INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Towards the future Internet: a step forward the successfully evolution of WSNs into the IoT

PhD defense

PhD candidate:
Stefano Bocchino

October 16th, 2014

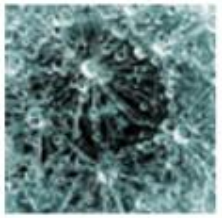
Outline

1. Background and motivation
2. Contributions
3. In-network processing and (Ease of) Reprogramming
4. Conclusions

Outline

1. **Background and motivation**
2. Contributions
3. In-network processing and (Ease of) Reprogramming
4. Conclusions

Internet Evolution

Internet of boffins	Internet of geeks	Internet of masses	Mobile Internet	Internet of things
				
1969 - 1995	1995 - 2000	2000 - 2007	2007 - 2011	2012 & beyond

*riteshjagzape.wordpress.com

+ Users
+ Devices

50 billion devices by 2020

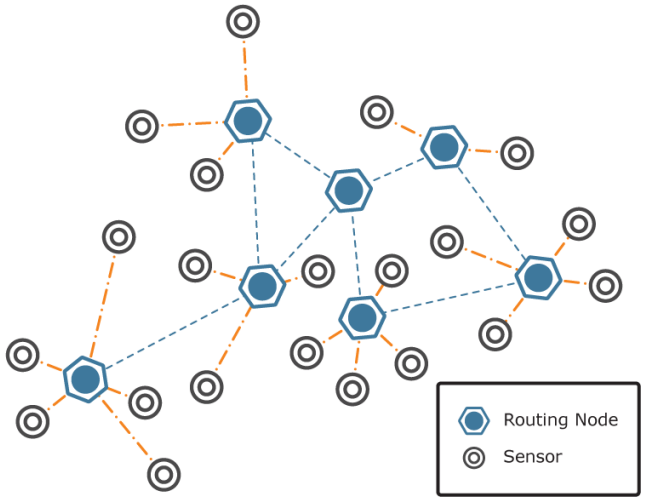
Internet of Things

- The term "Internet of Things" was first used by Kevin Ashton in 1999 (Auto-ID Center)
- Is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure

Wireless Sensor Network

Networks of *smart* sensors

Cheap and pervasive technology for monitoring and controlling the physical world

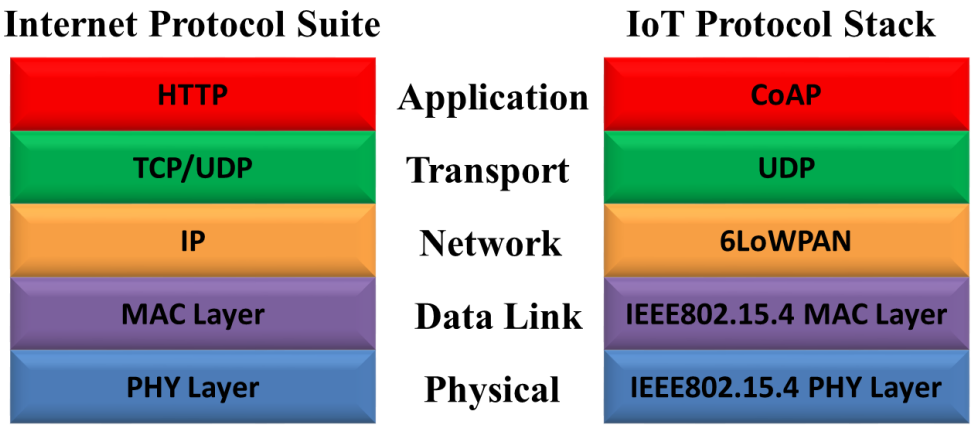


Cornerstone for enabling the Internet of Things paradigm

The Internet of Things approach

Enable the Internet protocol suite in the WSN domain

- Layered networking model
- Standard protocols



Each layer provides a different level of abstraction

WSN in the Internet domain

Next generation WSN

- Powerful
- Inter-operable
- Pervasive systems
- Connected to the Internet world

The adoption of WSNs is hindered by several limitations of current systems

Research challenges (1/3)

- High-performing protocol solutions have been developed for traditional WSNs:
  - E.g., energy saving policies, routing protocols, real-time constraints
- Partial or total inoperability of such protocols in the IoT domain

Protocol adaptation

Research challenges (2/3)

- WSN devices equipped with new powerful sensors:
  - E.g., microphone, camera
- Advanced applications:
  - E.g., pervasive video surveillance, objects tracking

Combine IoT protocols with new technologies

Research challenges (3/3)

- WSN devices equipped with more powerful resources and computational capability:
  - E.g., 32-bit microcontroller, 128 Kbyte of RAM, 512 Kbyte of Flash
- Use of embedded intelligence in the "things themselves"
- Change remotely the services in pervasive and uniquely identifiable devices

In-network processing and (Ease of) Reprogramming

Outline

1. Background and motivation
2. **Contributions**
3. In-network processing and (Ease of) Reprogramming
4. Conclusions

Contributions

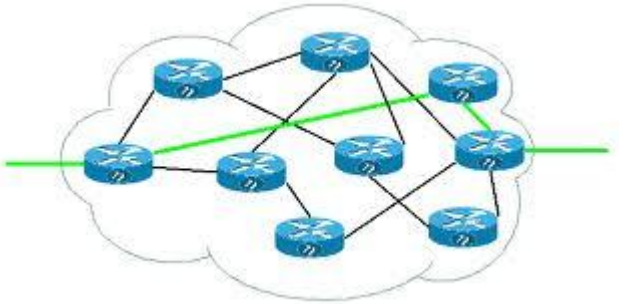
- *Protocol adaptation:*
  - Adaptation of SPEED geographic routing protocol
- *Combine IoT protocol with new technologies:*
  - Low-complexity algorithm for video streaming applications
  - Integration of WSN and RFID in the IoT scenario
- *In-network processing and (Ease of) Reprogramming:*
  - Python-based Framework for Ubiquitous Networked Sensors

Smart city scenario (Intelligent Transport System) use case

Protocol adaptation (1/2)

Adaptation of the SPEED geographic routing protocol targeted to 6LoWPAN networks

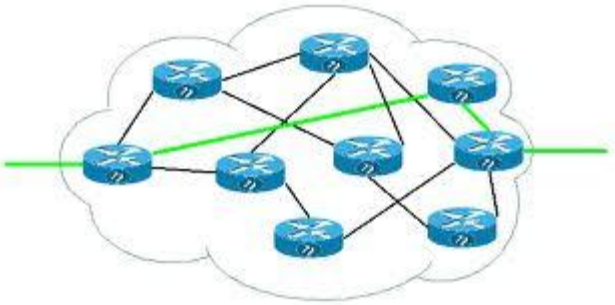
1. **SPEED-6LoWPAN:**
   - Protocol messages adaptation
   - Flow shaping capability added
   - Code source implementation in ERIKA OS
   - Experimental validation in a real test bed
   - Comparison against AODV routing protocol



Protocol adaptation (2/2)

2. SPEED-3D:

- Introduction of the third dimension (altitude)
- Enhancement of the modular task design
- Experimental validation in a real test bed
- Metrics:
  - Round Trip Time
  - Packet Loss Rate

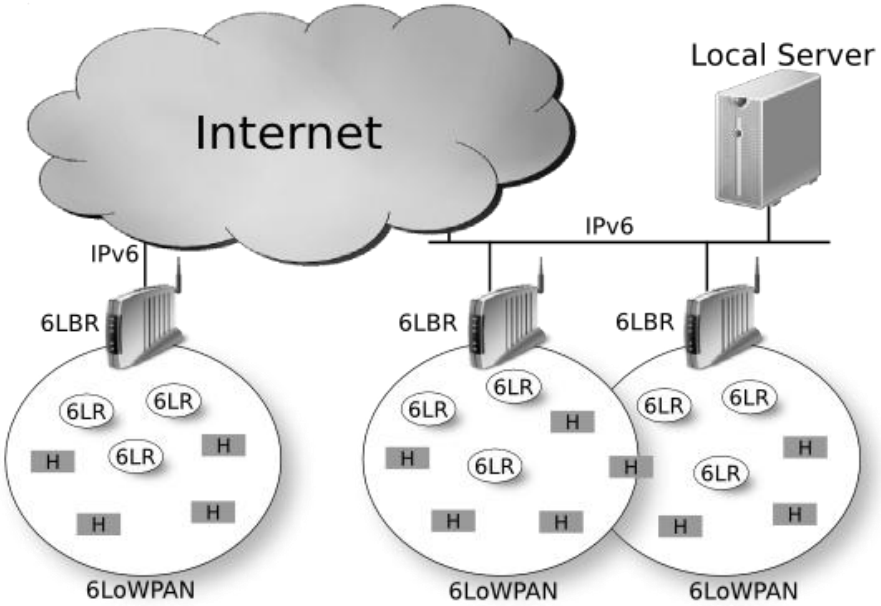


Support of third dimension, soft real-time, load balancing and flow shaping mechanisms

Combine IoT protocols with new technologies (1/3)

Video streaming support and

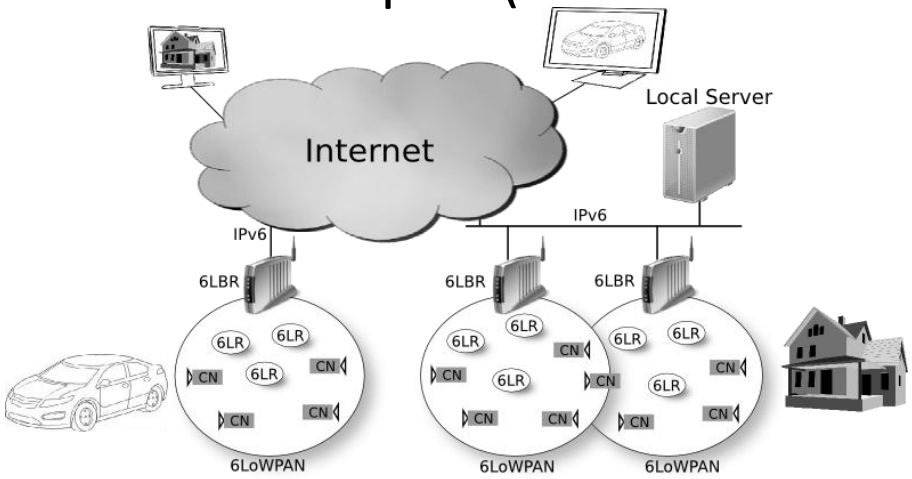
integration of Radio-Frequency IDentification in IoT



6LoWPAN reference architecture

Combine IoT protocols with new technologies (2/3)

1. **Video streaming:**
   - Constrained devices and limited transmission bandwidth
   - In-node video compression based on low-complexity computer vision techniques (Motion detection)

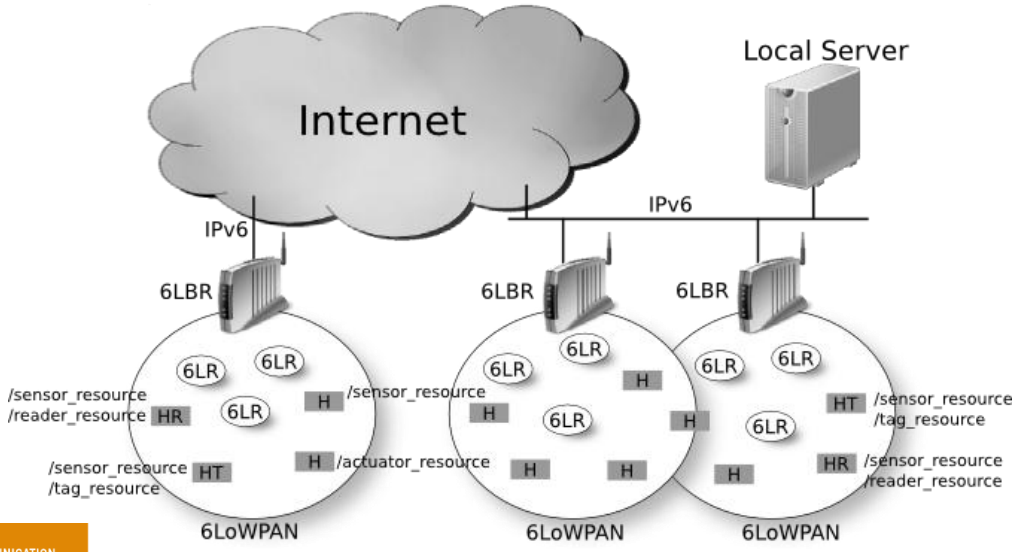


Camera Nodes in 6LoWPAN networks

Combine IoT protocols with new technologies (3/3)

2. Radio-Frequency IDentification:

- Increase the pervasiveness of WSNs
- Based on the REST paradigm
- Implemented and assessed for the Smart Factory use case



Reader (HR) and Tag (HT) in 6LoWPAN networks

Outline

1. Background and motivation
2. Contributions
3. **In-network processing and (Ease of) Reprogramming**
4. Conclusions

Motivations and the idea

Motivation

- Enable ease reprogramming of embedded devices
- Abstract embedded systems complexity and networking aspects
- Provide flexibility and reconfigurability

Idea

- To represent in-network processing applications as CoAP resources
- To use an easy programming language to develop new applications

New framework requirements

Design a **platform-independent solution**

- Native code cannot be used for defining the data-processing

Design a **feasible solution**

- Mote devices must be able to support it
- Resulting power consumption and application execution time must be acceptable

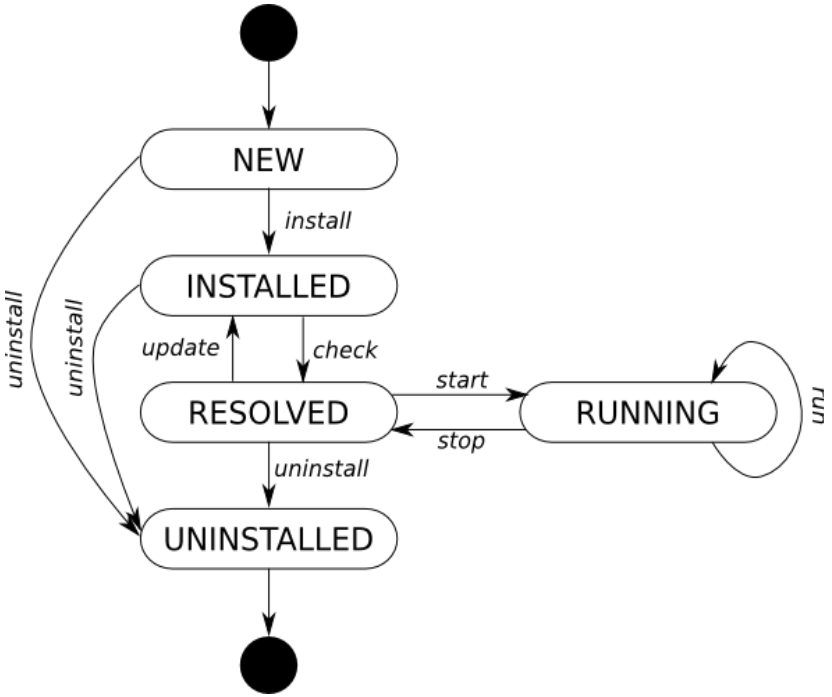
PyFUNS: Python Framework for Ubiquitous Networked Sensors

PyFUNS main components

- Finite State Machine:
  - Application life-cycle
- RESTful interface:
  - Application management
- APIs:
  - Interaction between application and Operating System

Finite State Machine

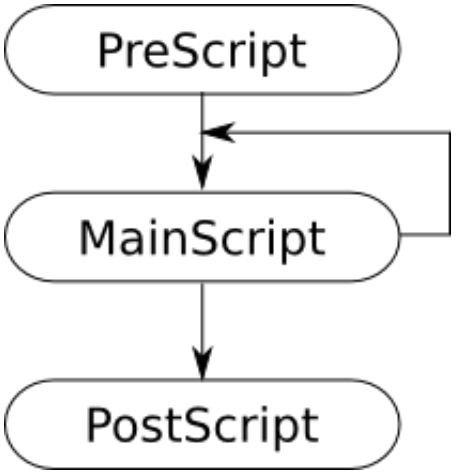
- Five possible states:
  - **NEW**: memory has been allocated successfully
  - **INSTALLED**: application has been installed
  - **RESOLVED**: application is ready to execute
  - **RUNNING**: application is active
  - **UNINSTALLED**: application has been removed



Application processing functions

Script: Python bytecode

- *PreScript (optional):* set up application environment
- *MainScript (mandatory):* core of the application
- *PostScript (optional):* clean application environment



RESTful interface (1/2)

Represent each application as a CoAP resource

- RESTful methods to manage resources

Resource	Content	Methods
/apps	List of installed applications	GET/PUT
/[app_name]	A specific application	GET/PUT/POST/DELETE
/period	Application period	GET/PUT
/preScript	PreScript bytecode	GET/PUT/DELETE
/version	Version of PreScript	GET/PUT
/mainScript	MainScript bytecode	GET/PUT/DELETE
/version	Version of MainScript	GET/PUT

RESTful interface (2/2)

Resource	Content	Methods
/postScript	PostScript bytecode	GET/PUT/DELETE
/version	Version of PostScript	GET/PUT
/variables	List of variables	GET
/[var_name]	A specific variable	GET/PUT

PyFUNS application is a CoAP resource

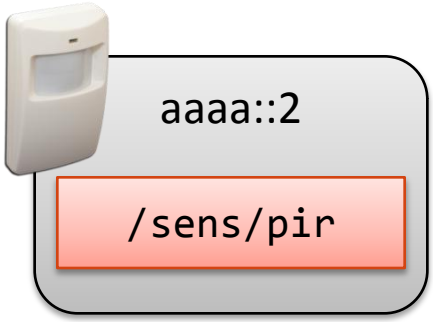
- Application management and execution through CoAP methods
- One application can start / stop another application

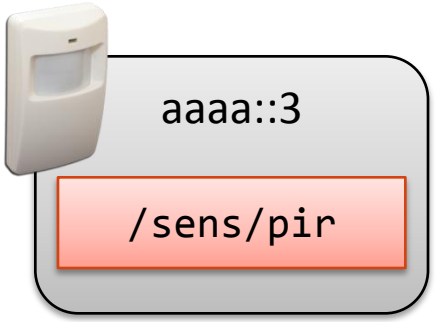
APIs

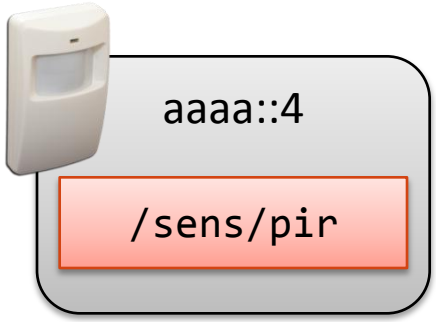
- Scripts are written in Python:
  - More precisely, a script is a Python script compiled into Python bytecode

API	Description
getVar (name)	Get the current value
setVar (name, value)	Set the value
sendMsg (method, address, uri, payload, variable)	Send CoAP message
obs (address, uri, variable)	Initialize the OBSERVE method
delObs (address, uri)	Delete the subscription

