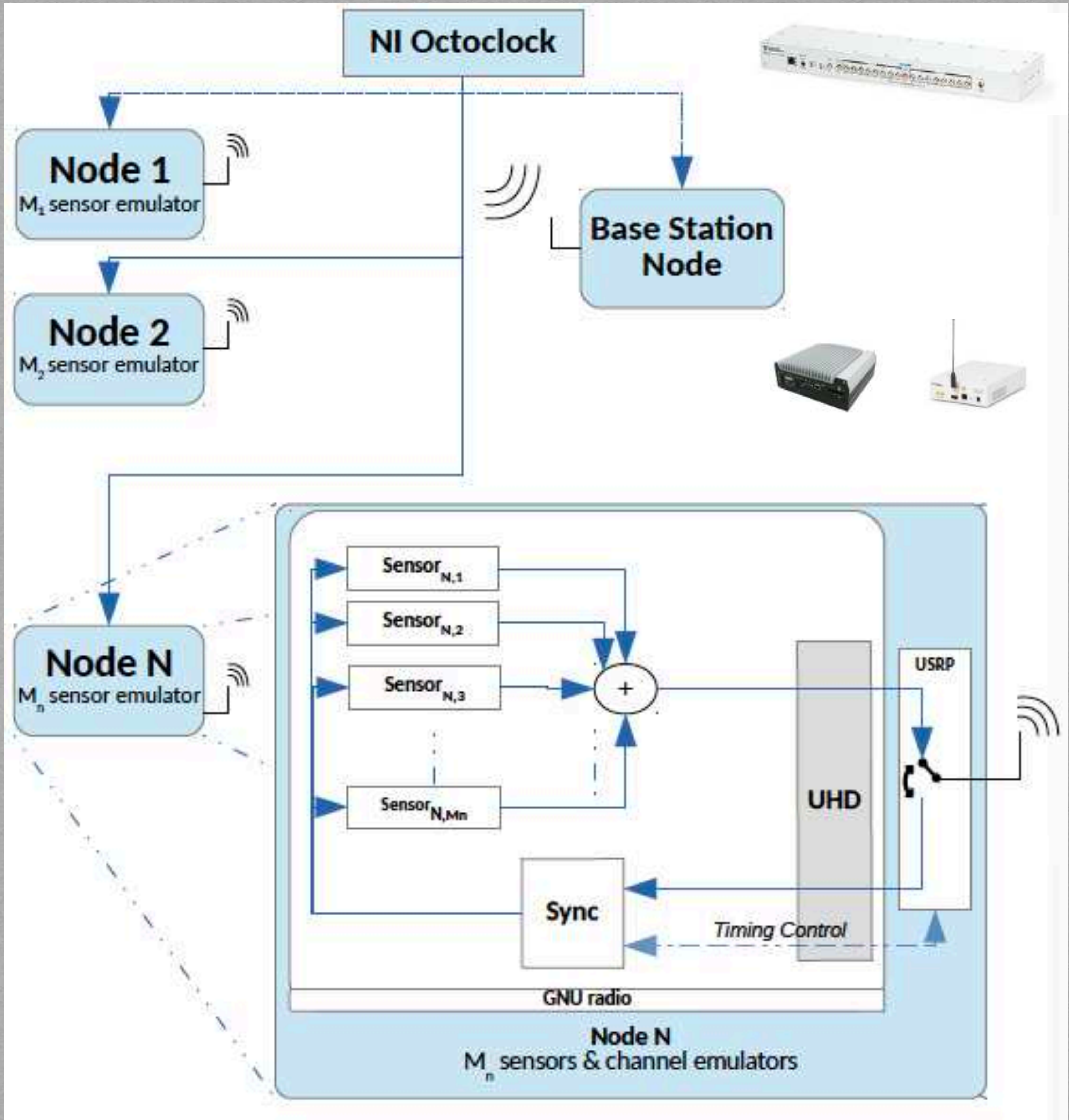
Ephyl framework on CorteXlab

Desired Architecture

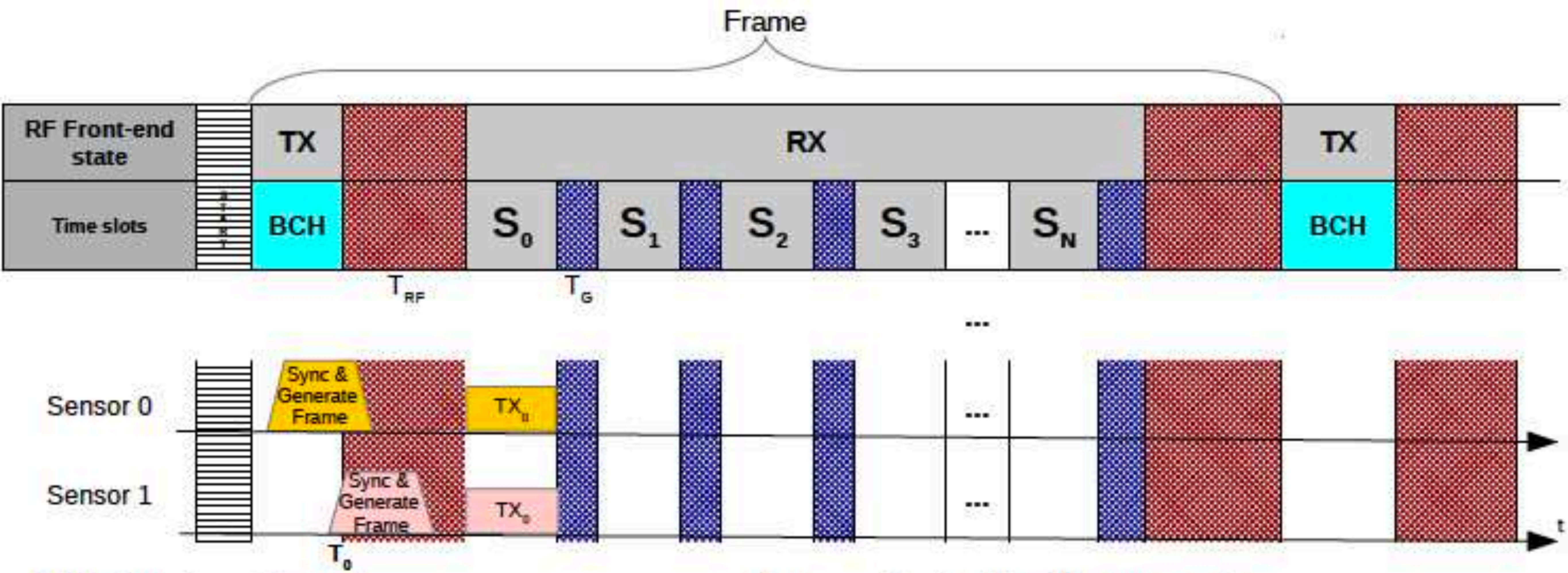


Proposed architecture on FIT/CorteXlab

- Deploy an IoT network with as many nodes as possible nodes :
  - M Sensors and Channels emulated on N (N<M) radio Nodes
  - One Base Station node (BS)
  - All nodes synchronized via NI Octoclock
  - A flexible PHY layer (inspired from NB-IoT Standard)



Reference Scheduler



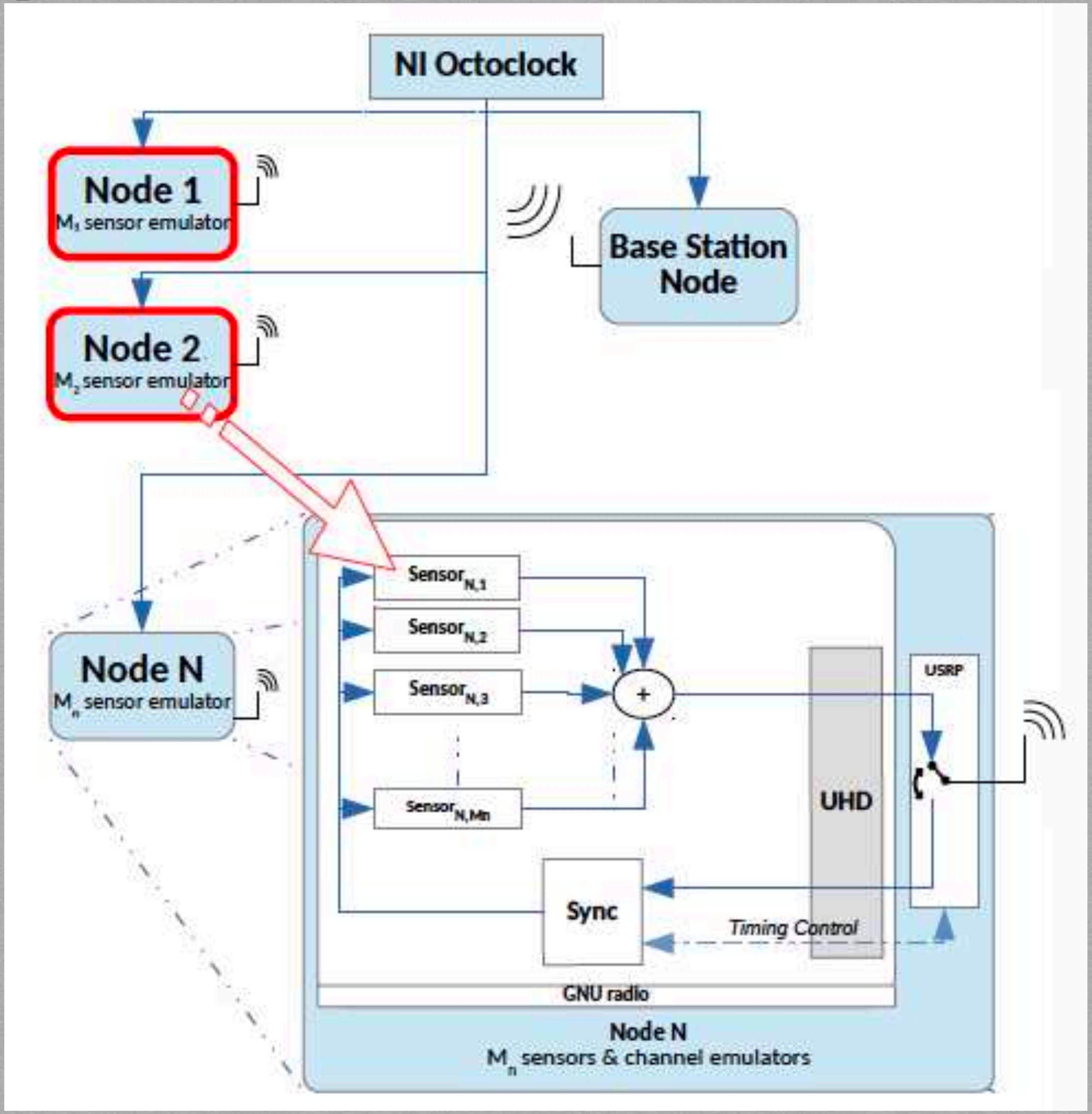
BCH : Broadcast Channel
S0,S1... : Shared Channel
T_{RF} : RF mode switch duration
T_G : Guard Time, to avoid slot **overlapping** and, if needed, **store/process** decoded data

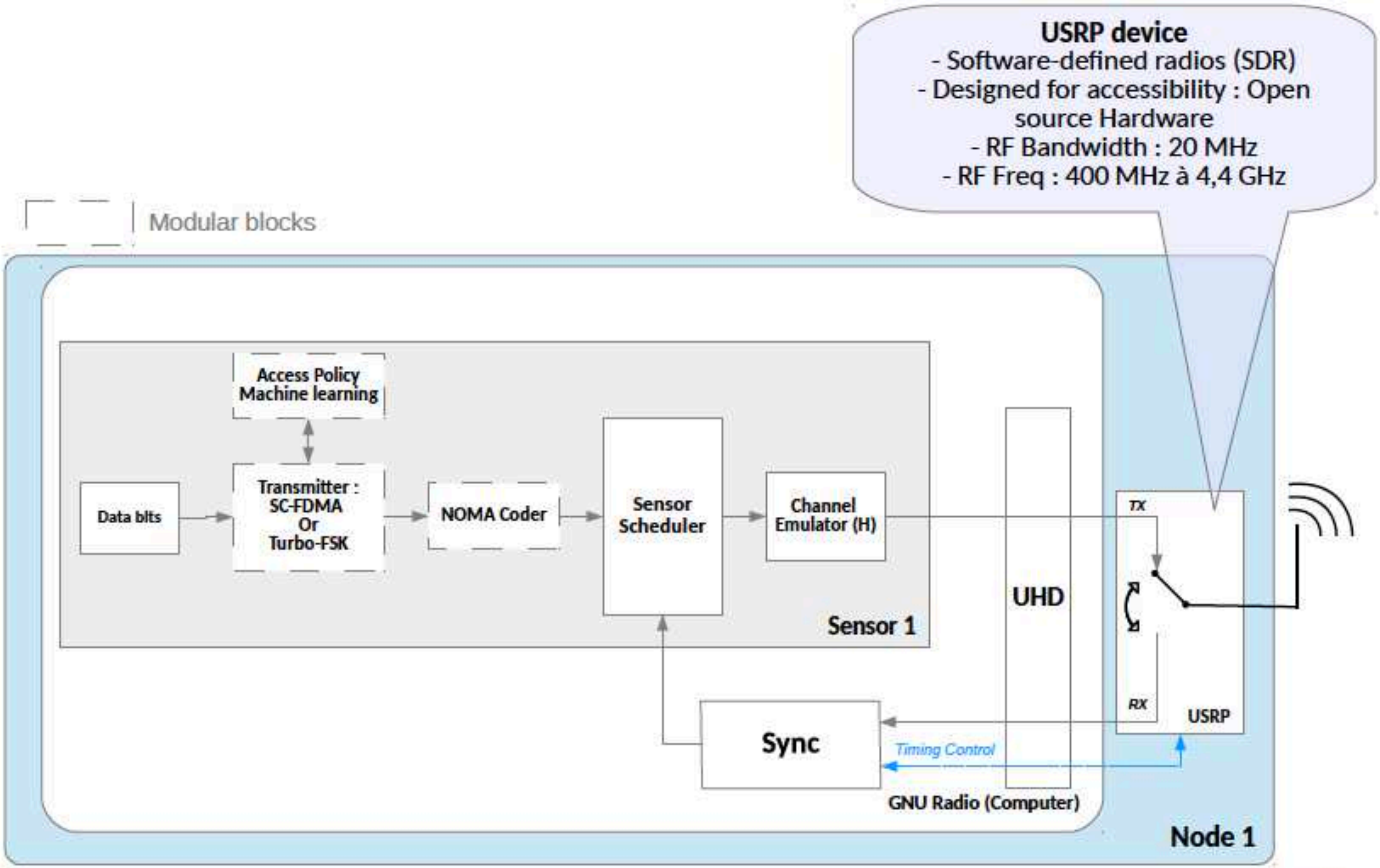
→ **Increase T_{RF} to allow the sensors to perfom Synchronization & Packetization**