Example



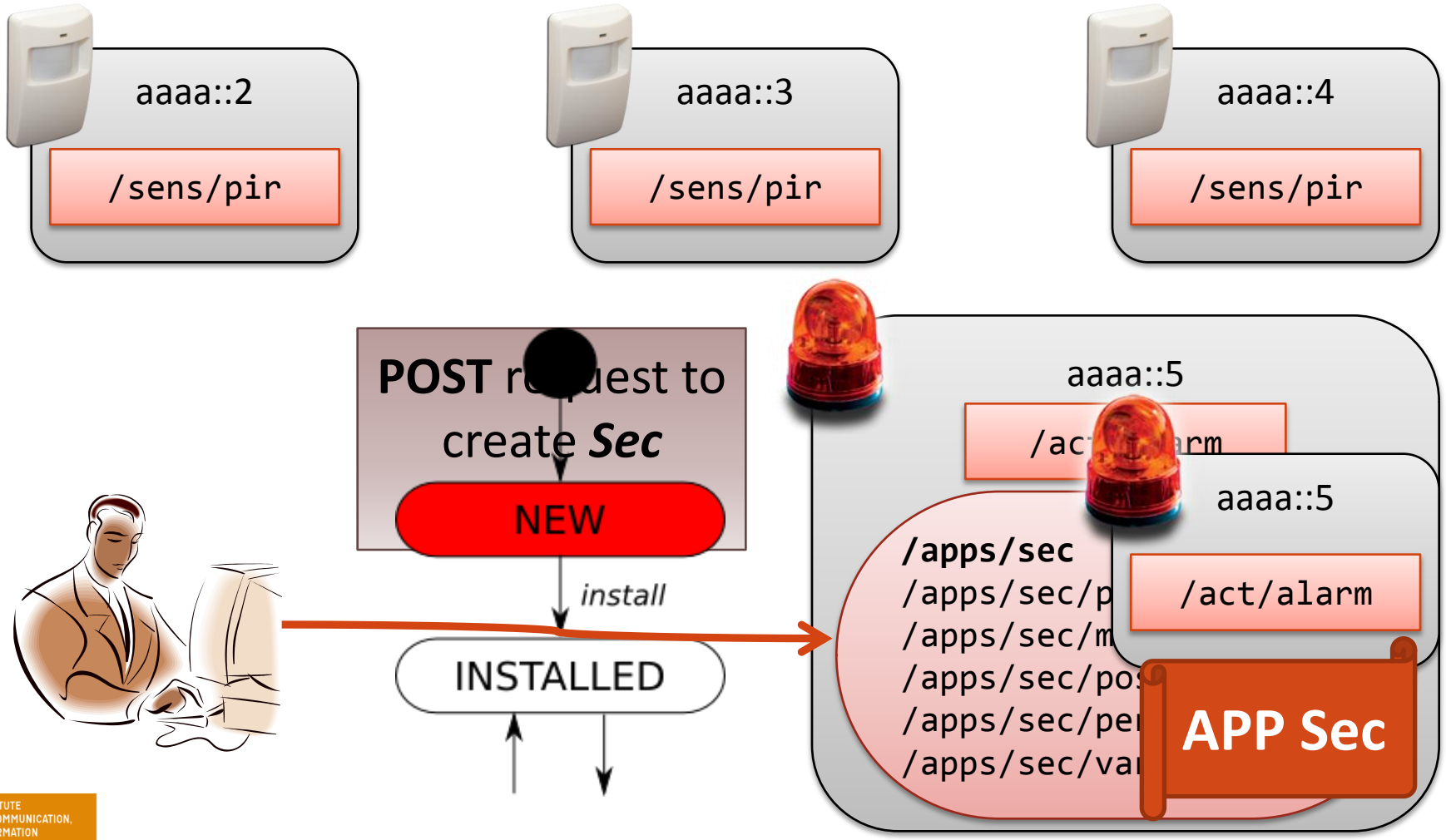




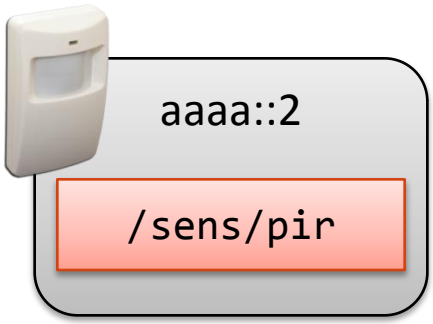
Detect any motion in a room and trigger an alarm

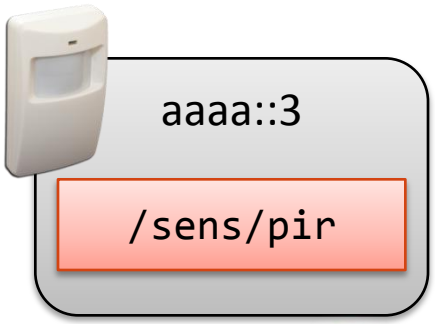
INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

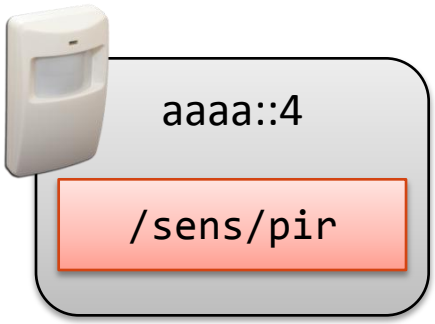
Create the application: NEW



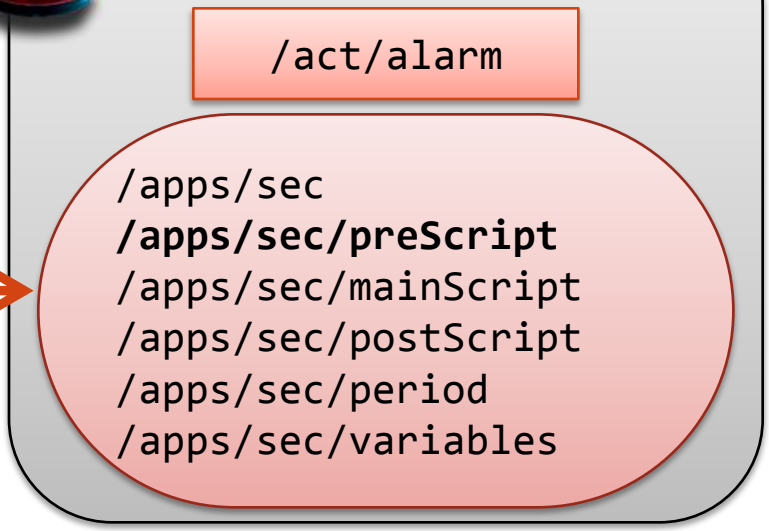
Install PreScript



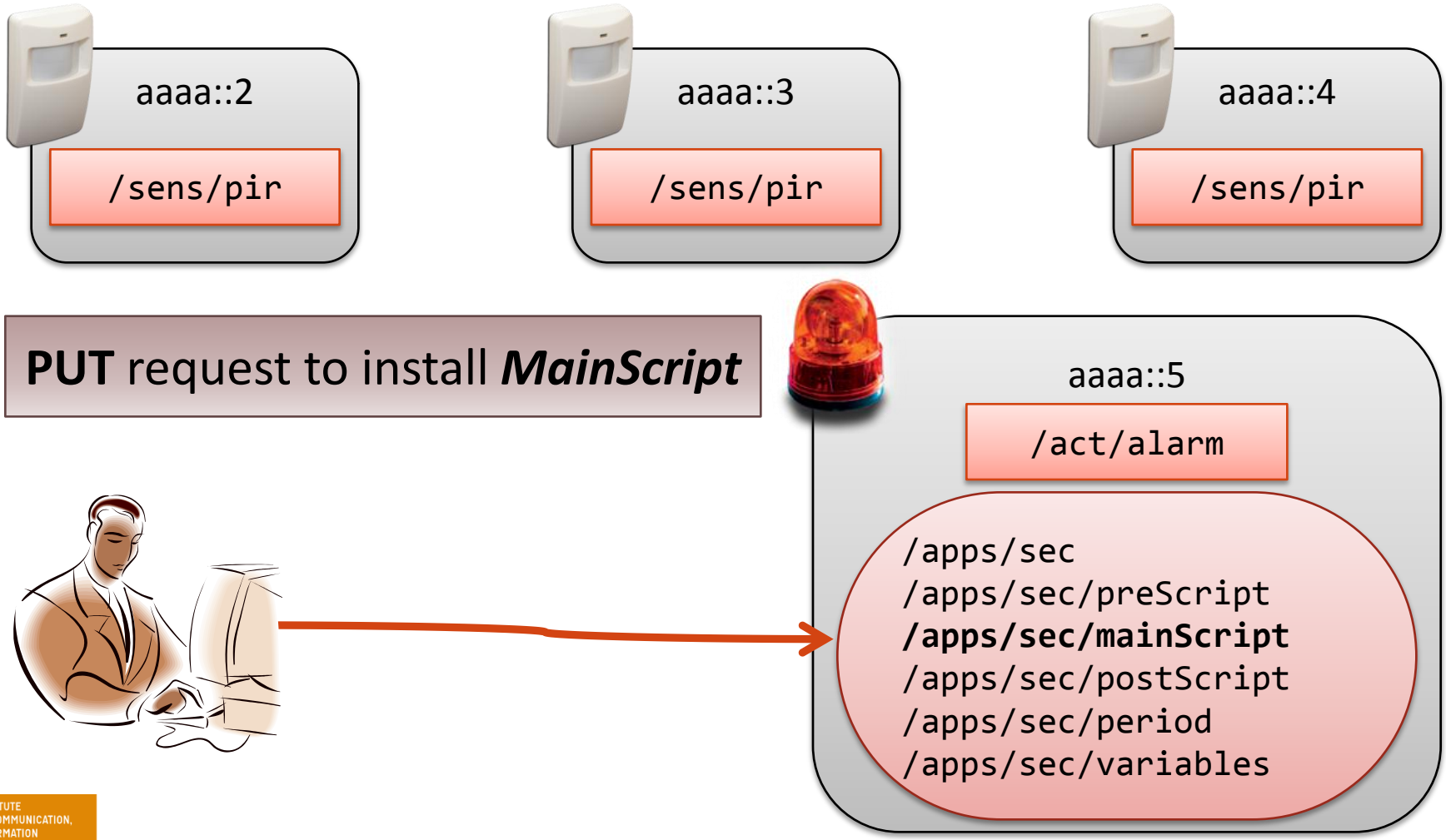




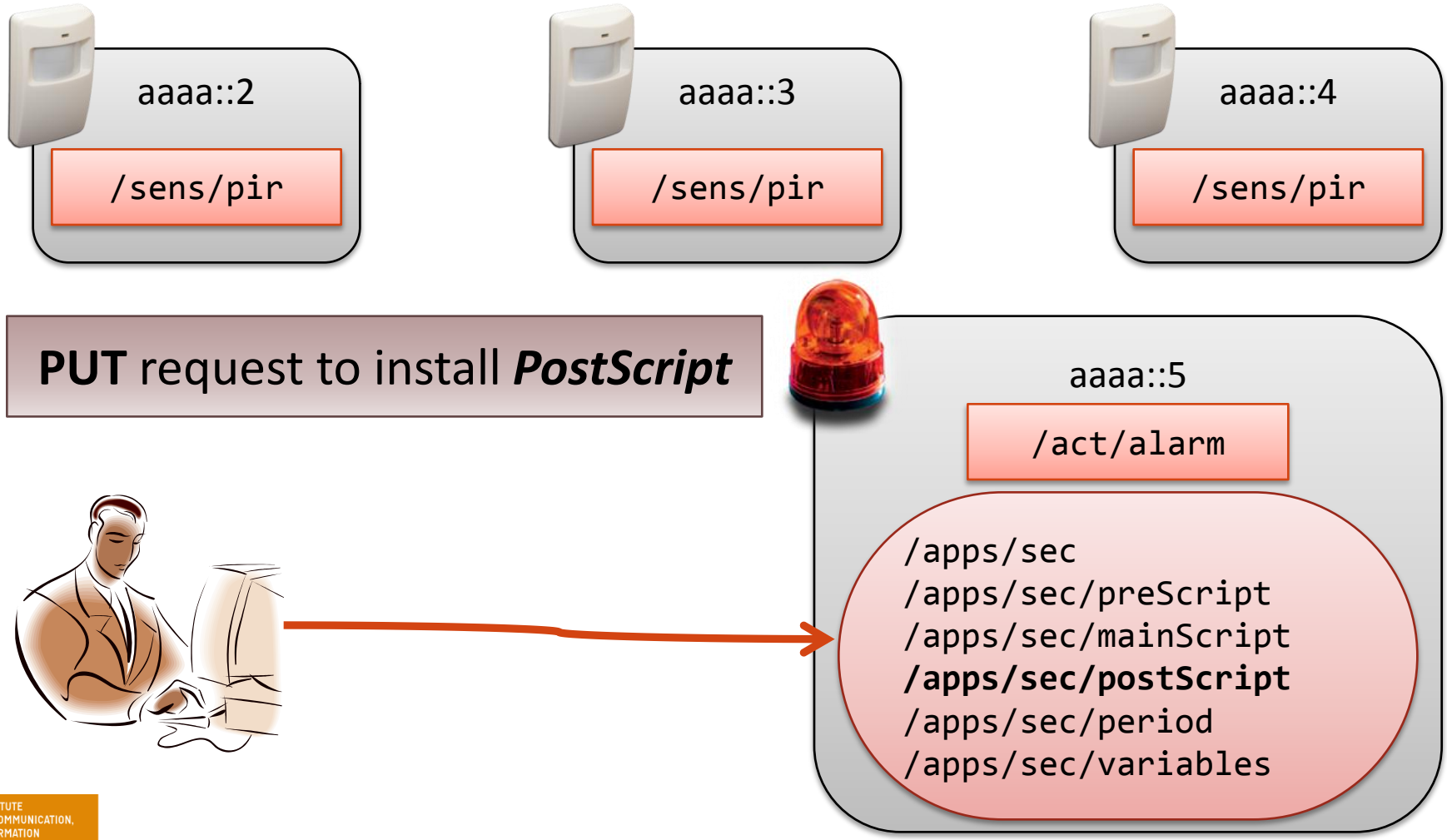
PUT request to install ***PreScript***



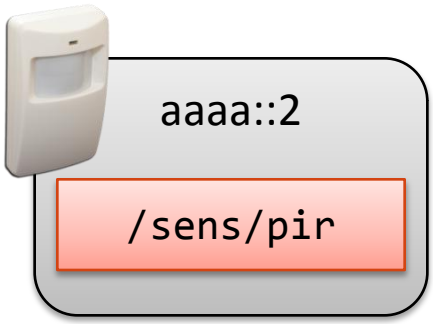
Install MainScript

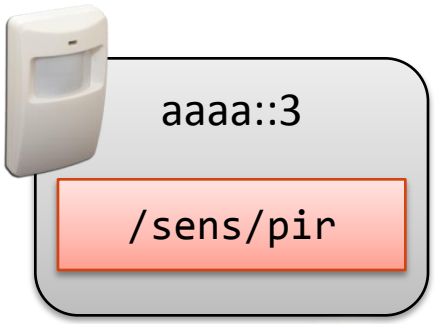


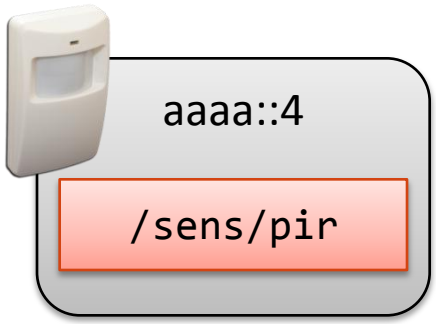
Install PostScript

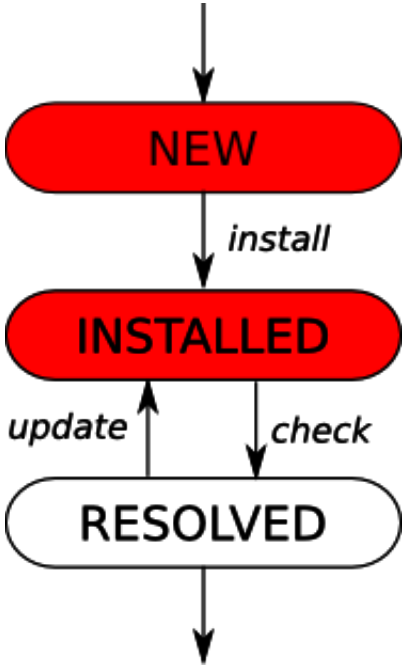


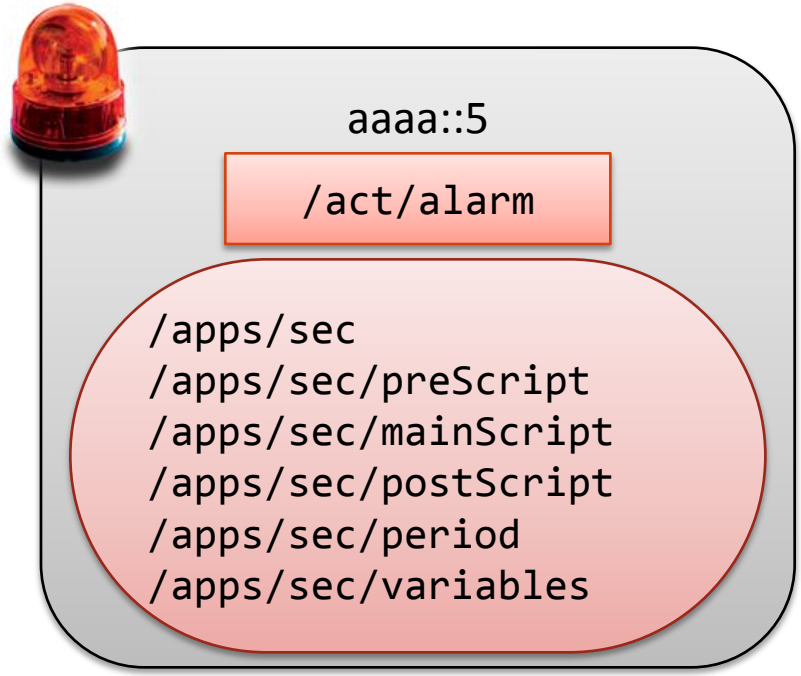
Scripts instelled: INSTALLED





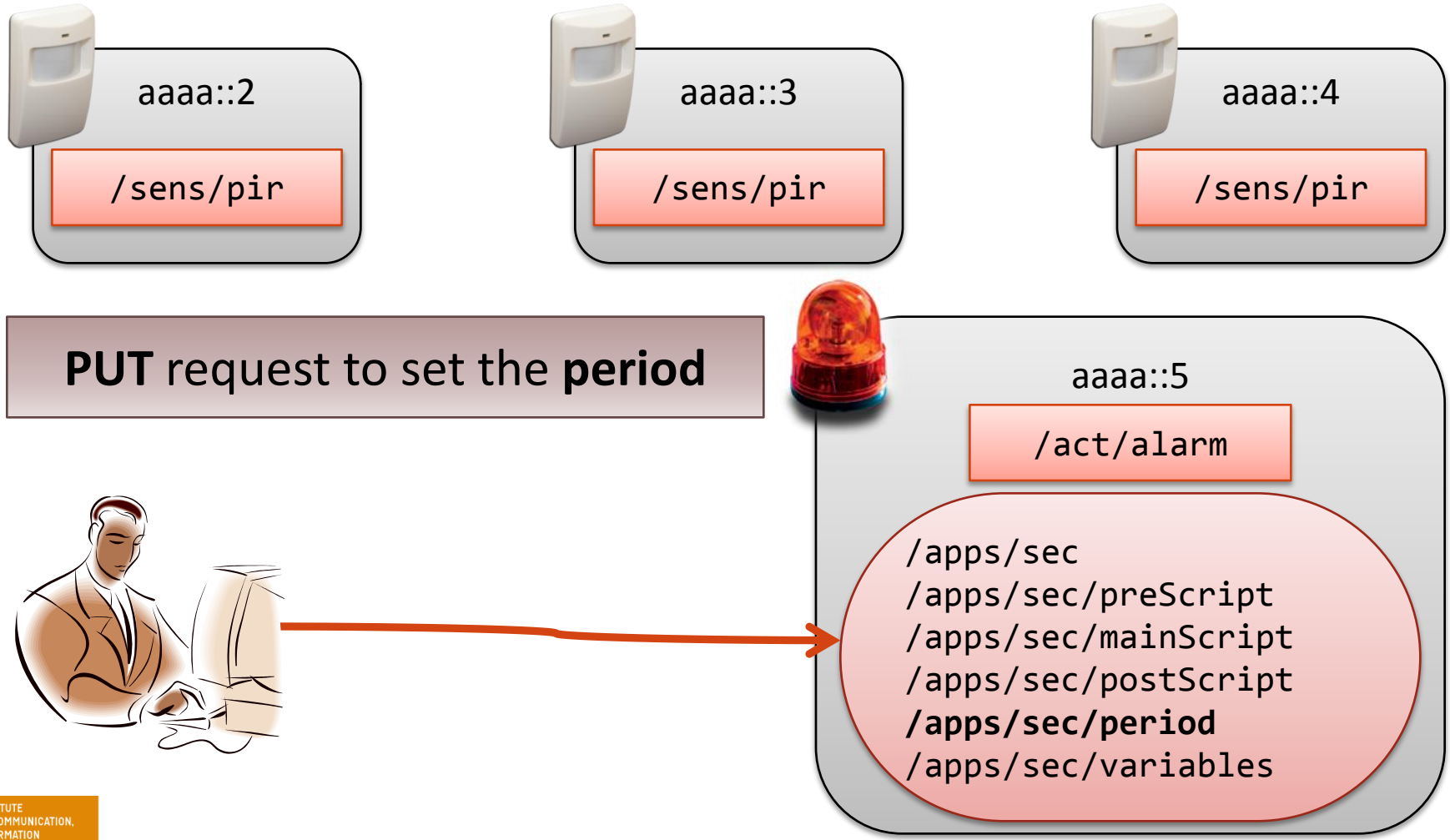




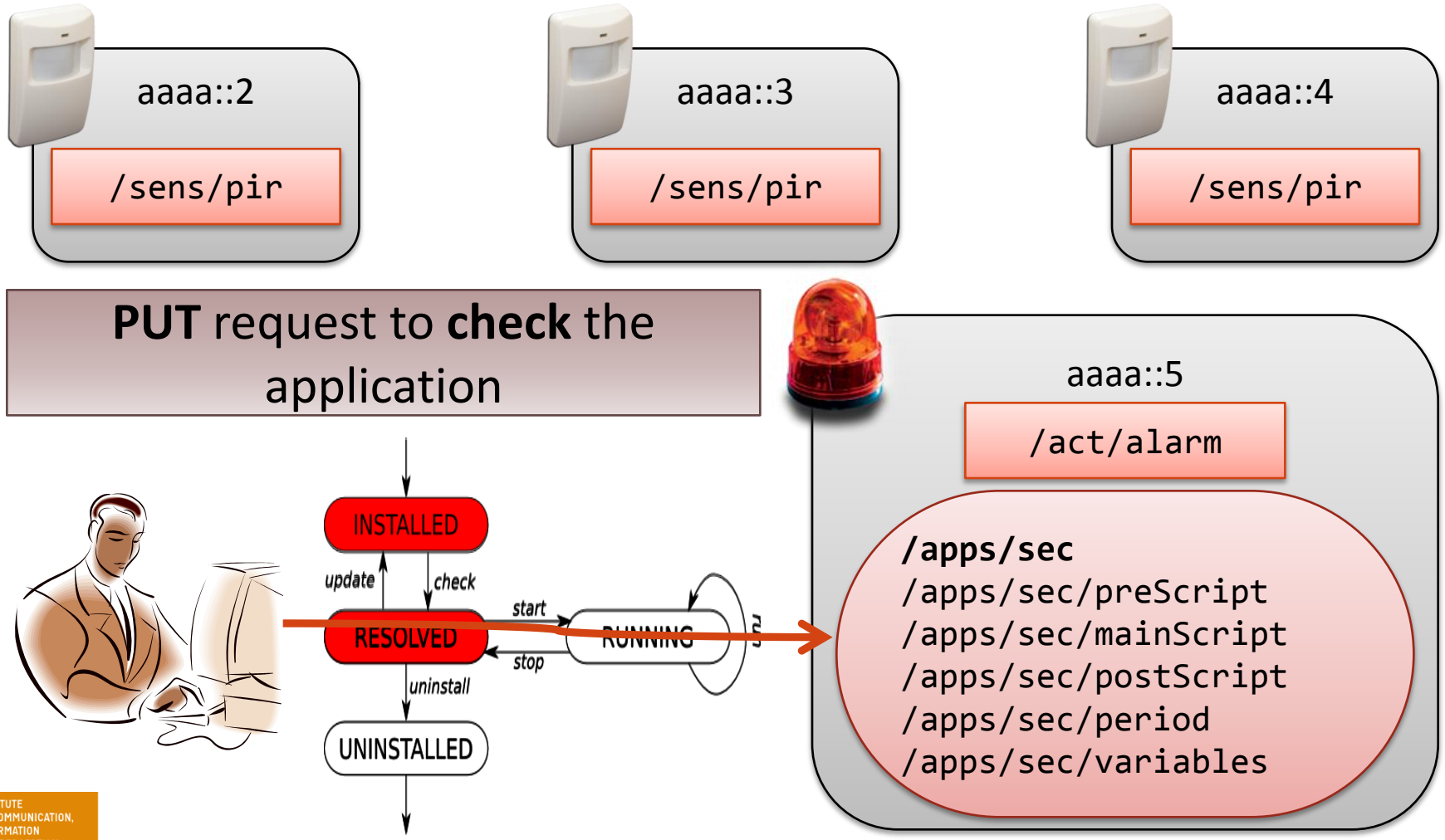


INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

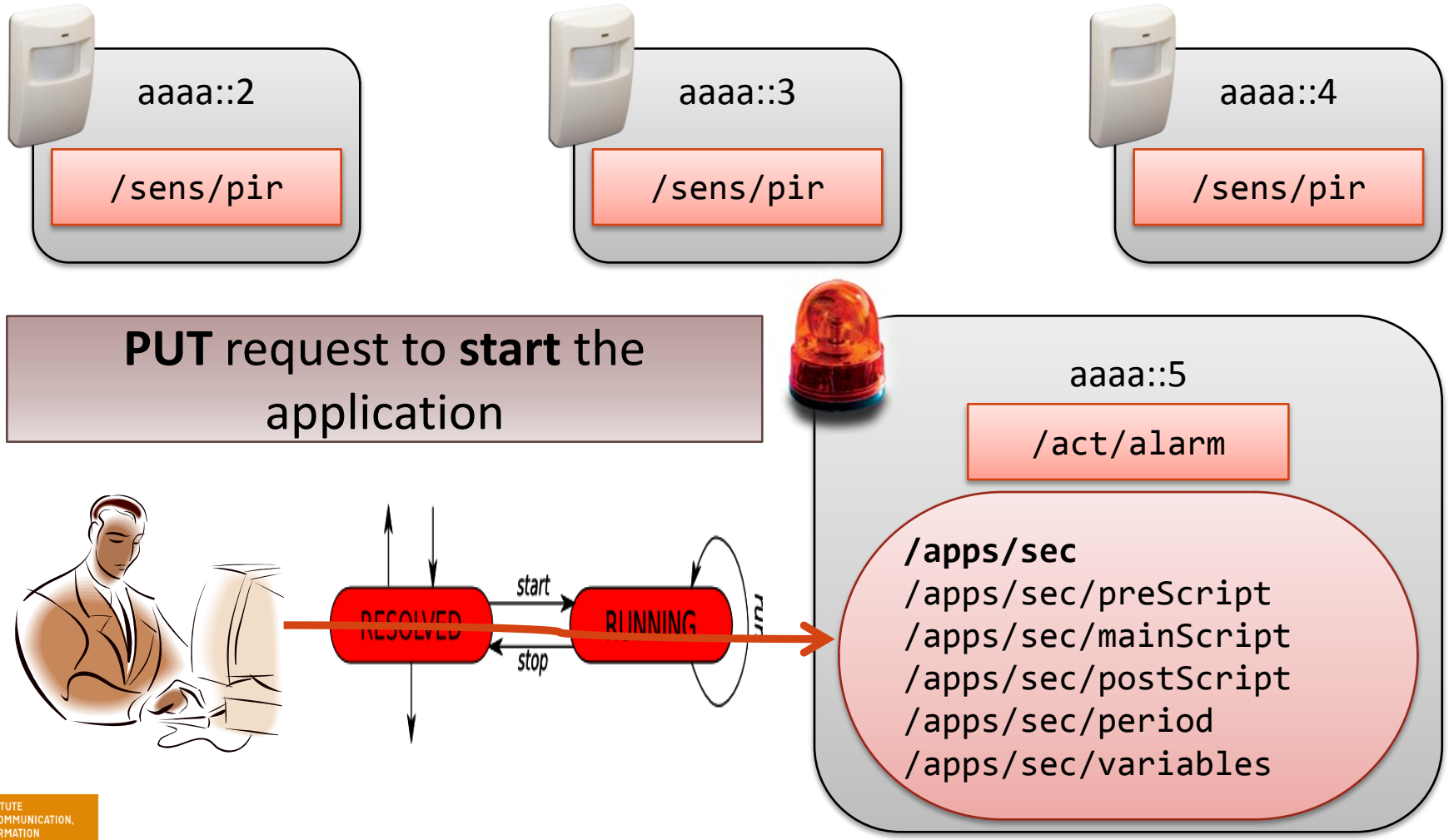
Set the period



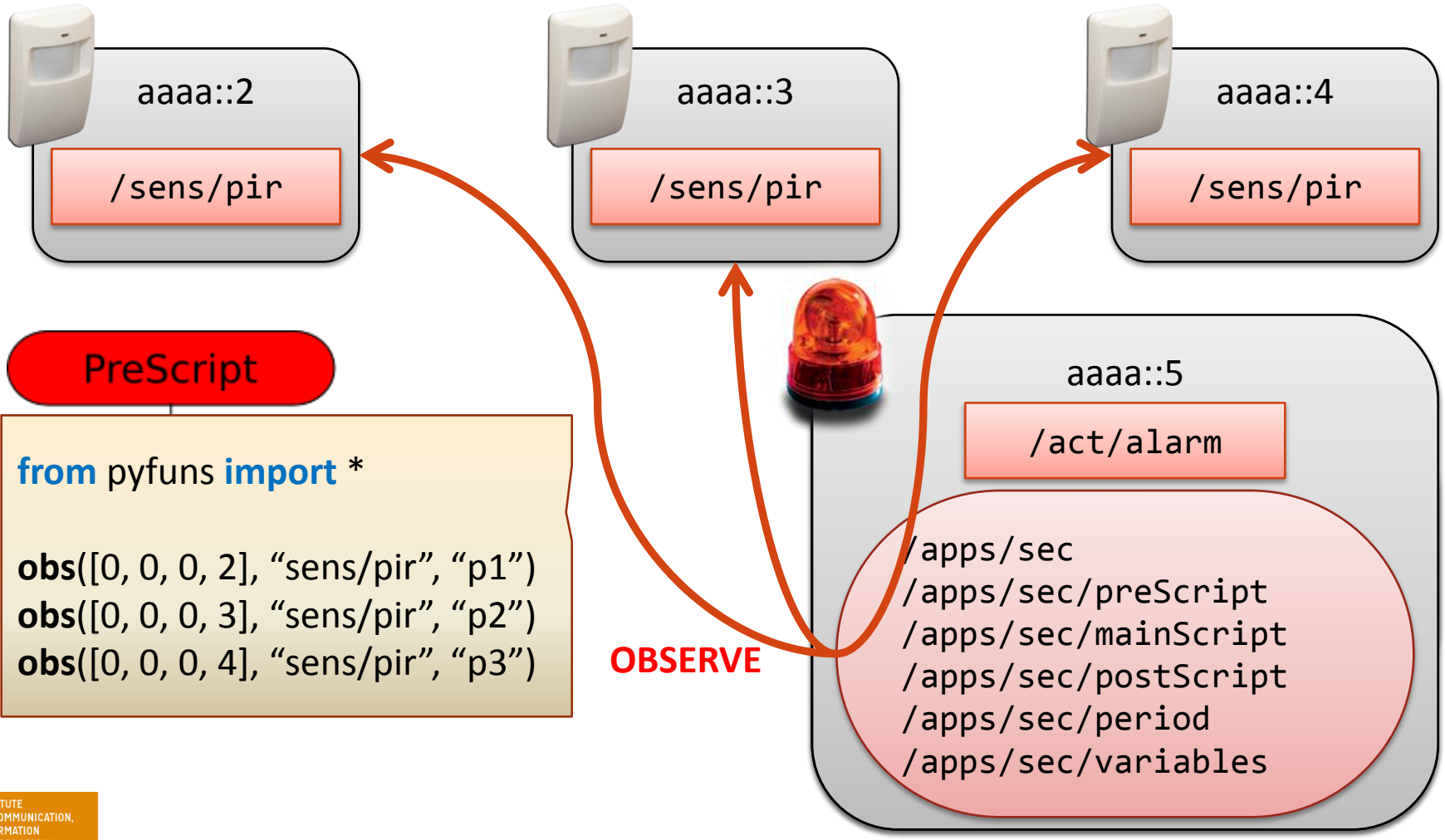
Check the application: RESOLVED



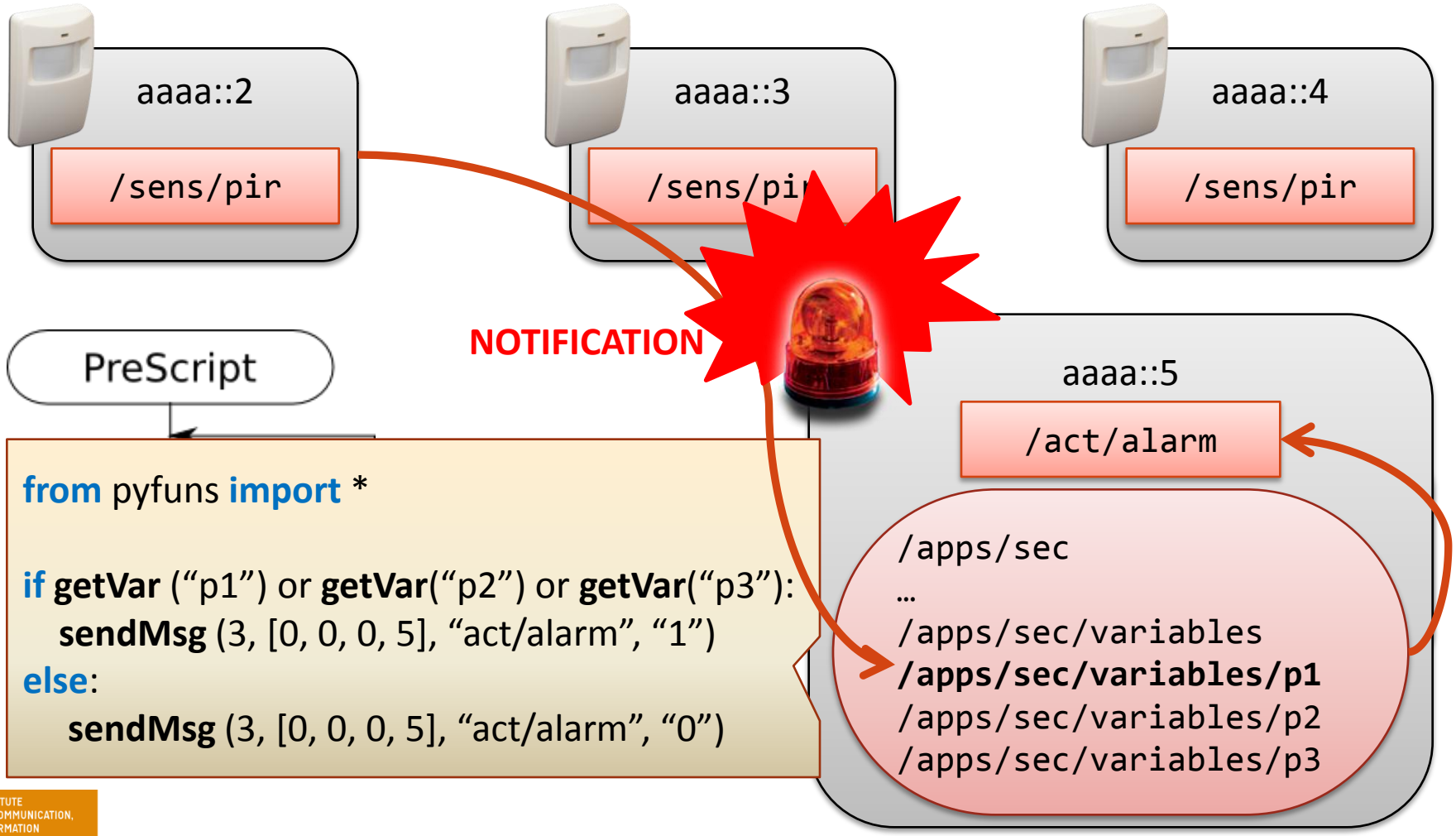
Start the application: RUNNING



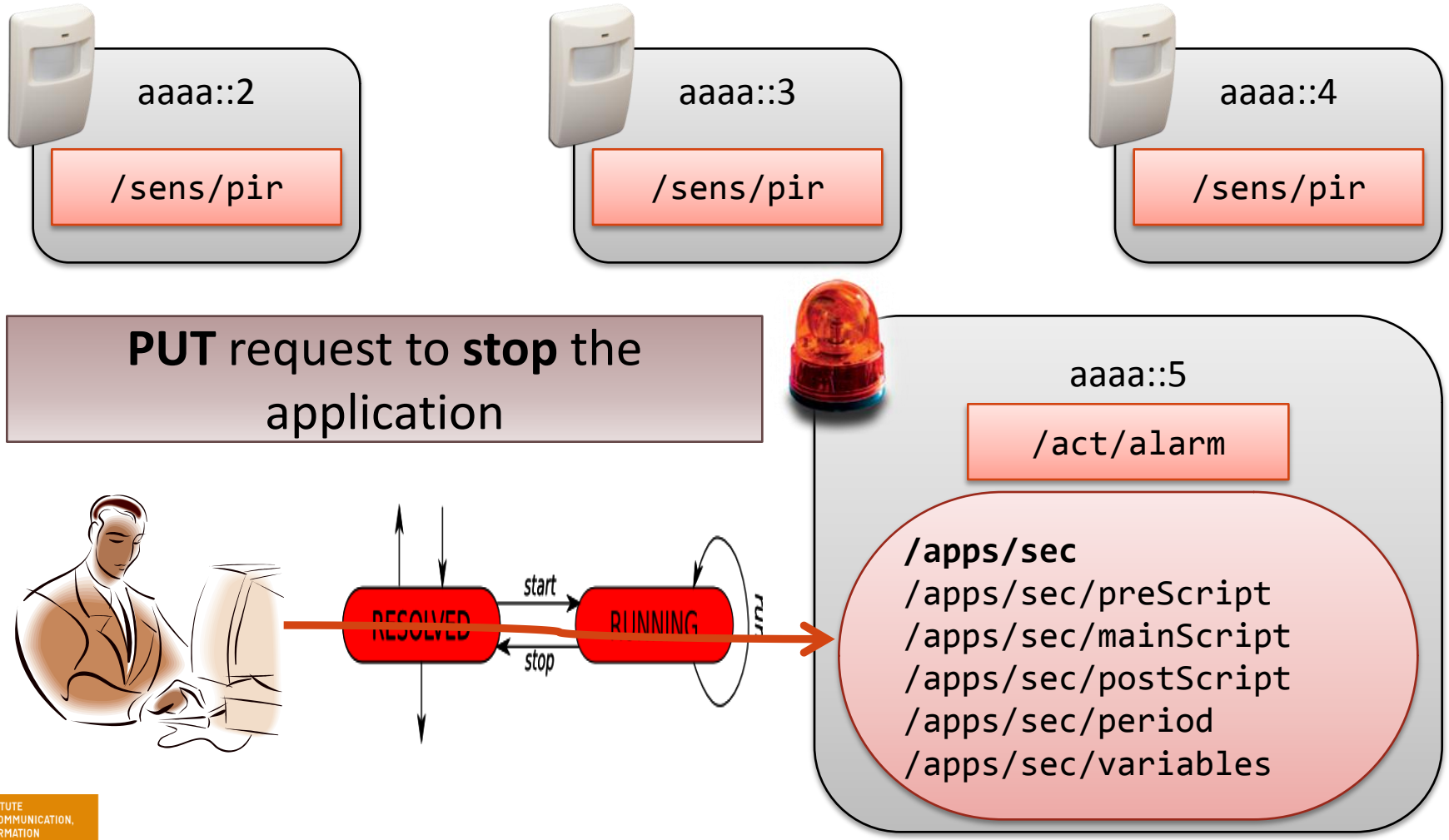
Run PreScript



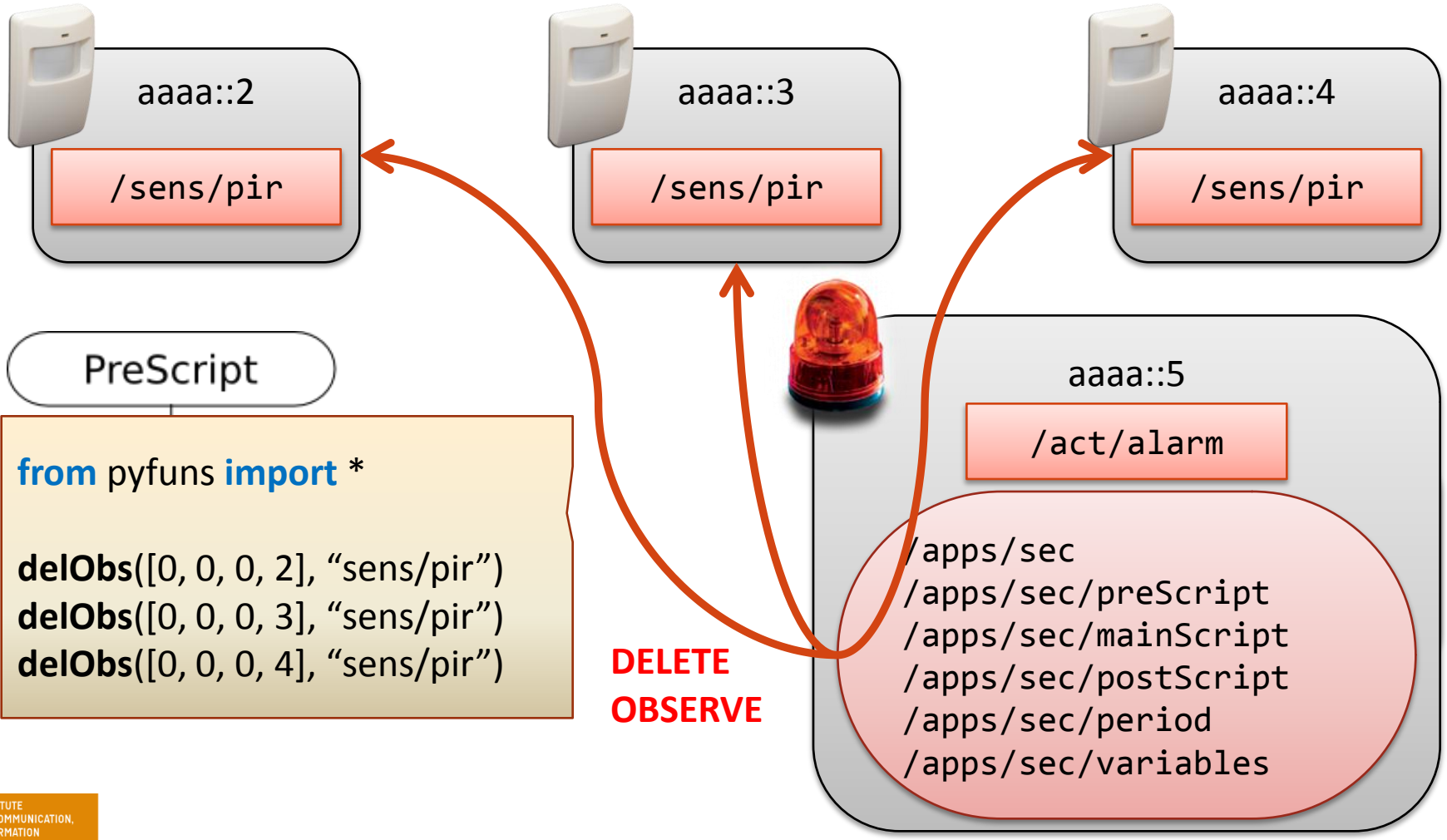
Run MainScript



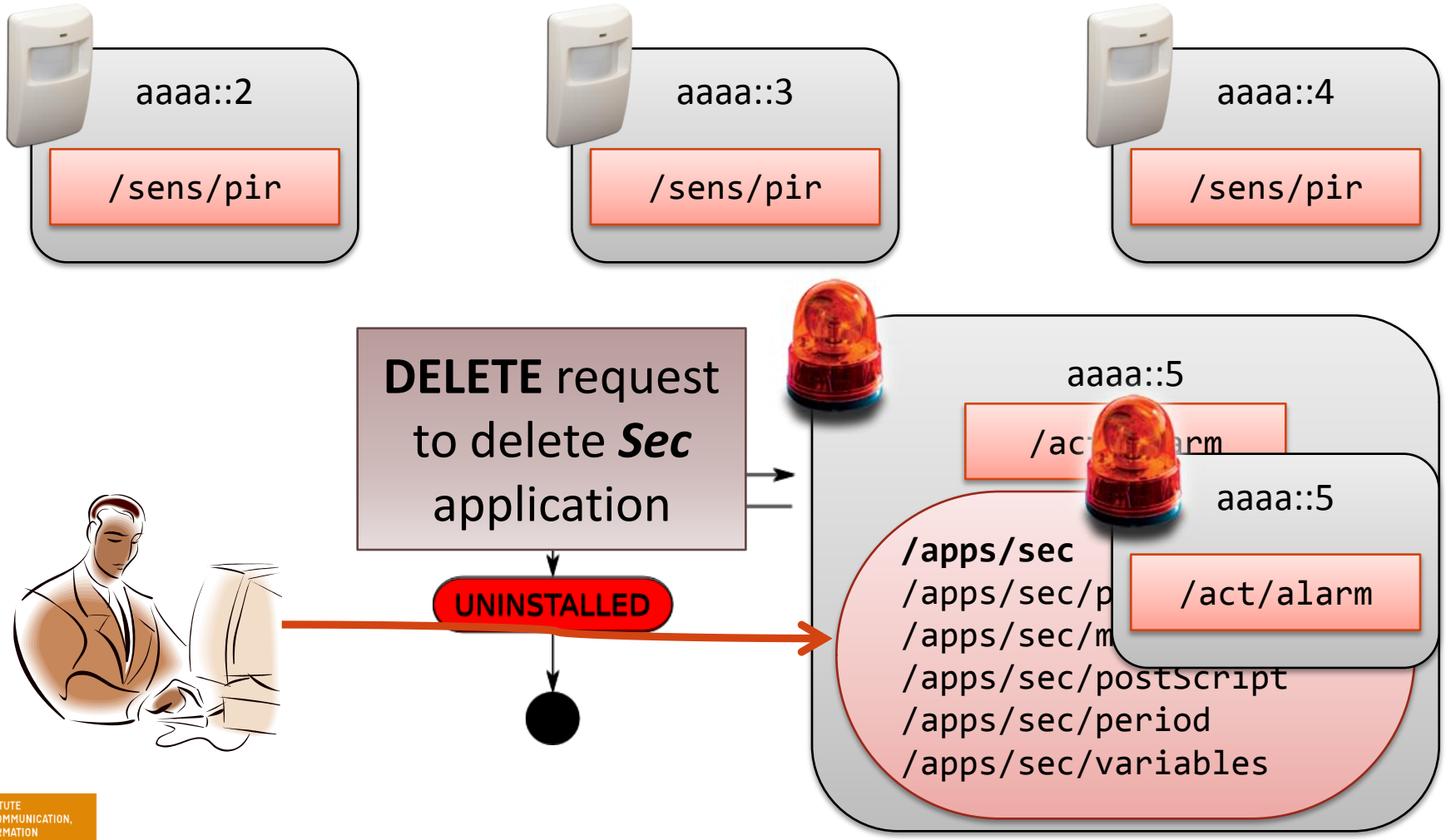
Stop the application: RESOLVED



Run PostScript



Delete the application



Implementation

- Built on top of the Contiki Operating System, extended with:
  - IPv6 loopback communication
  - Client-side support for CoAP observe
- PyMite used for running Python bytecode:
  - PyMite is a reduced Python interpreter for embedded systems
  - Integration of PyMite in Contiki

Development platform