Transmission in S_0 is possible for both cases

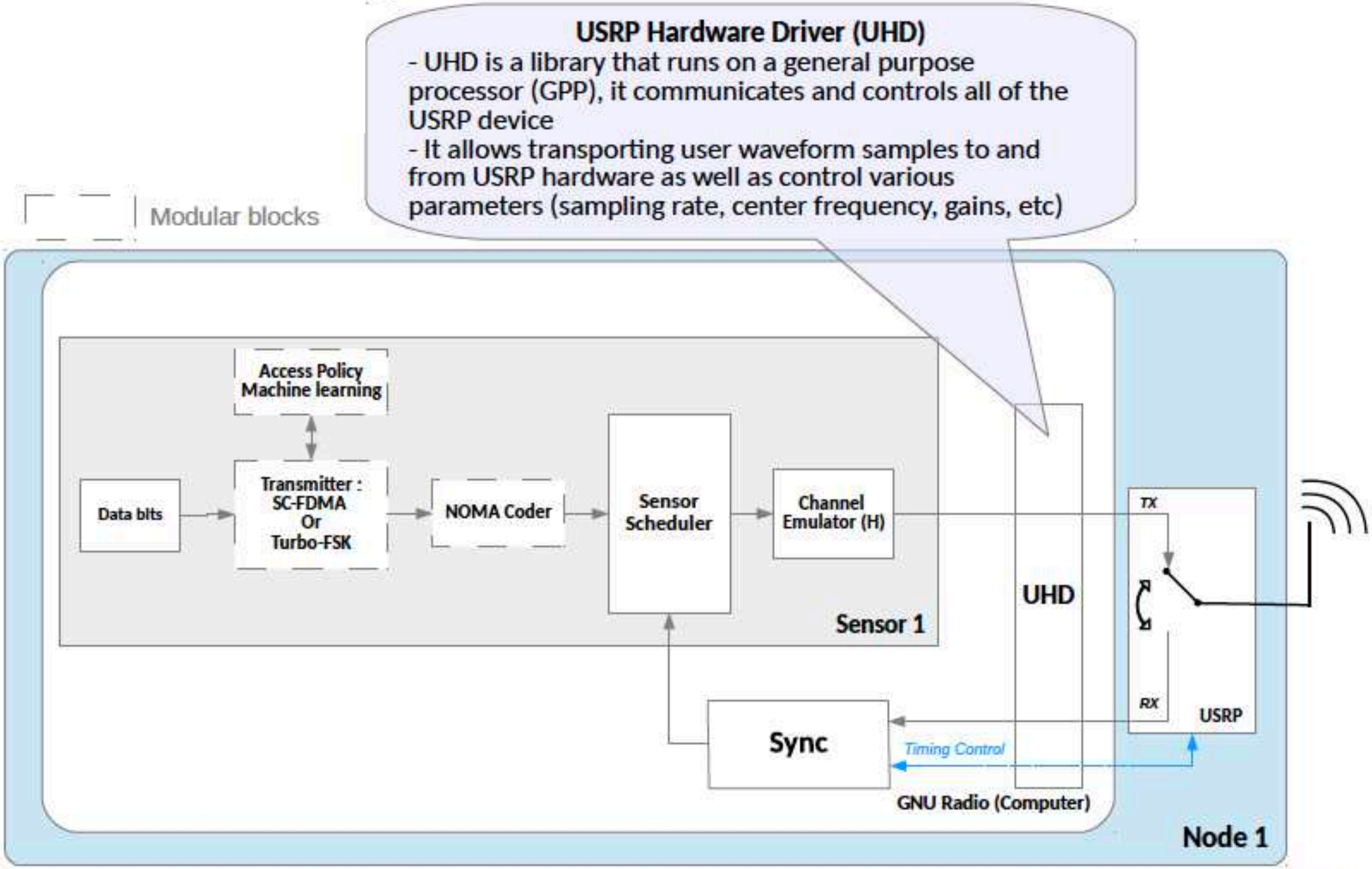
Inside a FIT/CorteXlab node

- To reduce computational time, Synch. Tasks are done once in a node for all emulated sensors
- The I/Q signals are built independently and summed after a digital channel emulator to add channel diversity