WiSMote

CC2538 EM

	WiSMote	CC2538 EM
CPU Frequency	16 MHz	32 MHz
RAM	16 kB	32 kB
Flash Memory	256 kB	512 kB

Memory requirements with PyFUNS

WiSMote

CC2538 EM

	WiSMote	CC2538 EM
RAM	~ 15 kB (93%)	~ 20 kB (62%)
Flash Memory	~ 98 kB (38%)	~ 97 kB (19%)

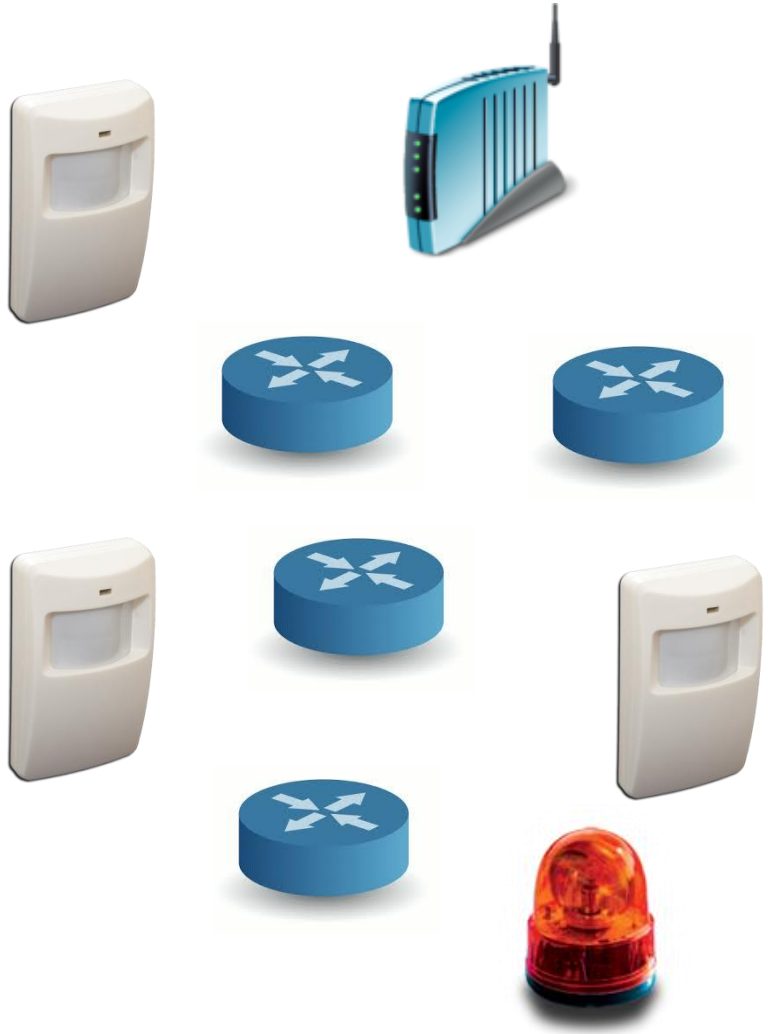
Native Code vs Python

- Experiments:
  - Send a PUT request
  - Send a PUT request and wait ACK message
  - Send a GET request, wait the reply, send a PUT request and wait ACK message

	C		Python		Python / C	
	Time (ms)	Energy (µJ)	Time (ms)	Energy (µJ)	Time (ms)	Energy (µJ)
PUT	1,22	0,009	5,35	0,039	4,4	4,3
PUT2	8,61	0,328	16,68	0,357	1,4	1,1
GET	17,26	0,604	26,19	0,671	1,5	1,1

Evaluation setup

- Security application:
  - 1 Border Router
  - 3 PIR sensors
  - 1 Alarm node
  - 4 Routers
- Evaluation of distributed (in-network processing) and centralized approach
- Metrics:
  - Network traffic
  - Actuation delay
  - Energy consumption
- Simulation using Cooja

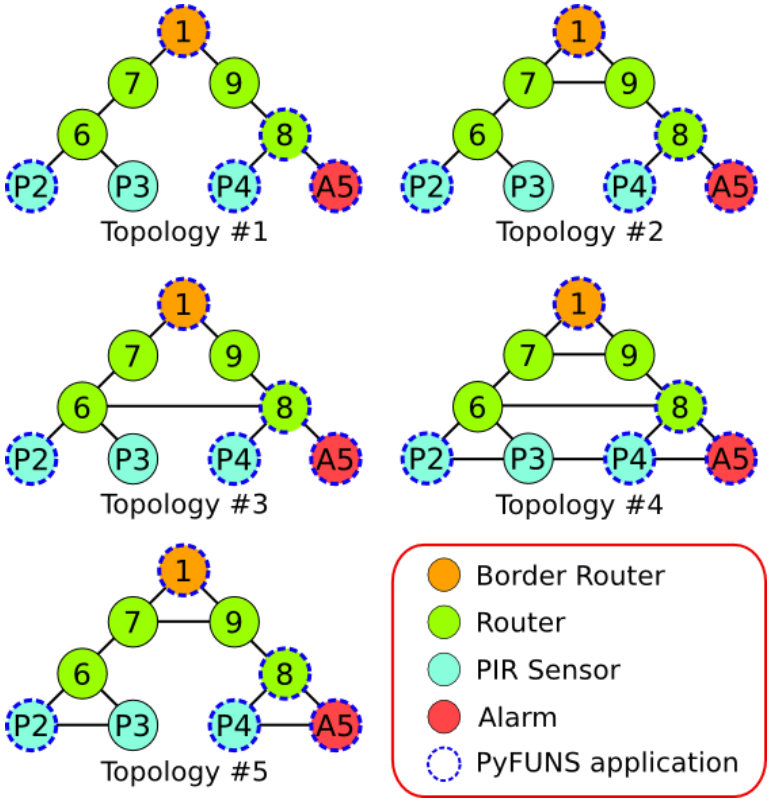


INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

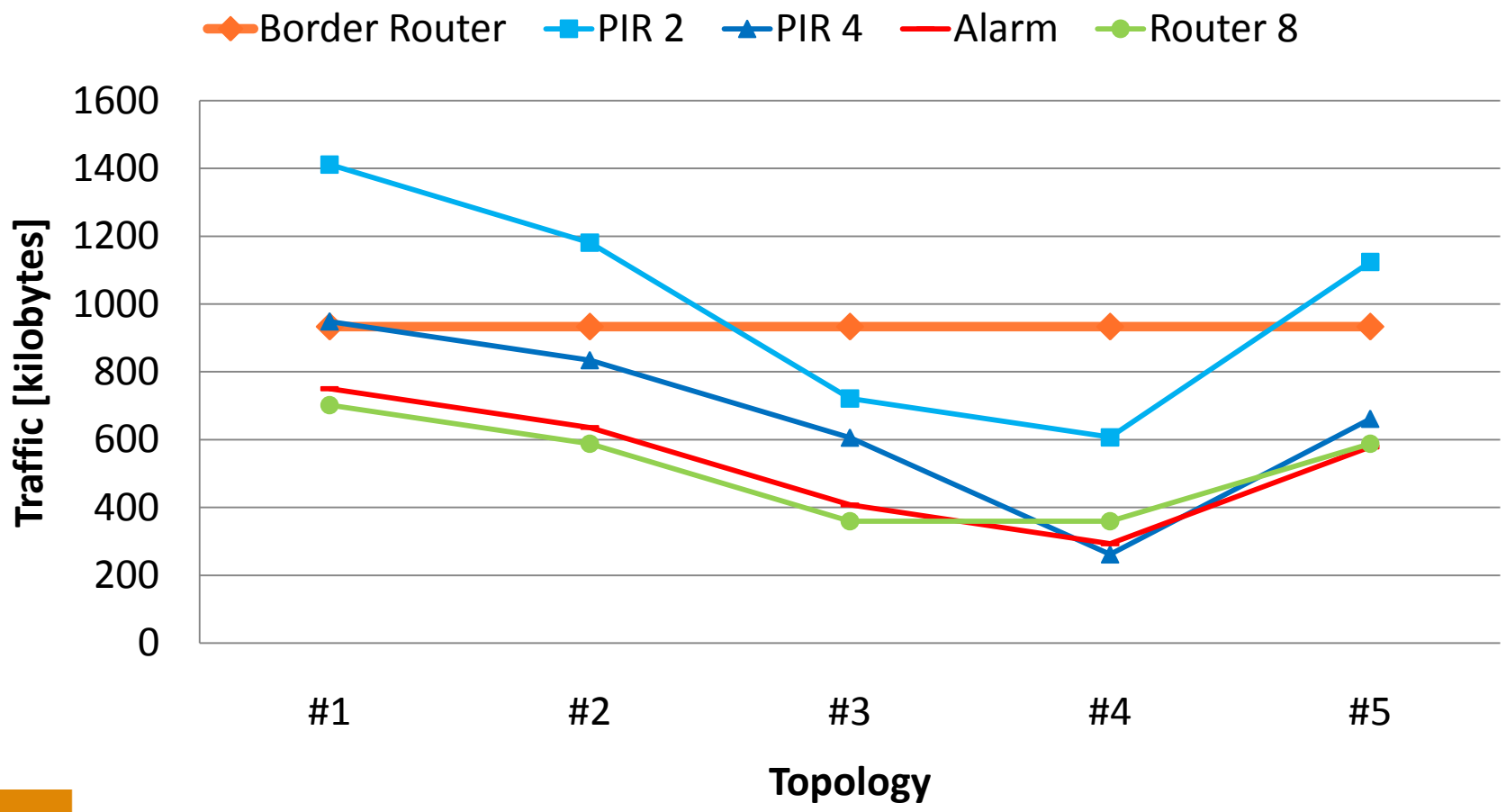
Scuola Superiore Sant'Anna

Evaluation setup

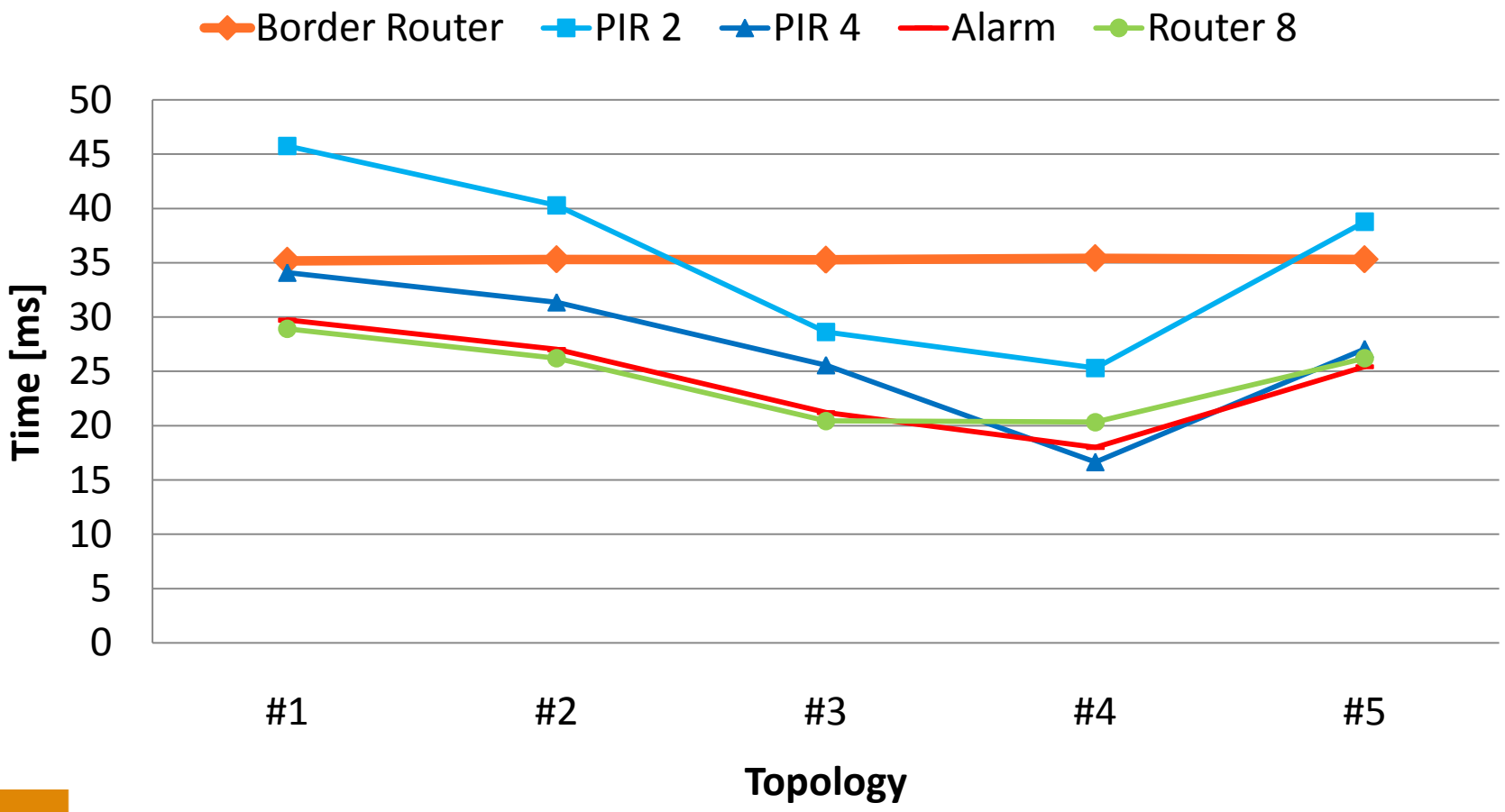
- Five different topologies
  - Static routing
- Application is installed on:
  - Border Router (centralized approach)
  - 2 PIR sensors (Node P2, P4) (in-network processing)
  - Alarm node (in-network processing)
  - 1 Router (Node 8) (in-network processing)