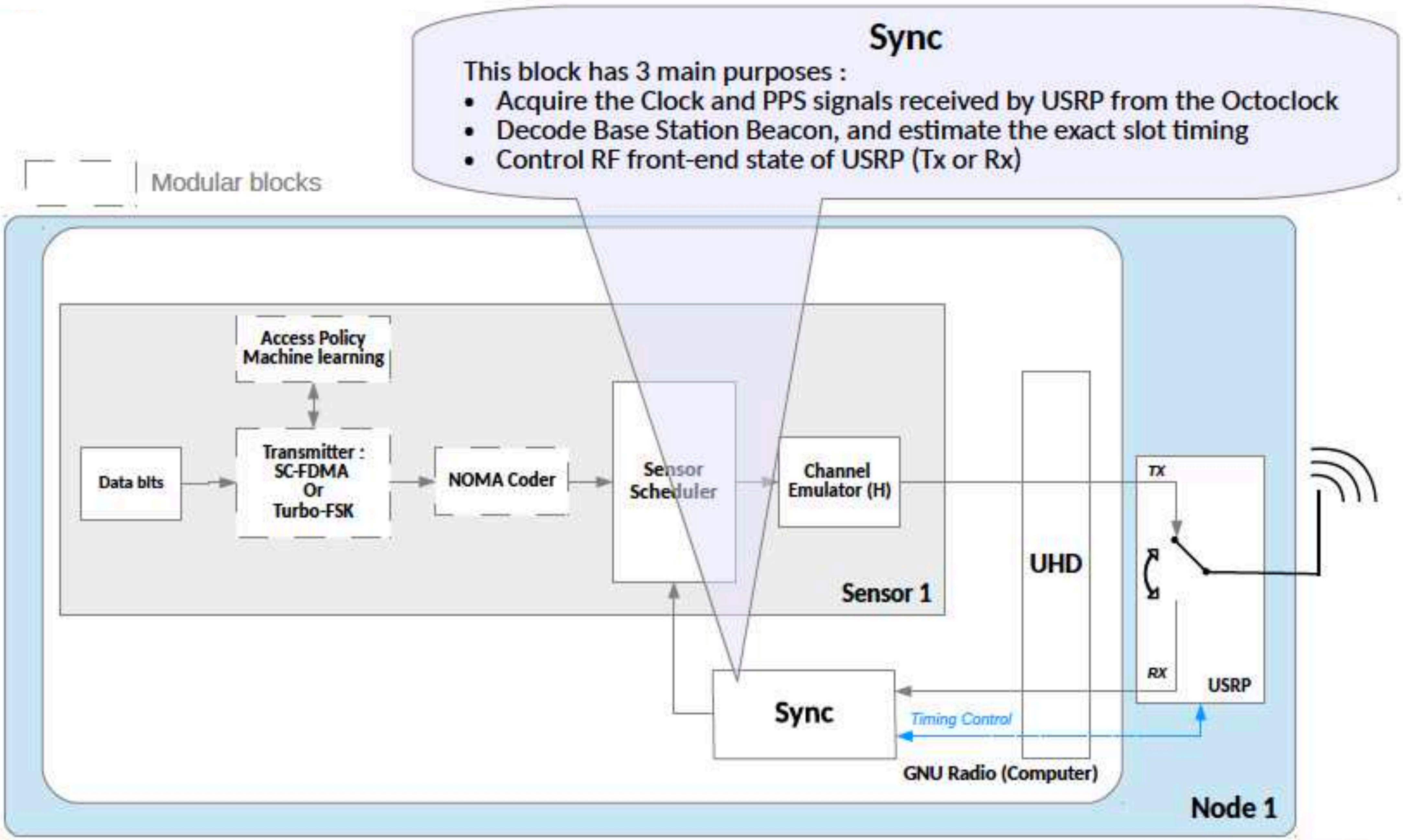




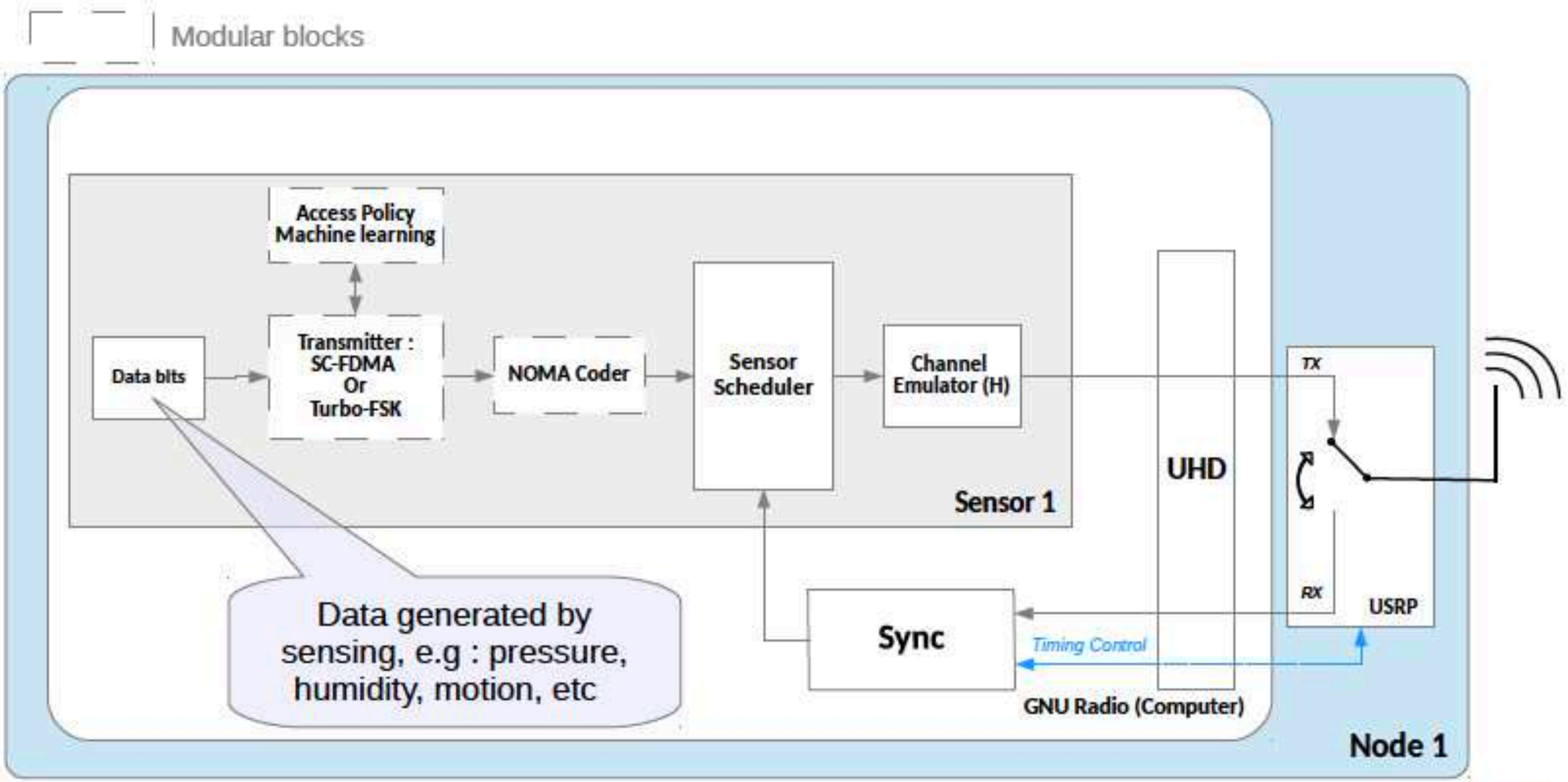
USRP device
- Software-defined radios (SDR)
- Designed for accessibility : Open source Hardware
- RF Bandwidth : 20 MHz
- RF Freq : 400 MHz à 4,4 GHz
Modular blocks
Access Policy Machine learning
Data bits
Transmitter : SC-FDMA Or Turbo-FSK
NOMA Coder
Sensor Scheduler
Channel Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1



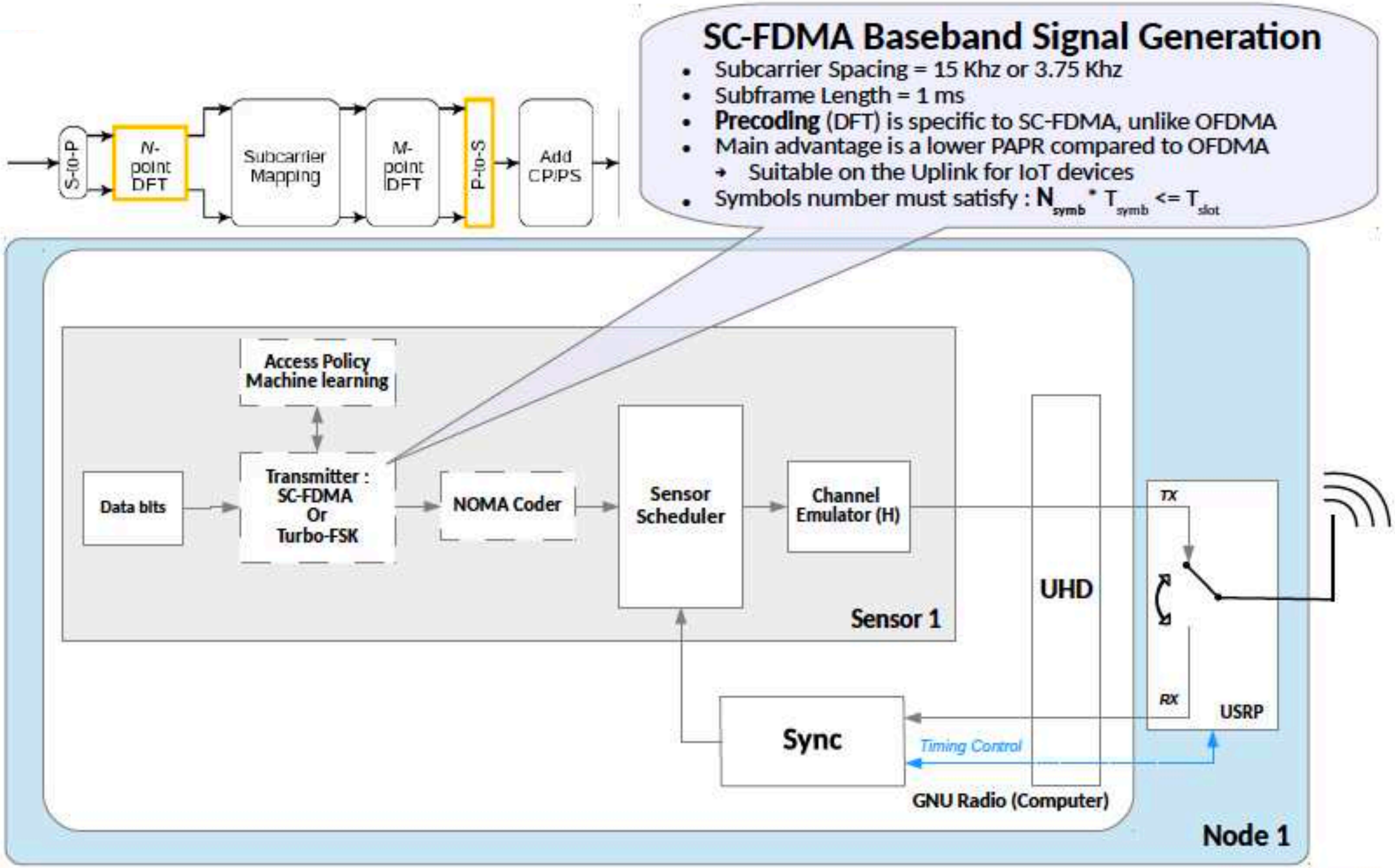
USRP Hardware Driver (UHD)
- UHD is a library that runs on a general purpose processor (GPP), it communicates and controls all of the USRP device
- It allows transporting user waveform samples to and from USRP hardware as well as control various parameters (sampling rate, center frequency, gains, etc)
Modular blocks
Access Policy
Machine learning
Data bits
Transmitter :
SC-FDMA
Or
Turbo-FSK
NOMA Coder
Sensor
Scheduler
Channel
Emulator (H)
UHD
TX
RX
USRP
Sensor 1
Sync
Timing Control
GNU Radio (Computer)
Node 1



Sync
This block has 3 main purposes :
• Acquire the Clock and PPS signals received by USRP from the Octoclock
• Decode Base Station Beacon, and estimate the exact slot timing
• Control RF front-end state of USRP (Tx or Rx)
Modular blocks
Access Policy
Machine learning
Data bits
Transmitter :
SC-FDMA
Or
Turbo-FSK
NOMA Coder
Sensor
Scheduler
Channel
Emulator (H)
UHD
TX
RX
USRP
Sensor 1
Sync
Timing Control
GNU Radio (Computer)
Node 1



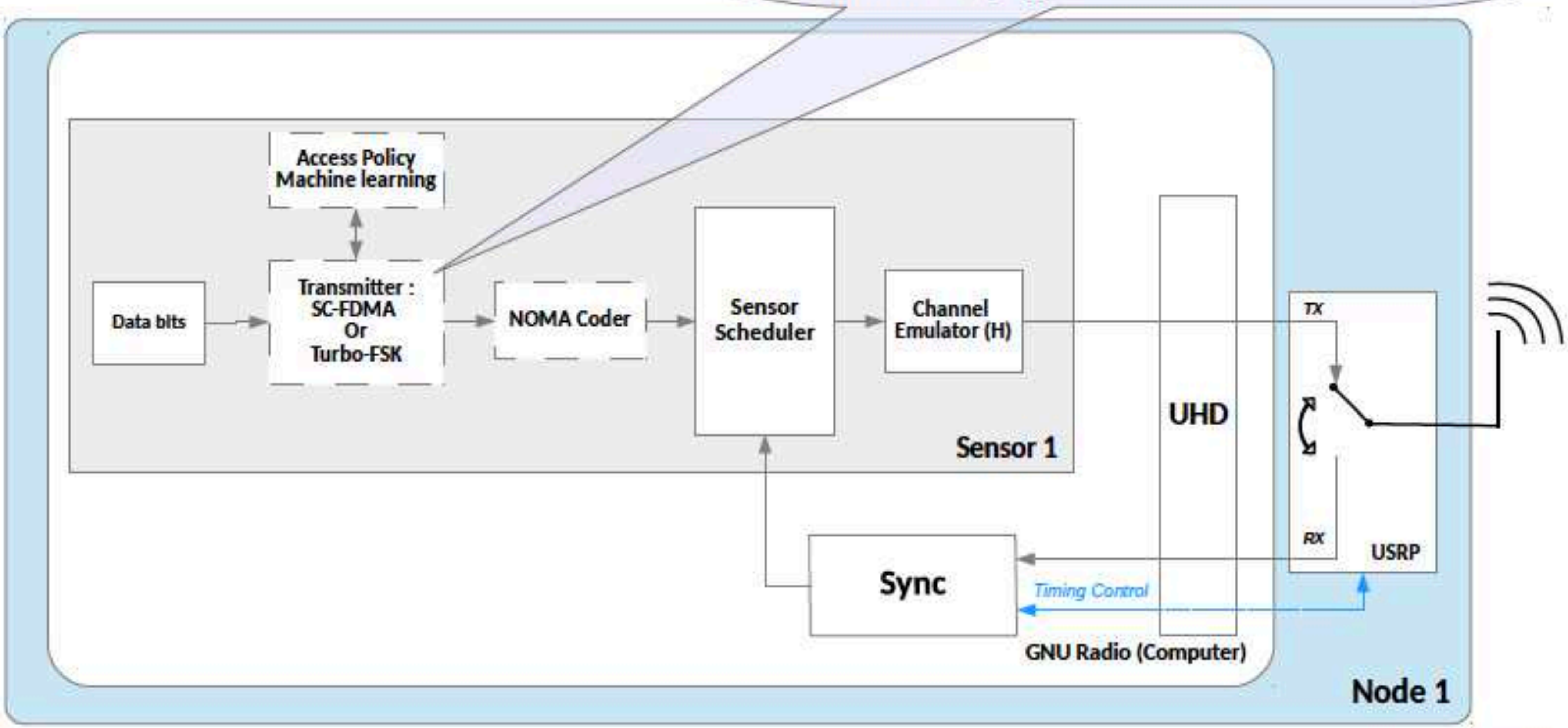
Modular blocks
Access Policy
Machine learning
Data bits
Transmitter :
SC-FDMA
Or
Turbo-FSK
NOMA Coder
Sensor
Scheduler
Channel
Emulator (H)
UHD
TX
RX
USRP
Sensor 1
Data generated by
sensing, e.g : pressure,
humidity, motion, etc
Sync
Timing Control
GNU Radio (Computer)
Node 1