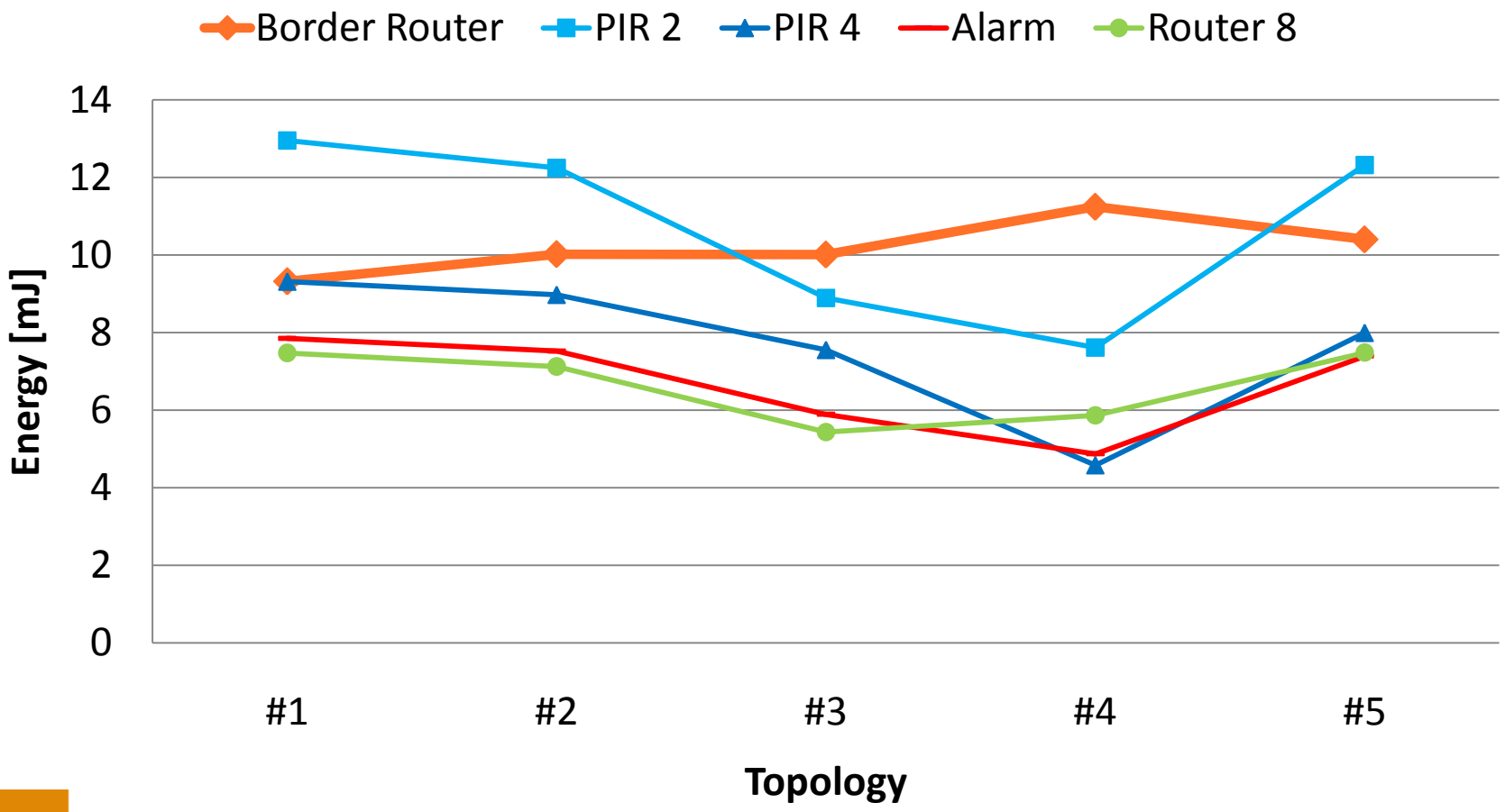
Network traffic



Actuation delay



Energy consumption



Summarizing …

- PyFUNS enables reconfigurable and reprogrammable in-network processing in IoT-based WSNs
- PyFUNS can run on constrained devices
  - E.g., WiSMote, CC2538dk
- Python vs Native Code:
  - Advantages:
    - Flexibility
    - Abstraction
    - Ease of reprogramming
  - Disadvantages:
    - Computation time
    - Energy consumption
- PyFUNS allows to reduce network traffic, energy consumption and actuation delay in respect of centralized approach (Border Router)

PyFUNS final remarks

- PyFUNS simplifies the development of IoT applications
  - PyFUNS abstracts embedded systems complexity and networking aspects
- PyFUNS allows to deploy dynamically distributed services
  - Applications can be moved anytime
- PyFUNS is a framework that can be used to extend WSN lifetime:
  - reducing network traffic and energy consumption

Outline

1. Background and motivation
2. Contributions
3. In-network processing and (Ease of) Reprogramming
4. **Conclusions**

Conclusions

Internet of Things is the next step for the Internet growth

- Main challenges treated:
  - Protocol adaptation
    - SPEED-6LoWPAN, SPEED-3D
  - Combine IoT protocls with new technologies
    - Video streaming, RFID
  - In-network processing and (Ease of) reprogramming
    - PyFUNS

List of publications (1/3)

- **S. Bocchino**, M. Petracca, P. Pagano, M. Ghibaudi, and G. Lipari
  SPEED-3D: a Geographic Routing Protocol for 6LoWPAN Networks
  to appear in International Journal of Ad Hoc and Ubiquitous Computing.
- M. Petracca, C. Salvadori, **S. Bocchino**, and P. Pagano
  Error Resilient Video Streaming with BCH Code Protection in Wireless Sensor Networks
  Journal of Communications Software and Systems, vol. 10, no. 1, pp. 31-40, March 2014.
- C. Salvadori, M. Petracca, **S. Bocchino**, R. Pelliccia, and P. Pagano
  A low-cost vehicle counter for next-generations ITS
  Journal of Real Time Image Processing, March 2014, doi: 10.1007/s11554-014-0411-4
- C. Salvadori, M. Petracca, S. Madeo, **S. Bocchino**, and P. Pagano
  Video streaming applications in wireless camera networks: A change detection based approach targeted to 6LoWPAN
  Journal of Systems Architecture, May 2013 (ISSN: 1383-7621)
- M. Petracca, **S. Bocchino**, A. Azzarà, R. Pelliccia, M. Ghibaudi and P. Pagano
  WSN and RFID Integration in the IoT scenario: an Advanced safety System for Industrial Plants
  Journal of Communications Software and Systems, vol. 9, no. 1, pp. 104-113, March 2013 (ISSN: 1845-6421)

List of publications (2/3)

- A. Azzarà, D. Alessandrelli, **S. Bocchino**, M. Petracca, and P. Pagano
  PyoT: a Macroprogramming Framework for the Internet of Things
  Proceedings of IEEE International Symposium on Industrial Embedded Systems, Pisa, June 2014.
- A. Azzarà, **S. Bocchino**, P. Pagano, G. Pellerano, and M. Petracca
  Middleware solutions in WSN: the IoToriented approach in the ICSI project
  Proceedings of IEEE International Conference on Software Telecommunications and Computer Networks, Split, Croatia, September 2013.
- P. Pagano, C. Salvadori, S. Madeo, M. Petracca, **S. Bocchino**, D. Alessandrelli, A. Azzarà, M. Ghibaudi, G. Pellerano, and R. Pelliccia
  A Middleware of Things for supporting distributed vision applications
  Proceedings of the 1st Workshop on Smart Cameras for Robotic Applications , Vilamoura, Algarve, Portugal, October 2012.
- C. Salvatore, **S. Bocchino**, M. Petracca, R. Pelliccia, M. Ghibaudi, and P. Pagano
  WSN and RFID Integrated Solution for Advanced Safety Systems in Industrial Plants
  Proceedings of IEEE International Conference on Software Telecommunications and Computer Networks, Split, Croatia, September 2012.

List of publications (3/3)

- A. Azzarà, D. Alessandrelli, **S. Bocchino**, M. Petracca, and P. Pagano
  Architecture, functional requirements, and early implementation of an instrumentation grid for the IoT
  Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC), Liverpool, UK, June 2012.
- B. Abid, M. Petracca, P. Pagano, **S. Bocchino**, and D. Alessandrelli
  Experimental validation of wireless localization techniques in IEEE802.15.4 networks
  Proceedings of 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, September 2011.
- **S. Bocchino**, M. Petracca, P. Pagano, M. Ghibaudi and F. Lertora
  SPEED routing protocol in 6LoWPAN networks
  Proceedings of IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA), Tolouse, France, September 2011.
- E. D. Gutierrez Mlot, **S. Bocchino**, A. Azzarà , M. Petracca, and P. Pagano
  Web services transactions in 6LoWPAN networks
  IEEE International Symposium on World of Wireless Mobile and Multimedia Networks (WoWMoM), June 2011.

THANK YOU!

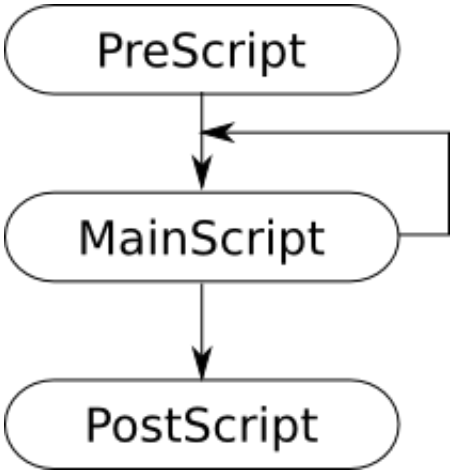
PYFUNS

Application structure (1/2)

1. Name: uniquely identifies the application
2. Period:
   - *Greater than zero:* periodic application
   - *Equal to zero:* one time execution application
   - *Less than zero:* application blocked waiting for an answer
3. Timer: used in case of periodic application
4. State: current state

Application structure (2/2)

5. Script: Python bytecode
   - *PreScript (optional):* set up application environment
   - *MainScript (mandatory):* core of the application
   - *PostScript (optional):* clean application environment



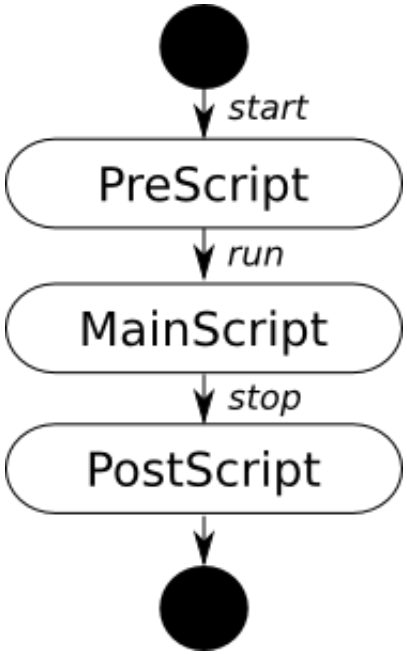
6. Variables: list of variables to store data

Application Life-cycle

Current State	Input	Next State	Output
NEW	install uninstall	INSTALLED UNINSTALLED	Scripts have been installed Application deleted
INSTALLED	check uninstall	RESOLVED UNINSTALLED	Application ready to execute Application deleted
RESOLVED	start update uninstall	RUNNING INSTALLED UNINSTALLED	Start the application Changes in installed scripts Application deleted
RUNNING	run stop	RUNNING RESOLVED	Wait for next execution Stop the application

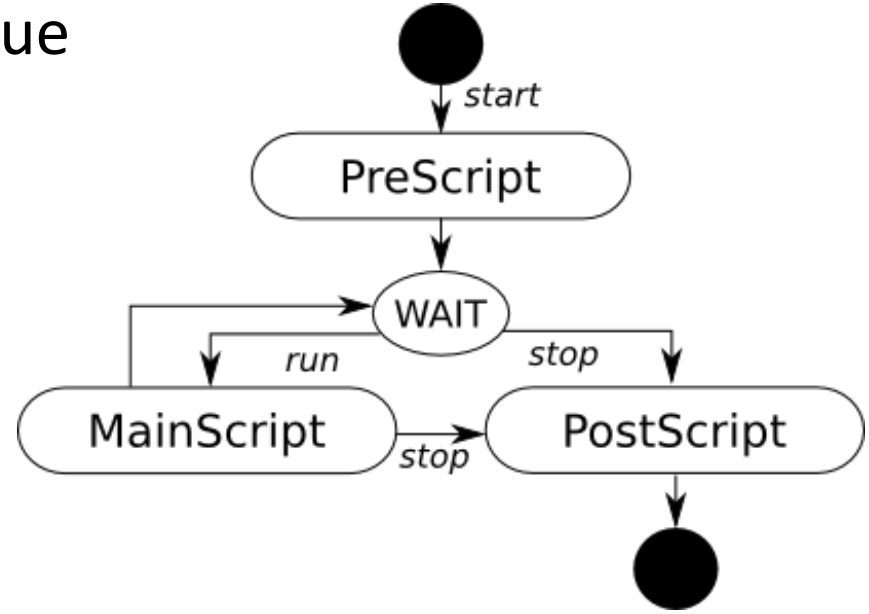
Application Flow: Period = 0

- From *PreScript* to *PostScript* directly
- *MainScript* runs one time
- Purpose:
  - Change the resource representation only one time



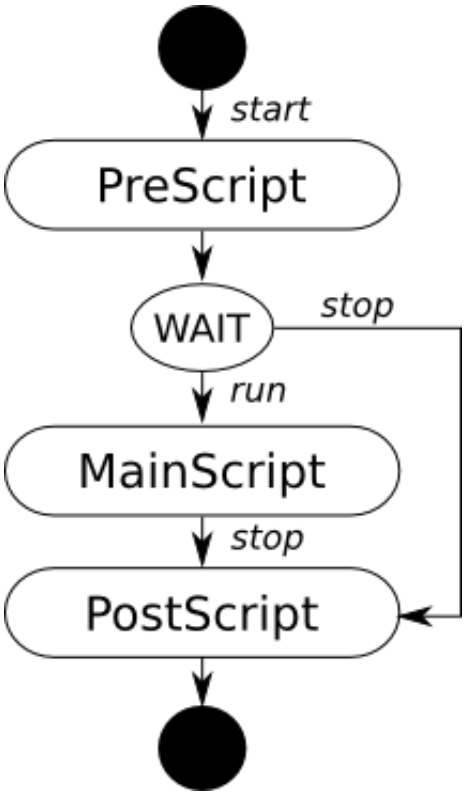
Application Flow: Period != 0

- After *PreScript*, the application waits for an event to continue its execution
- Two types of events:
  - Periodic
  - Sporadic
- Purpose:
  - Change the resource representation periodically
  - Perform activities in case of observed resources changes

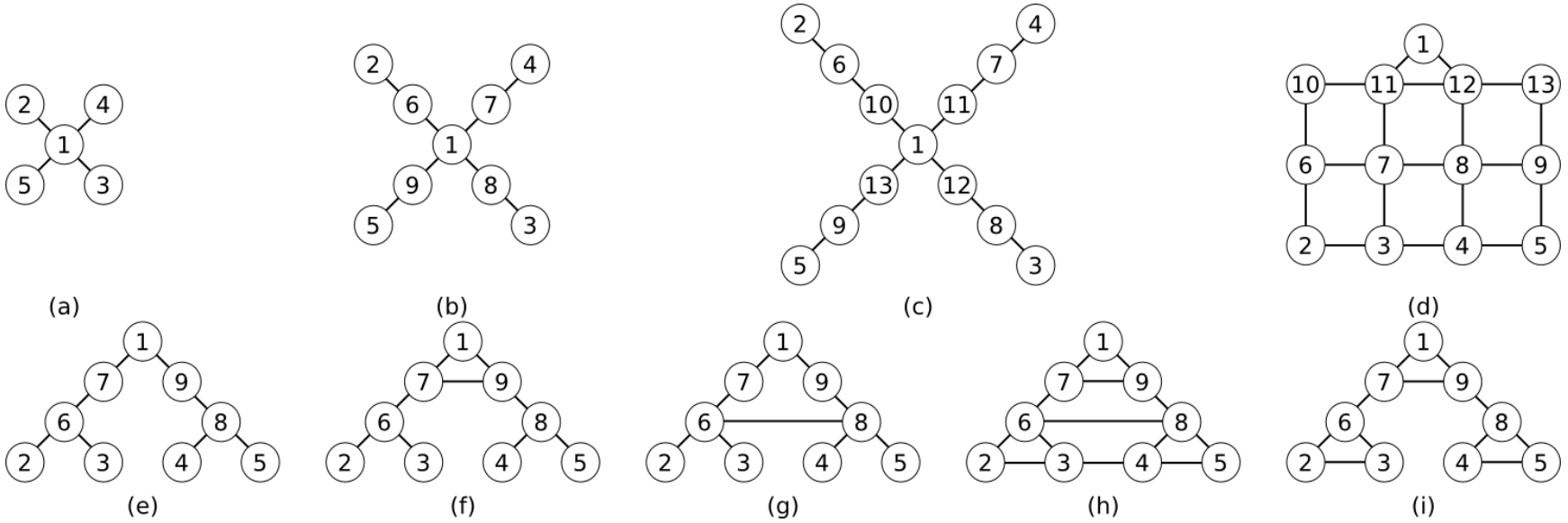


Application Flow: Period < 0

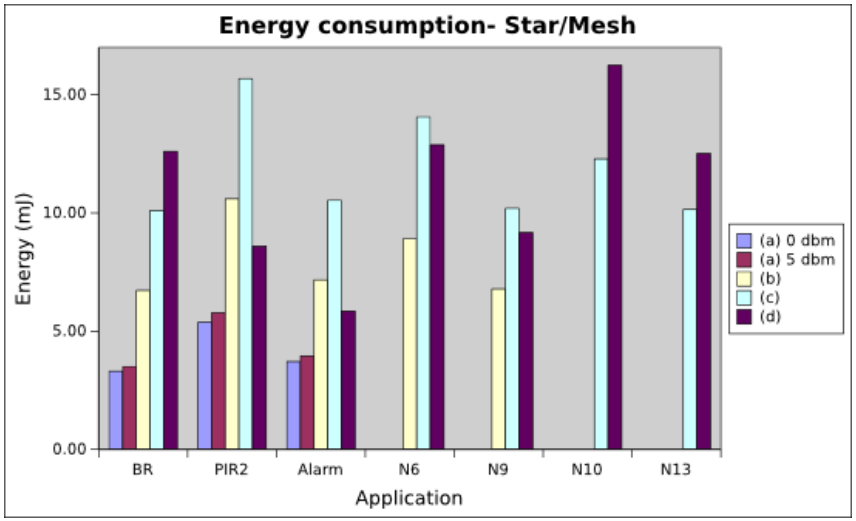
- Particular application flow based on a period less than zero
- *MainScript* runs once after sporadic event occur
- Purpose:
  - Retrieve resources representation and perform activities