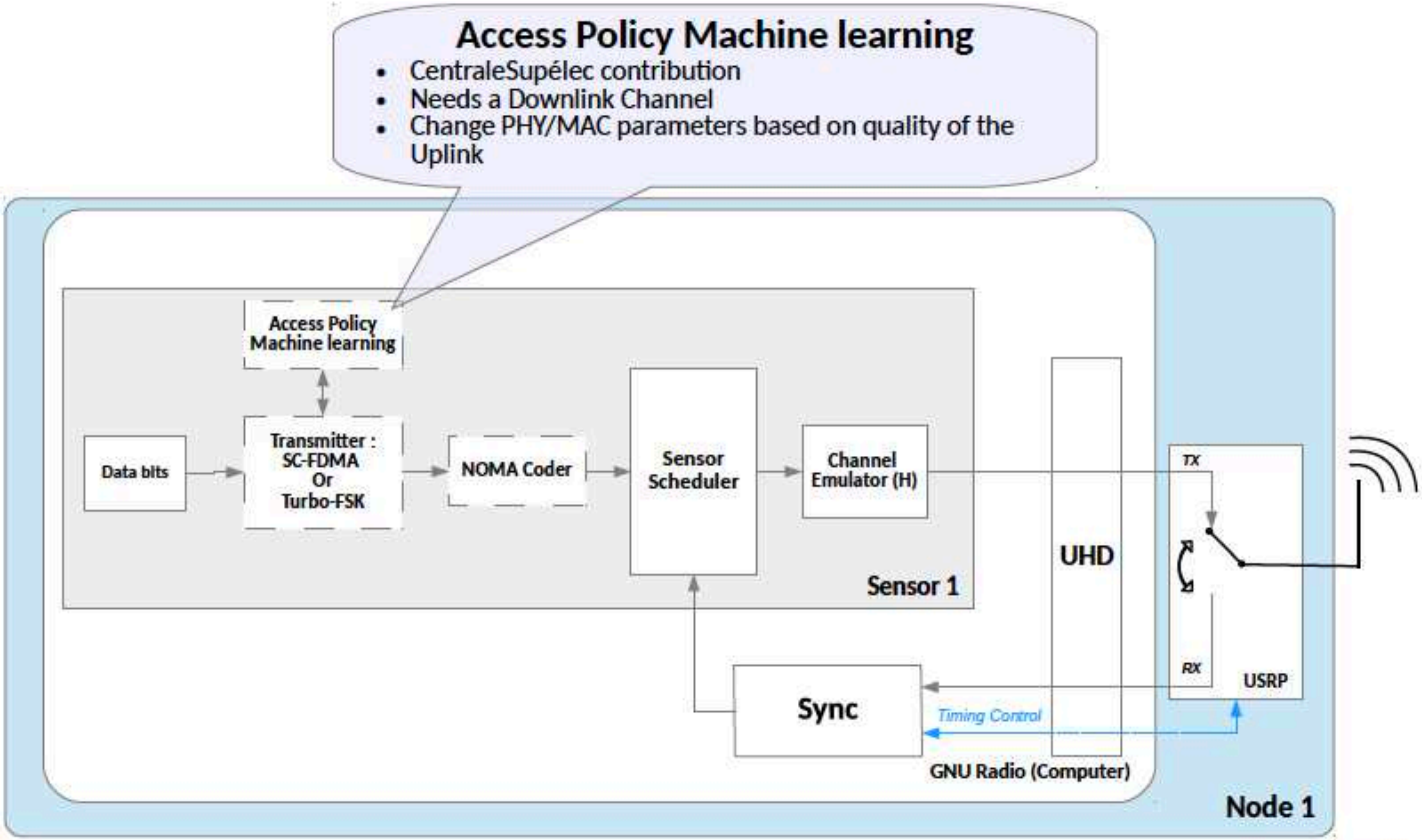


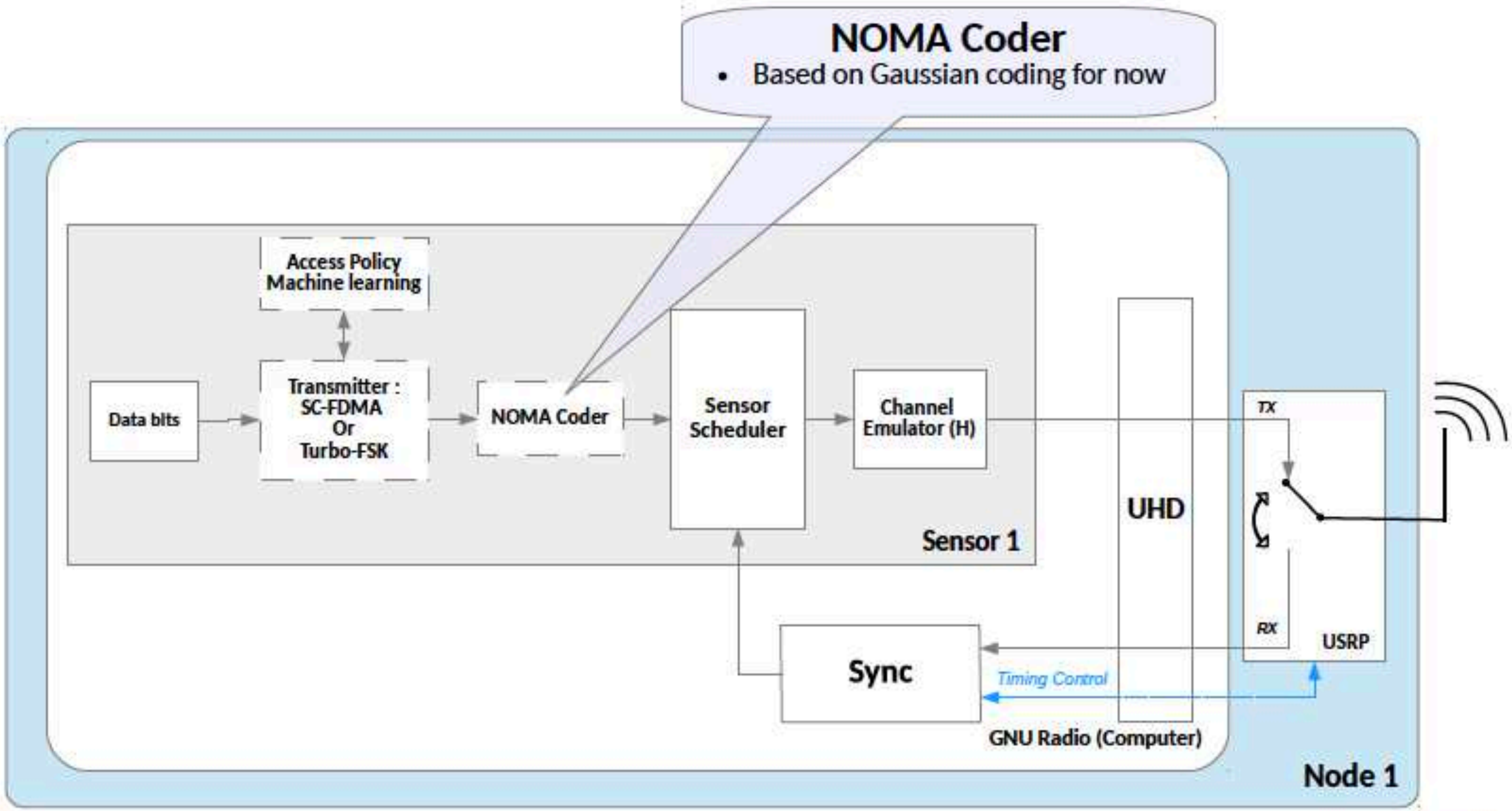
SC-FDMA Baseband Signal Generation
Subcarrier Spacing = 15 Khz or 3.75 Khz
Subframe Length = 1 ms
Precoding (DFT) is specific to SC-FDMA, unlike OFDMA
Main advantage is a lower PAPR compared to OFDMA
Suitable on the Uplink for IoT devices
Symbols number must satisfy : N symb * T symb <= T slot
S-to-P
N-point DFT
Subcarrier Mapping
M-point IDFT
P-to-S
Add CP/PS
Access Policy Machine learning
Data bits
Transmitter : SC-FDMA Or Turbo-FSK
NOMA Coder
Sensor Scheduler
Channel Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1

Turbo-FSK Baseband Signal Generation

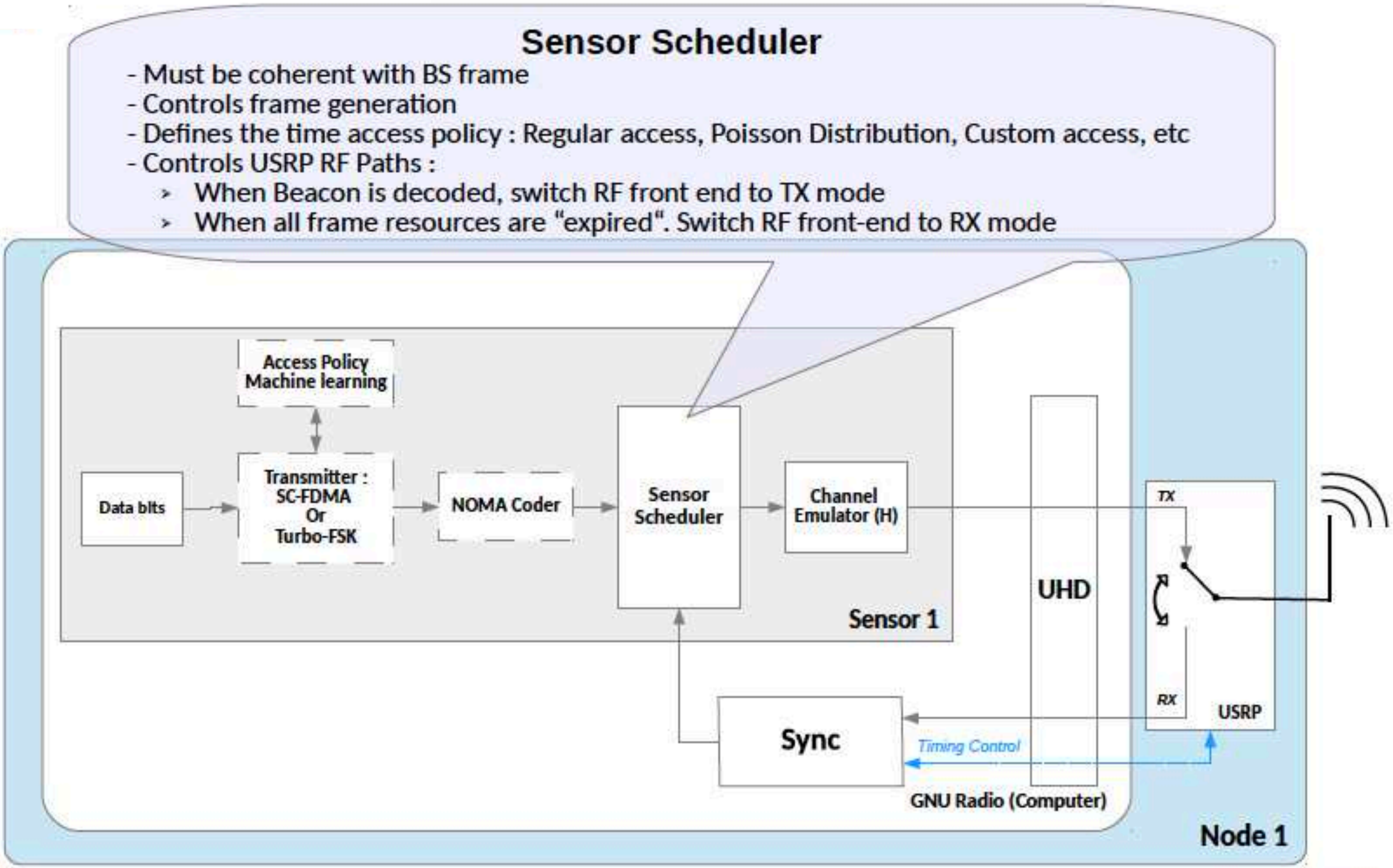
- CEA Leti contribution
- Compatibility with CorteXlab. Possible solutions :
  - Convert all DSP to C++/Python library
  - Or generate I/Q data "offline" > Decoding also offline
- Discussion is in progress