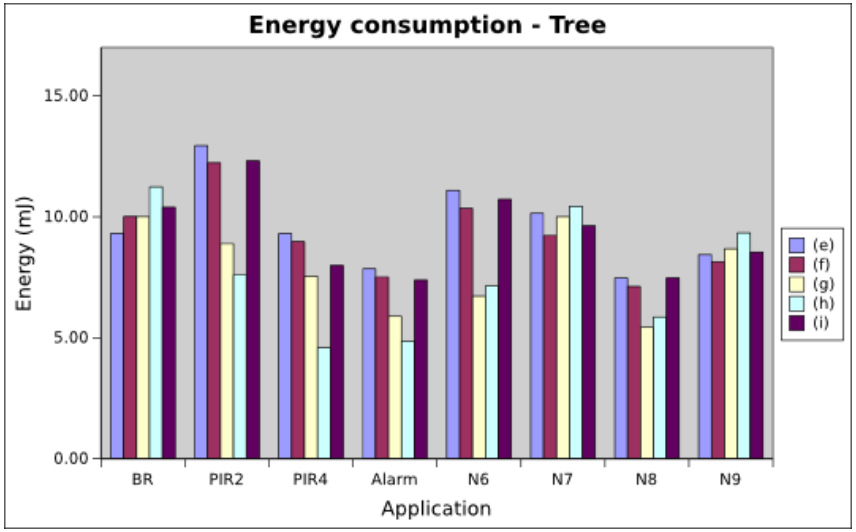


Topologies

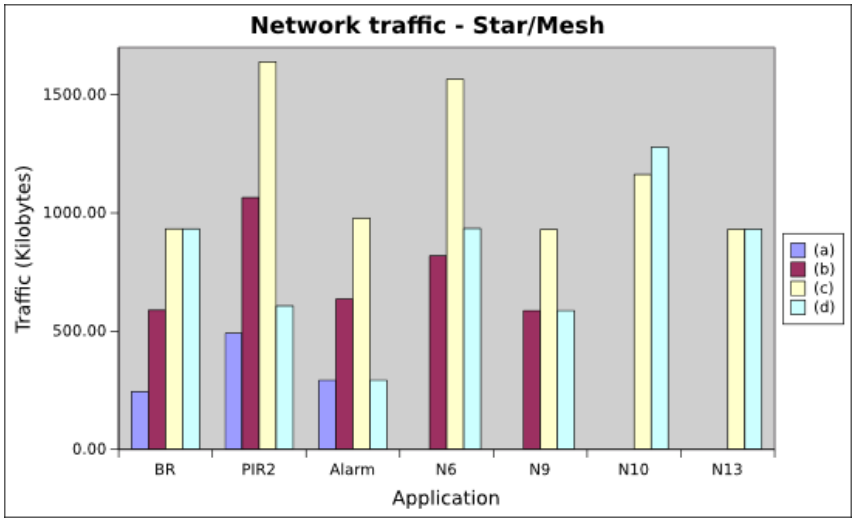


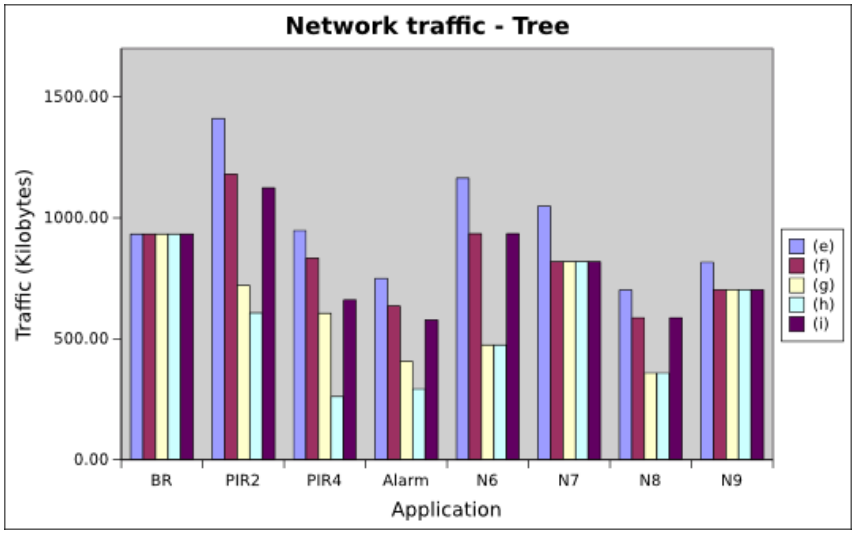
Security: Energy Consumption



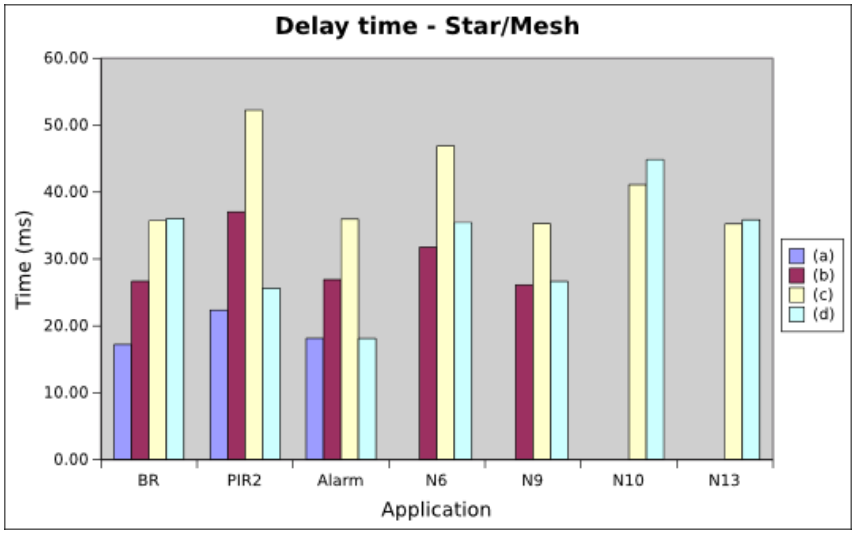


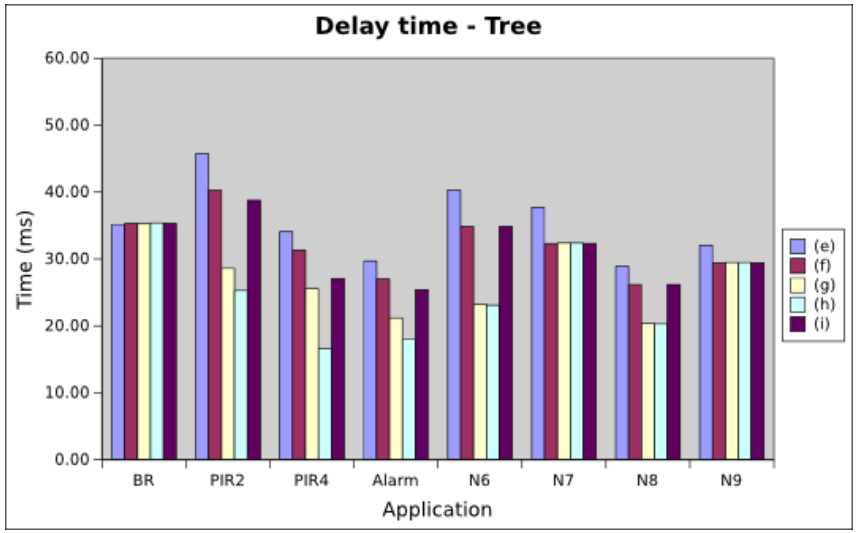
Security: Network Traffic



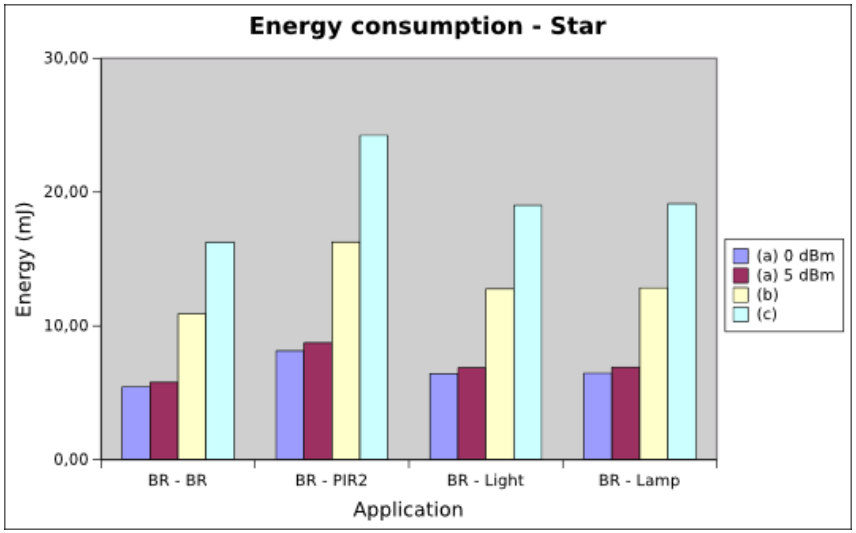


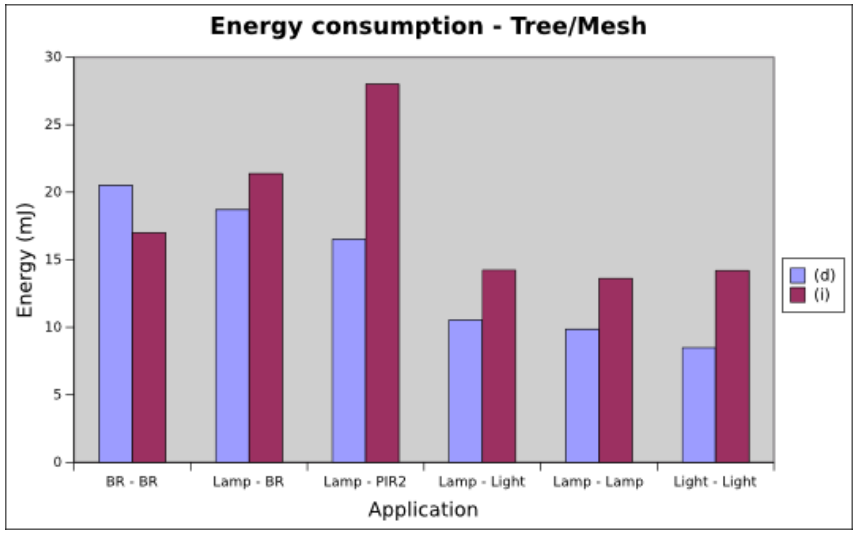
Security: Actuation Delay





Light: Energy Consumption





PyFUNS extension

- PyFUNS decouples IoT services from the sensing infrastructure:
  - Application can be installed everywhere
  - Tool to manage applications deployment
- PyFUNS can be integrated with a *centralized* framework:
  - PyoT

PyoT: a macro-programming framework for IoT-based WSNs

Python-based macro-programming

- resources are abstracted as Python objects
- communication details are completely hidden

Other features

- Automatically discover available resources
- Monitor sensor data
- Handle its storage
- Interact with resources using scripting language

```
temps = Resource.objects.filter(uri='temp')
results = []
for temp in temps:
  results.append(temp.GET())
avg = sum(results)/len(results)
if avg > 24:
  Resource.objects.get(ip='2001::1',
                       uri='fan').PUT('on')
```

PyFUNS final remarks

- PyFUNS enables reconfigurable in-network processing in IoT-based WSNs
- PyFUNS simplifies the development of IoT applications
  - PyFUNS abstracts embedded systems complexity and networking aspects
- PyFUNS allows to deploy dynamically distributed services
  - Applications can be moved anytime
- PyFUNS is a framework that can be used to extend WSN lifetime:
  - reducing network traffic and energy consumption

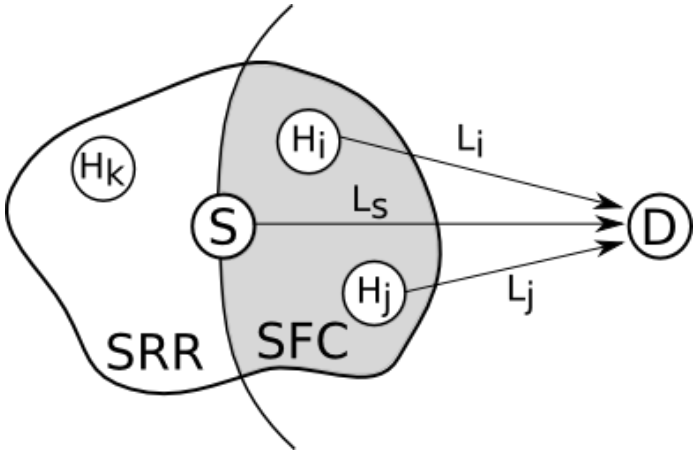
SPEED-6LOWPAN

SPEED Protocol

- Sending packet:
  - Two possible space regions:
    - *Sender Radio Range (SRR)* – contains all the nodes reachable from sender node
    - *Sender Feasible Candidate (SFC)* – contains all the nodes possible next-hop candidates to reach the destination
  - Next-hop is selected

    between the nodes in SFC

    that is nearest to destination

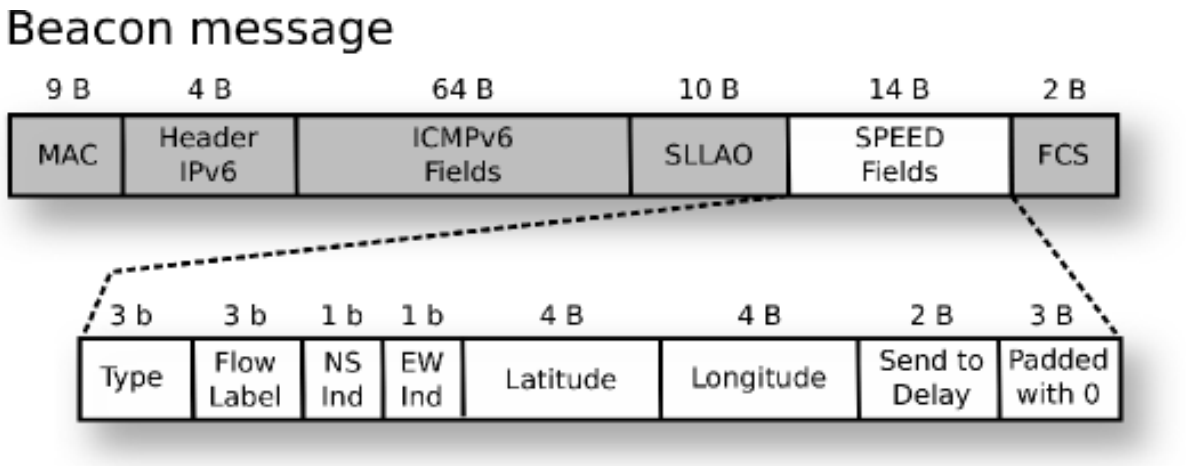


SPEED-6LoWPAN

- The adaptation of SPEED protocol in a 6LoWPAN context require to define:
  - Encapsulation strategies for the protocol messages
  - Optimizations based on the services provided by the ICMPv6 protocol
- Three types of messages:
  - *Beacon message*
  - *Backpressure beacon message*
  - *Data packet message*

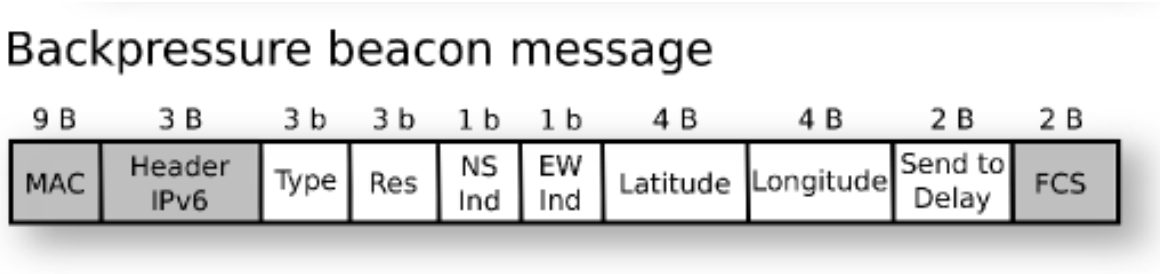
Beacon message

- Periodically broadcasted by each node in order to announce its presence to neighbors
- Is sent through the Routing Advertisement



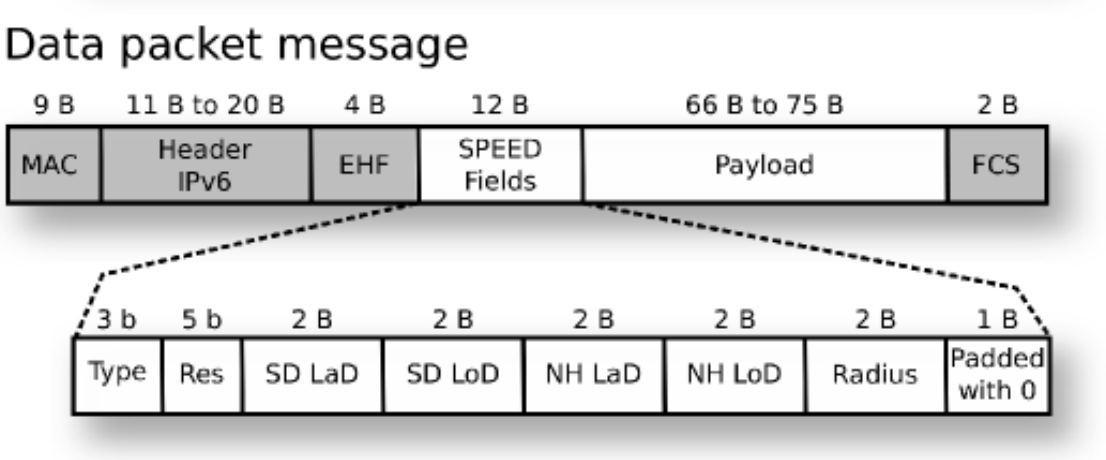
Backpressure beacon message

- Main purpose is to:
  - reduce network congestion
  - solve void avoidance problems
  - It is sent in regular 6LoWPAN message
- If the node can not reach the destination due to unfeasible paths the SendToDelay is set equal to infinity



Data Packet message

- The protocol provides three types of real-time communication services:
  - *Real-time unicast*
  - *Real-time area-multicast*
  - *Real-time area-anycast*

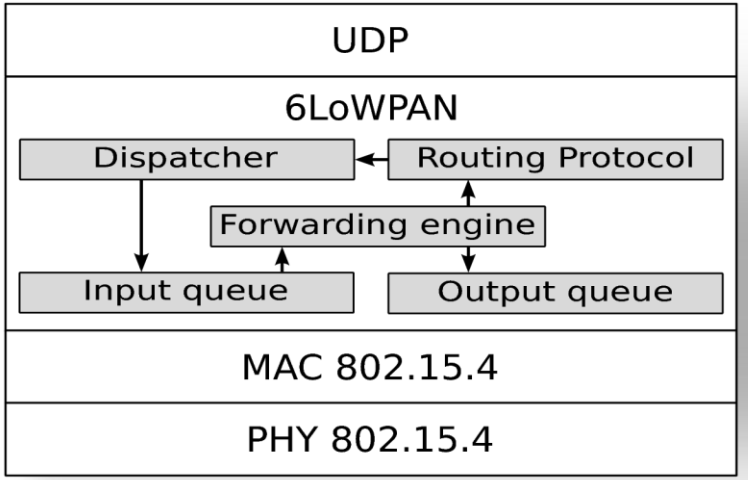


Data Packet Message

- It is sent in regular 6LoWPAN message with additional information
- The additional information must be encapsulated in 6LoWPAN packets adding the Extension Header Fields (EHF)
- Geographic destination of the packet is due by the difference in latitude and longitude between next-hop and destination
- The difference in latitude and in longitude between sender and destination are used for end-to-end communications

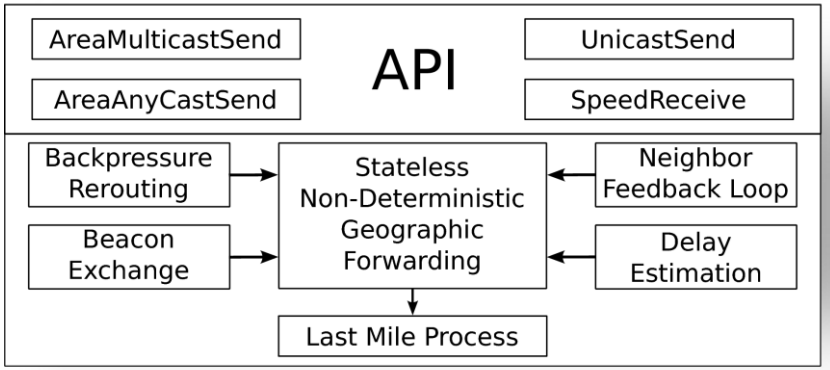
SPEED implementation in Erika OS

- Input and output queue manage incoming and outcoming messages respectively
- Forwarding engine analyzes incoming messages