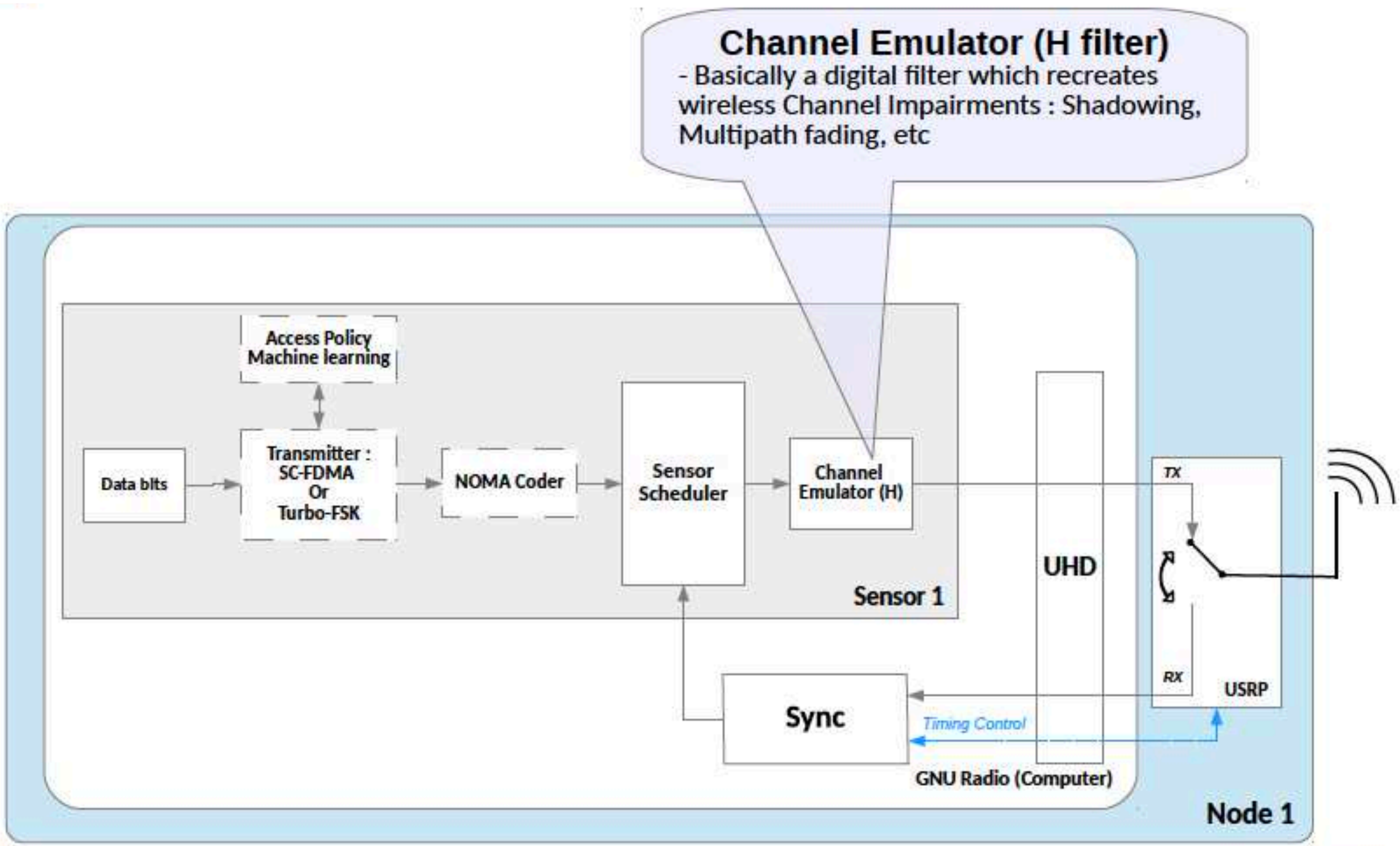



Access Policy Machine learning
CentraleSupélec contribution
Needs a Downlink Channel
Change PHY/MAC parameters based on quality of the Uplink
Access Policy Machine learning
Data bits
Transmitter : SC-FDMA Or Turbo-FSK
NOMA Coder
Sensor Scheduler
Channel Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1



NOMA Coder
• Based on Gaussian coding for now
Access Policy
Machine learning
Data bits
Transmitter :
SC-FDMA
Or
Turbo-FSK
NOMA Coder
Sensor
Scheduler
Channel
Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1


Sensor Scheduler
- Must be coherent with BS frame
- Controls frame generation
- Defines the time access policy : Regular access, Poisson Distribution, Custom access, etc
- Controls USRP RF Paths :
› When Beacon is decoded, switch RF front end to TX mode
› When all frame resources are "expired". Switch RF front-end to RX mode
Access Policy Machine learning
Data bits
Transmitter : SC-FDMA Or Turbo-FSK
NOMA Coder
Sensor Scheduler
Channel Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1



Channel Emulator (H filter)
- Basically a digital filter which recreates wireless Channel Impairments : Shadowing, Multipath fading, etc
Access Policy Machine learning
Data bits
Transmitter : SC-FDMA Or Turbo-FSK
NOMA Coder
Sensor Scheduler
Channel Emulator (H)
Sensor 1
UHD
TX
RX
USRP
Sync
Timing Control
GNU Radio (Computer)
Node 1

Some tricks in the implementation

- Nodes emulation to increase the number of nodes.
- Tunable waiting periods to absorb computational overload.
- Mixture of data flow programming and state machines (thanks to Tags and message passing).

Take Away messages

About using FIT/CorteXlab

- **Can you use FIT/CorteXlab ?**
  - YES you can … take a tour on our website or visit us.
  - You can even use the Ephyl Framework for your own test.
  - We suggest to put your code or data on the repository to increase your impact
  - Don’t forget to call it **“FIT/CorteXlab”** in your publications.

About ideas for FIT/CorteXlab

- Some ideas for future research
  - To try new PHY layer (LORA under implementation) or MAC layer.
  - To develop FPGA blocks to fasten the running phase (using RFNOC ?)
  - Any willingness to try deep learning ? A GPU server has just been added in FIT/CortexLab.
  - To deploy FIT/CorteXlab nodes somewhere else.
  - To put your nodes inside the room with a remote access to test it in a controllable interfered environment.

- Feel free to contact us… we may help, you can help

Thank you!

more info: http://www.cortexlab.fr and @FITCorteXlab