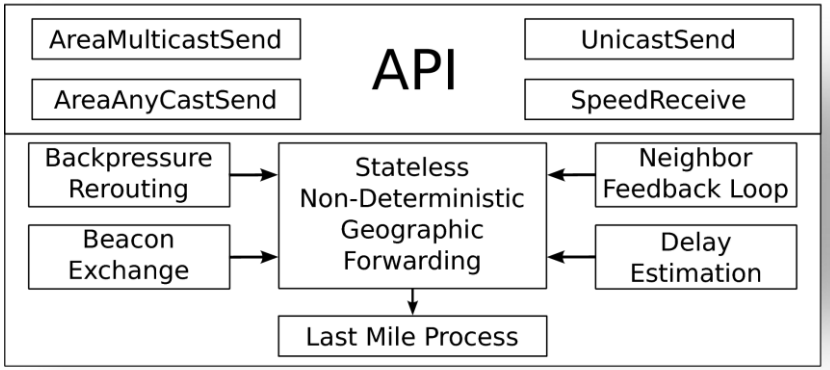
SPEED implementation in Erika OS

- Four APIs:
  - *UnicastSend* to send a packet to a node identified by its IPv6 address
  - *AreaMulticastSend* to send a packet to nodes in an area identified by its center and radius
  - *AreaAnycastSend* to send a packet to at least one node in an area identified by its center and radius
  - *SpeedReceive* to give a node the possibility to accept packets targeted for it



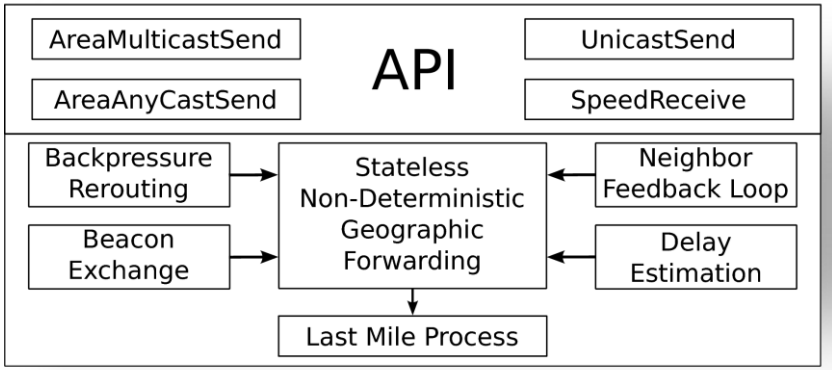
SPEED implementation in Erika OS

- Six modules:
  - *Stateless non-deterministic geographic forwarding*. It is in charge of the next-hop selection
  - *Neighbor beacon exchange*. It sends periodically the beacon messages
  - *Delay estimation*. This Module measure the delay for a one-hop transmission



SPEED implementation in Erika OS

- *Neighborhood feedback loop*. The main goal of this module is to maintain the single hop relay speed
  - *Backpressure rerouting*. Once a network congestion is identified the module start a backpressure rerouting procedure
  - *Last mile process*. This module enters in action only when a packet is in the delivering area



SPEED-6LoWPAN: Performance Evaluation

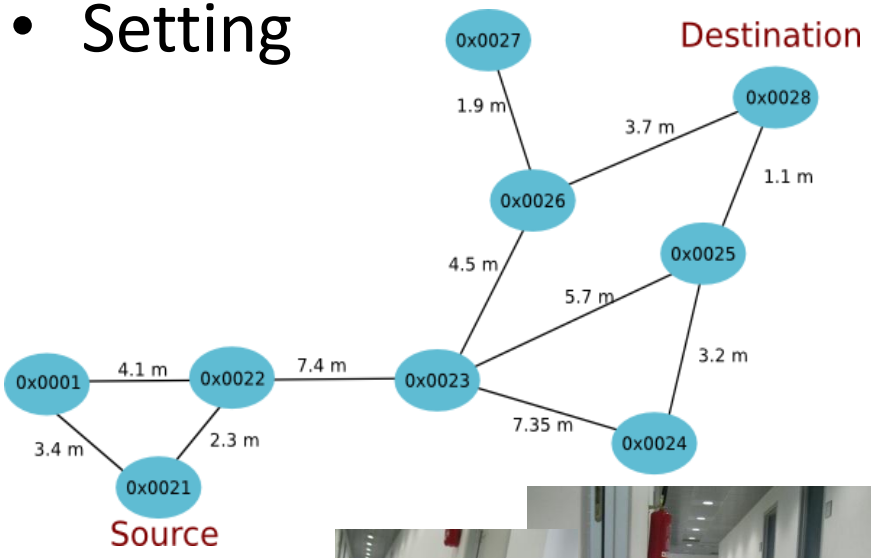
- AODV:
  - Path creation from source to destination
    - Flooding
    - Routing table
- *Path with the minimum number of hop*

- SPEED-6LoWPAN:
  - Use of geographic position for each device
  - No path from source to destination
    - No routing table
    - Next-hop evaluation for each packet
  - *Path with the minimum geographical distance*

SPEED-6LoWPAN: Performance Evaluation

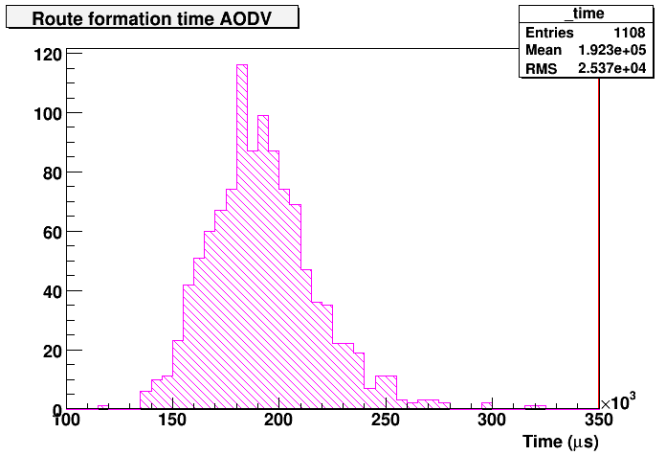
- Evaluated parameters:
  - *Route formation time*
  - *Routing table memory occupancy*
  - *Round trip time*
  - *Path management: load balancing*
  - *Path management: flow shaping*

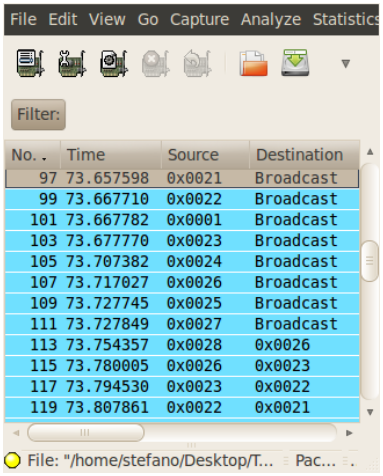
- Setting



Route Formation Time

- AODV:
  - Average: ~ 192 ms
  - Max: ~ 417 ms
  - Min: ~ 117 ms
- SPEED-6LoWPAN:
  - No route formation time required
  - Next-hop evaluation for each packet





Routing Table Memory Occupacy

- AODV
  - Destination IP: 8 bytes
  - Destination Sequence Number: 4 bytes
  - Valid Sequence Number: 1 bit
  - State: 1 byte
  - Hop Count: 1 byte
  - Next Hop: 8 bystes
  - Precursor: 8 bytes

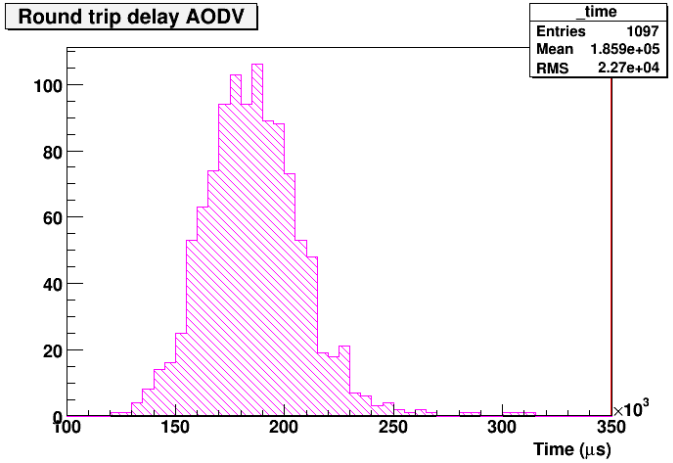
  TOT= ~30 bytes

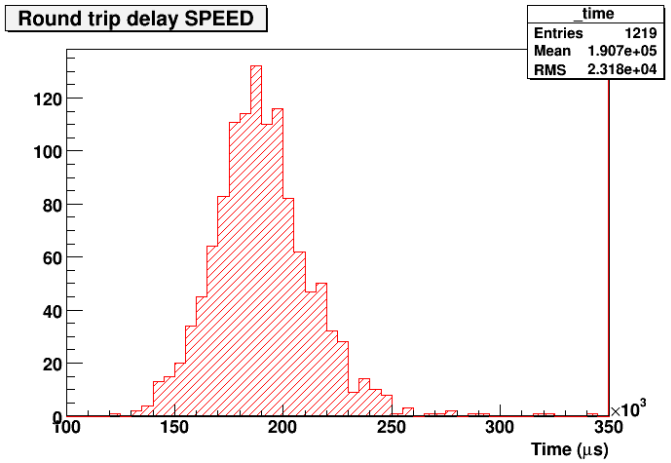
- SPEED-6LoWPAN
  - Latitude: 4 bytes
  - Longitude: 4 bytes
  - NS Indicator: 1 bit
  - EW Indicator: 1 bit
  - Send To Delay: 2 byte
  - Expire Time: 2 bystes

  TOT= ~12 bytes

Round Trip Time

- AODV:
  - Average: ~ 185 ms
  - Max: ~ 473 ms
  - Min: ~ 123 ms
- SPEED-6LoWPAN:
  - Average: ~ 190 ms
  - Max: ~ 341 ms
  - Min: ~ 124 ms

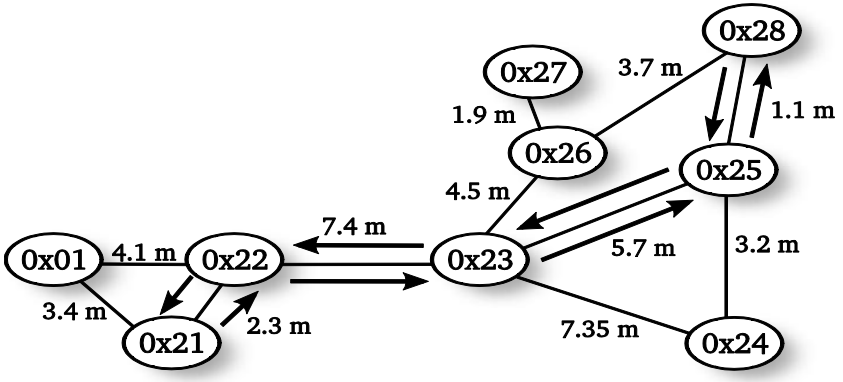




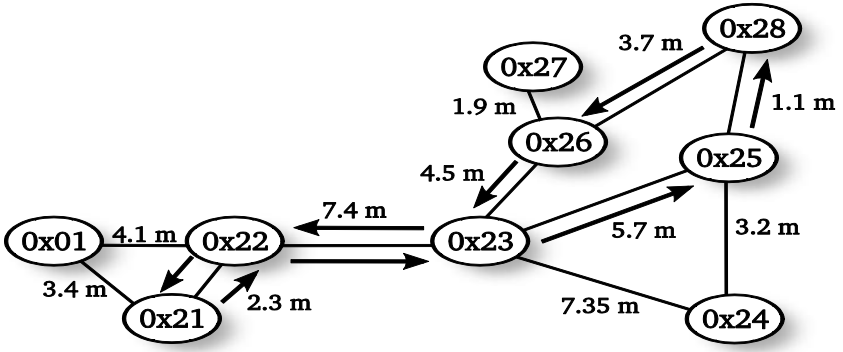
Path Management: Load Balancing

Test: 50 packets from 0x21 to 0x28 and replies

- AODV:
  - N. of packet forward by 0x25: 100
  - N. of packet forward by 0x26: 0



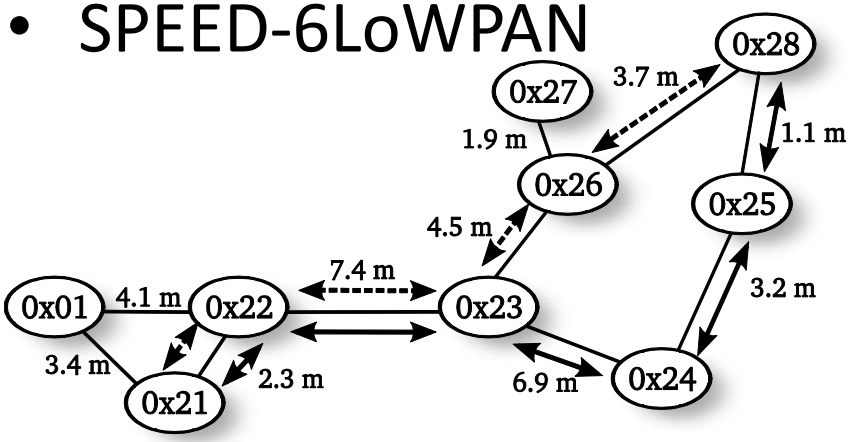
- SPEED-6LoWPAN:
  - N. of packet forward by 0x25: 50
  - N. of packet forward by 0x26: 50

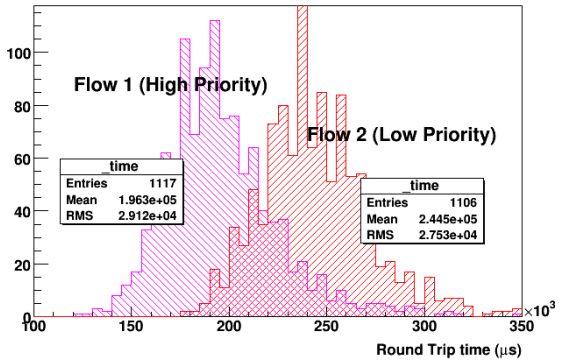


INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

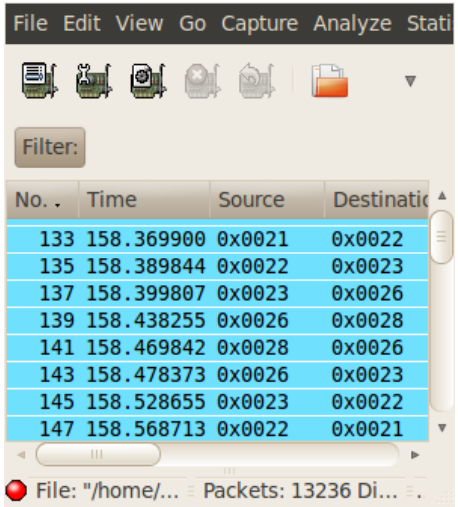
Path Management: Flow Shaping

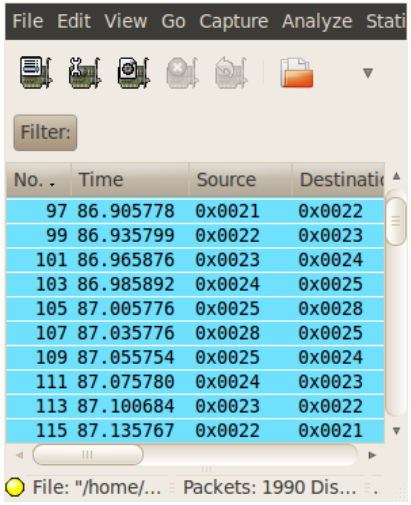
- SPEED-6LoWPAN





- AODV:
  - No possibility to choose high priority path

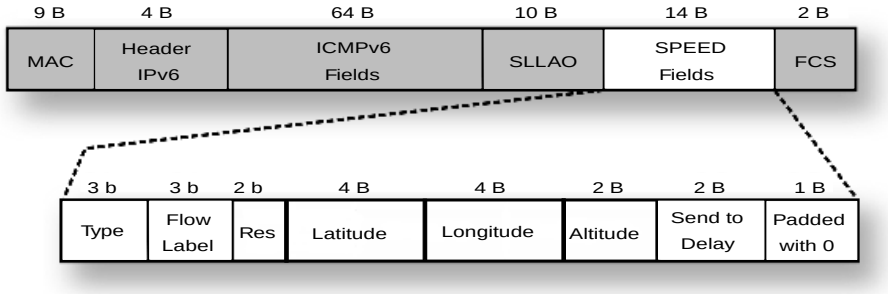




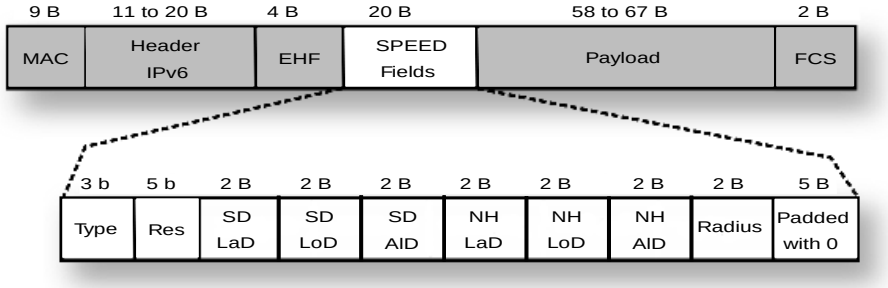
SPEED-3D

SPEED-3D messages

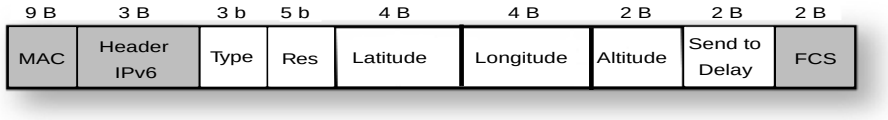
Beacon message



Data packet message



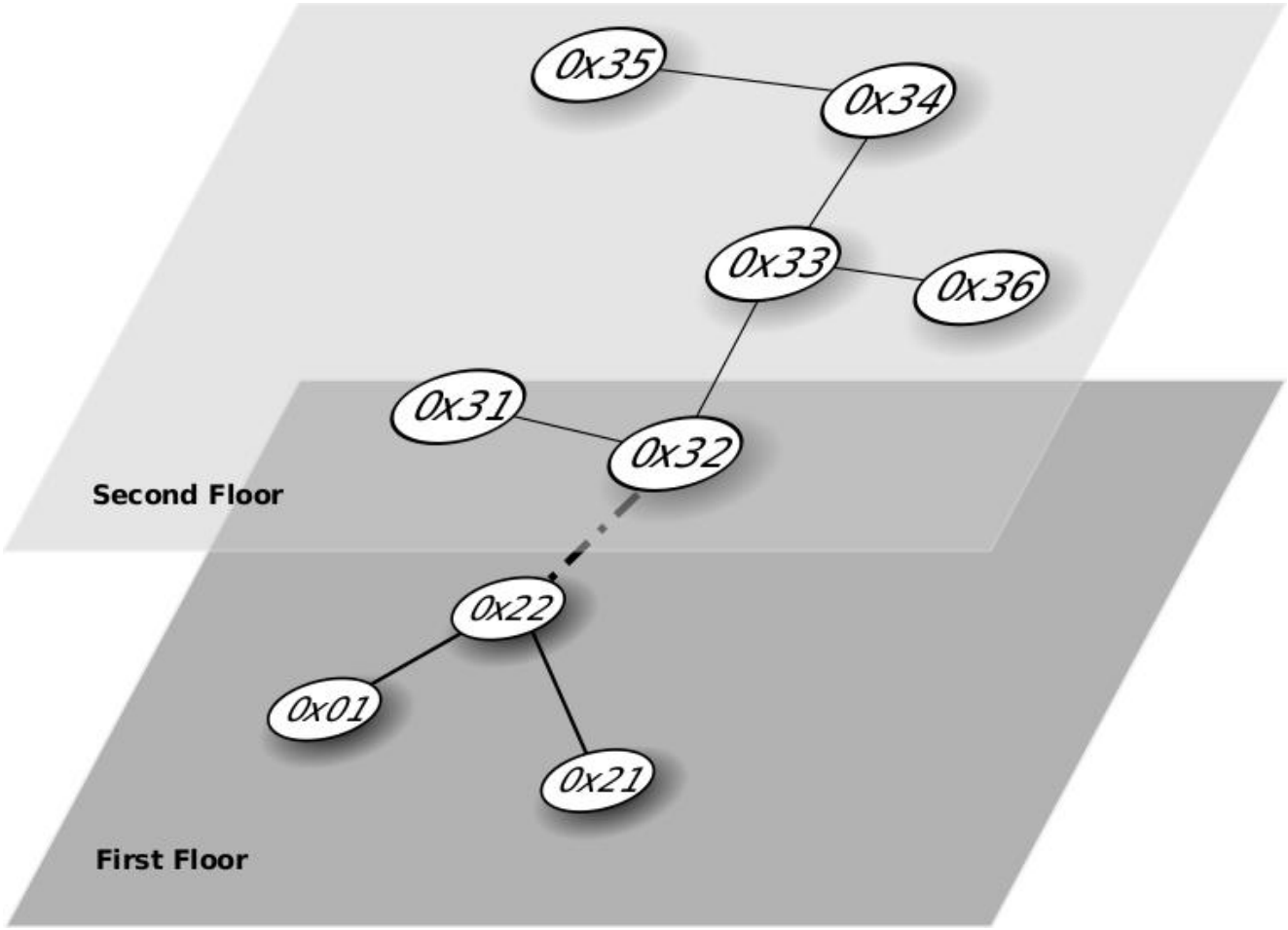
Backpressure beacon message



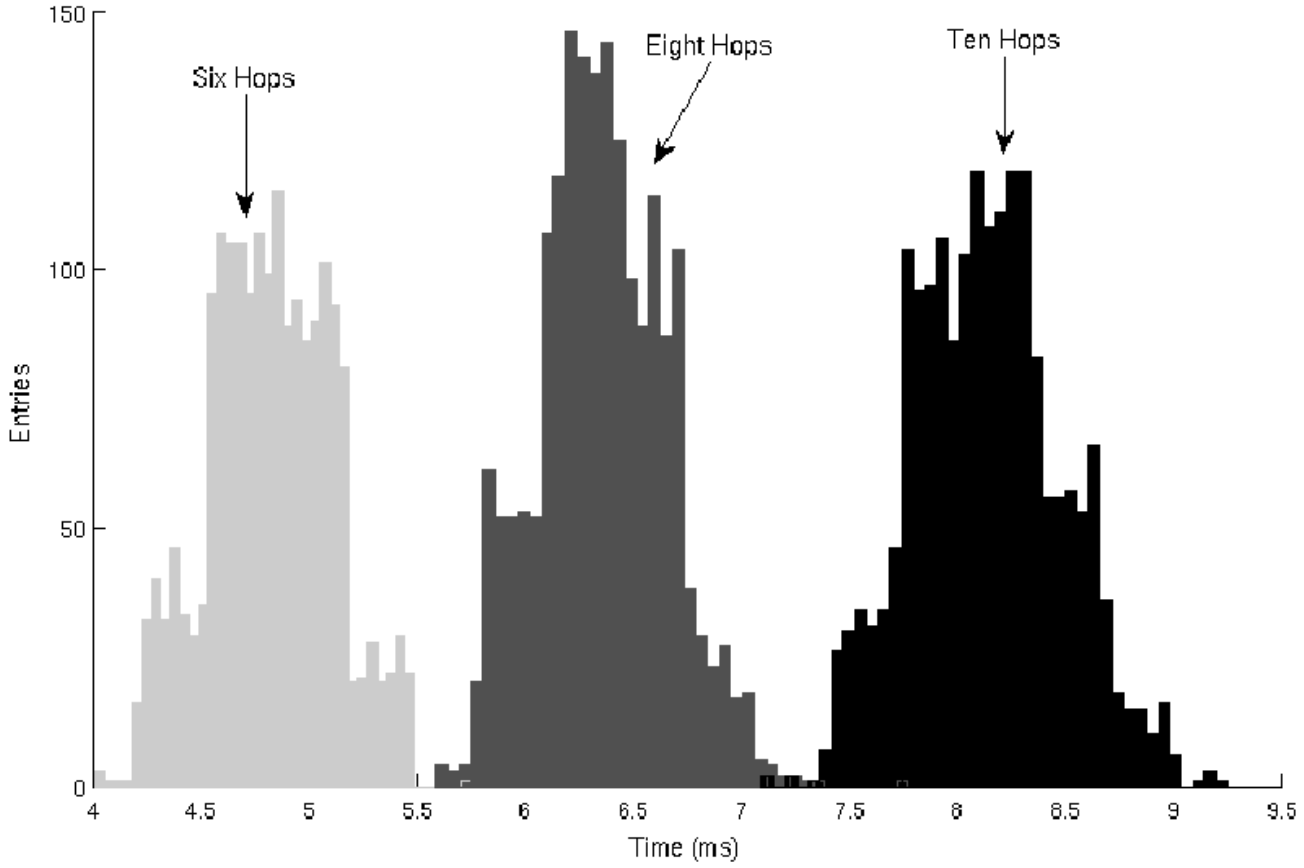
INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

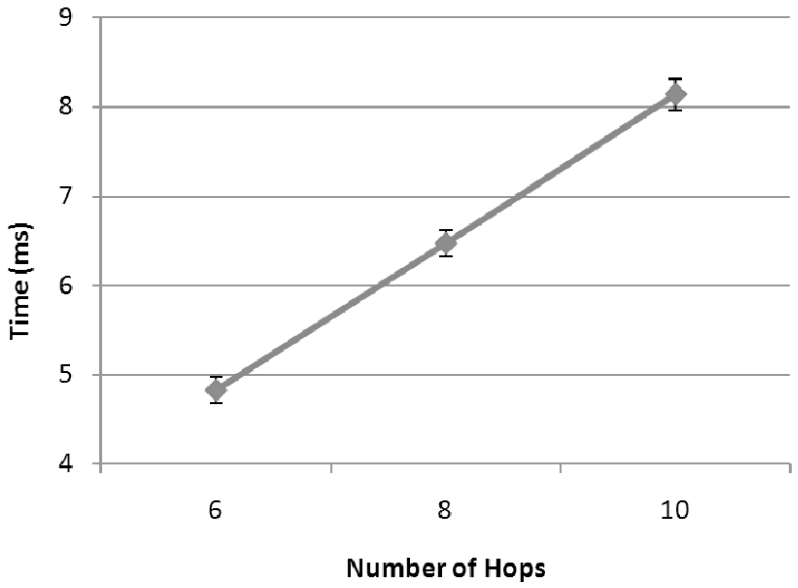
SPEED-3D evaluation setup

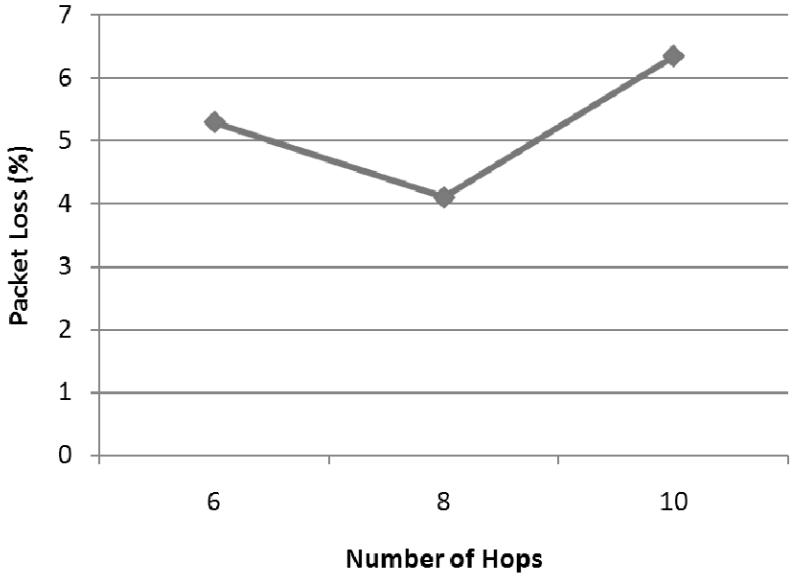


SPEED-3D: hops based RTT

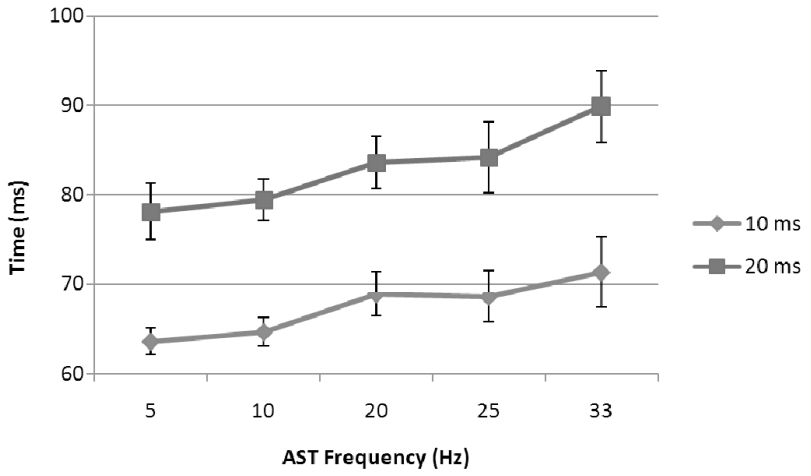


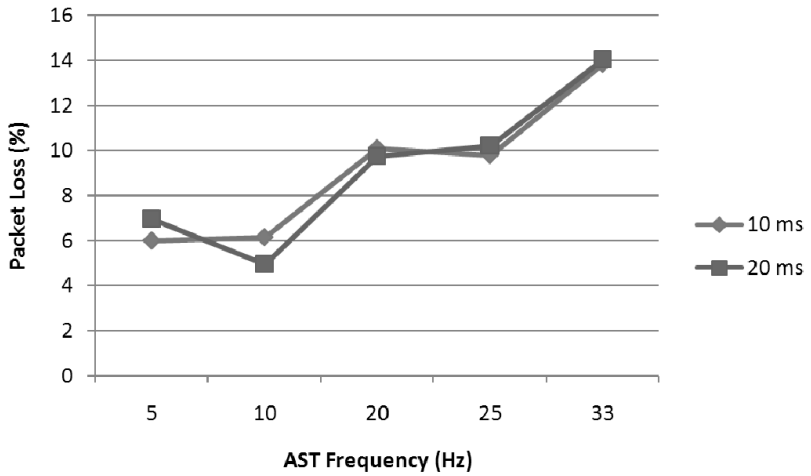
SPEED-3D: RTT & PLR



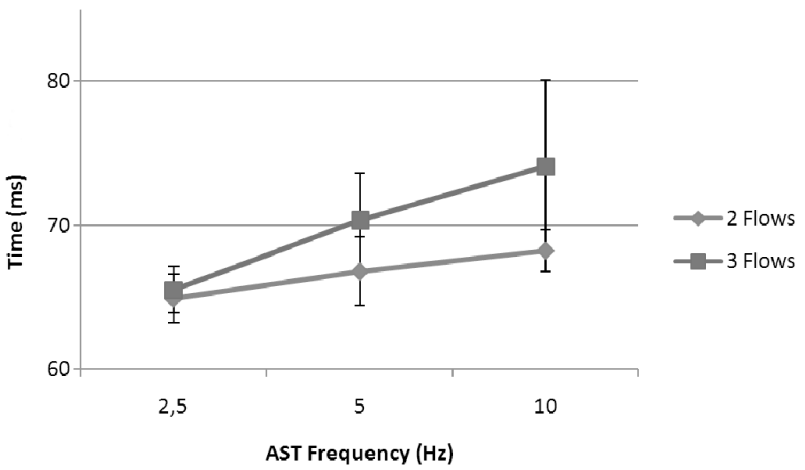


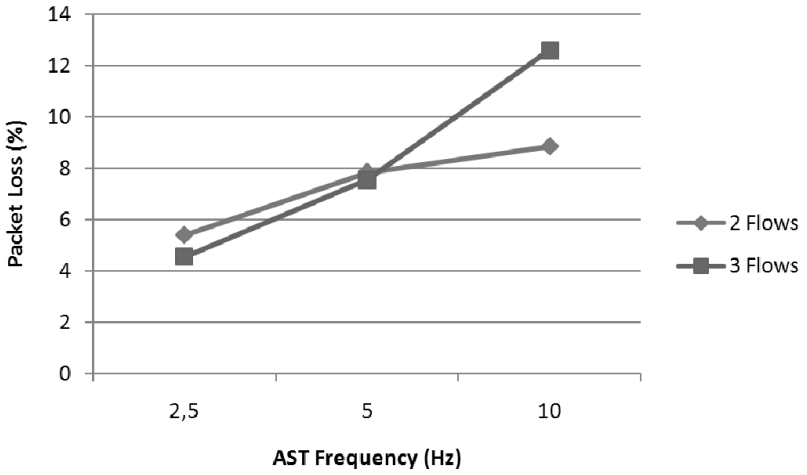
SPEED-3D: RTT & PLR





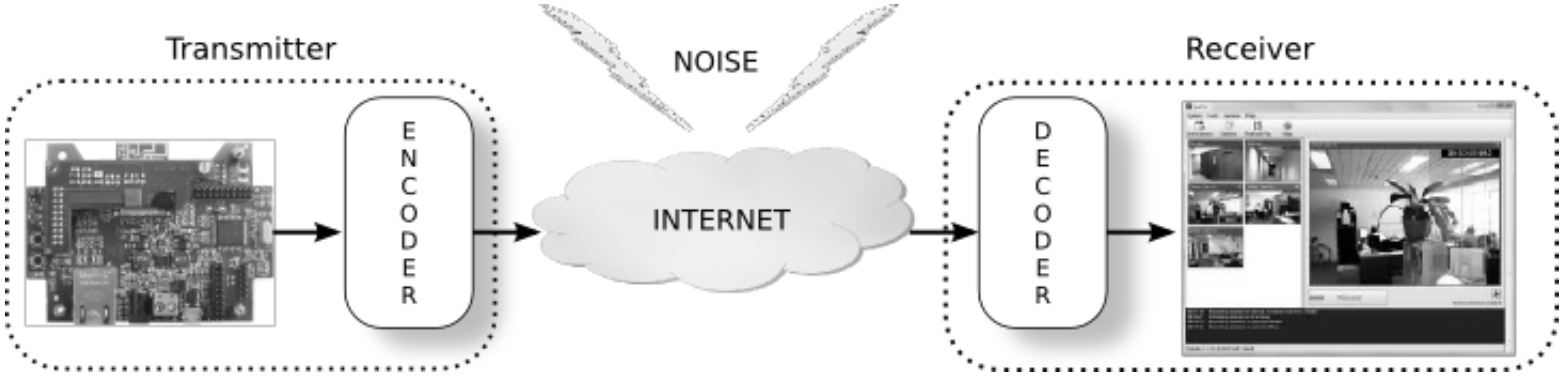
SPEED-3D: RTT & PLR





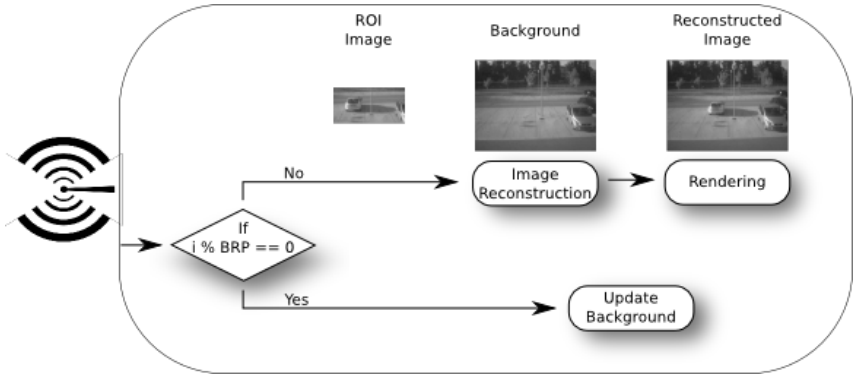
VIDEO STREAMING

CHAIN



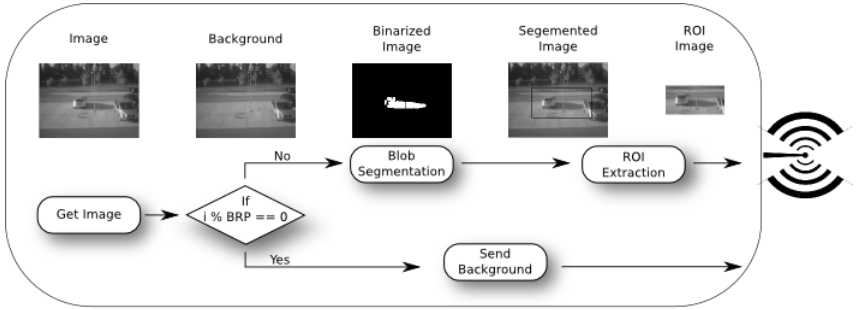
Decoder

```
Frame_id = 0;
while (1) {
   get_image ( image );
   if (( Frame_id % BRP) == 0) {
      send_image ( image );
      bgnd = set_background ( image );
   } else {
      diff = blob_segmentation (image , bgnd );
      roi = roi_extraction ( diff );
      send_roi (roi , header );
   }
   Frame_id ++;
}
```



Encoder

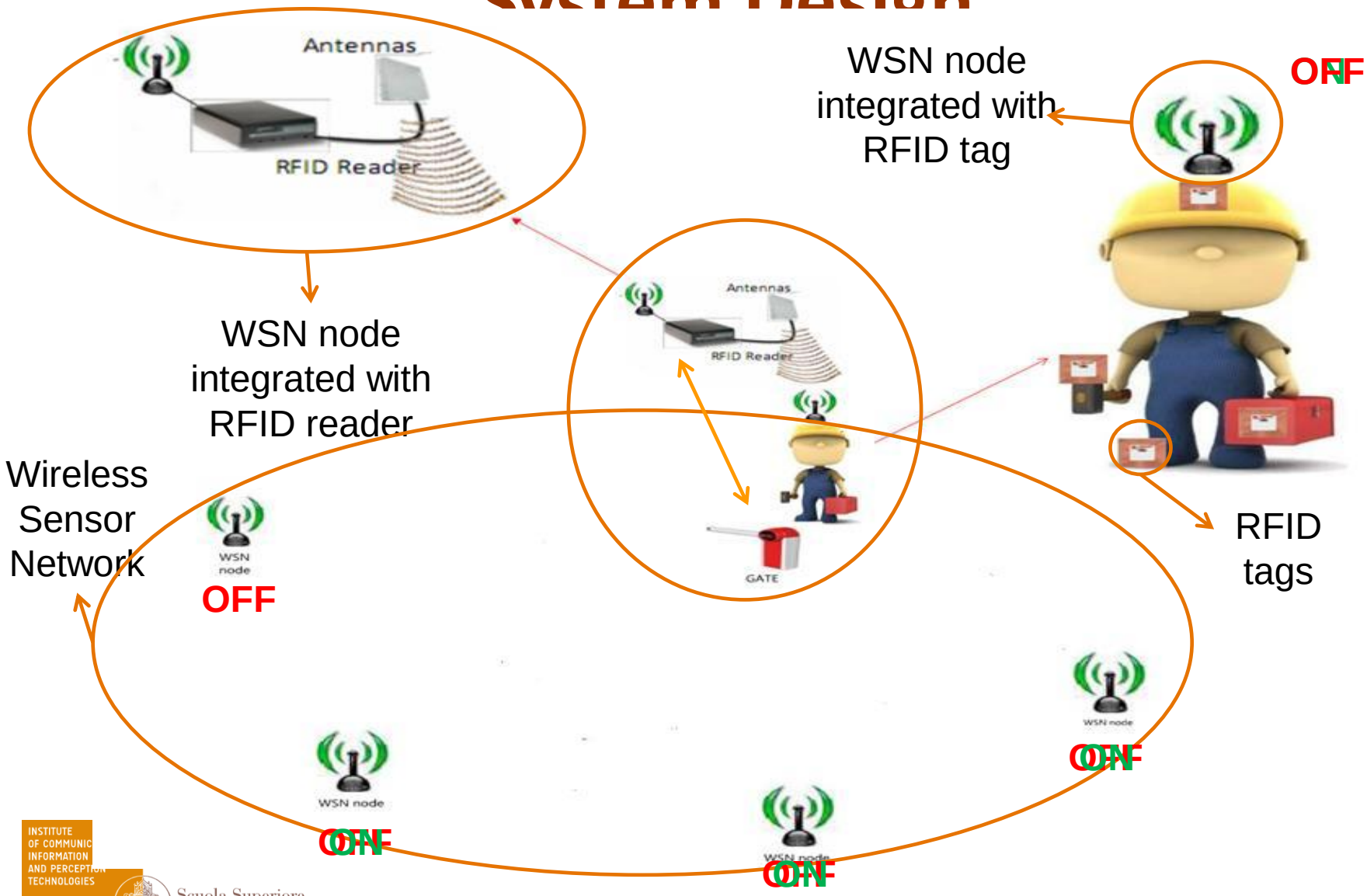
```
Frame_id = 0;
while (1) {
   receive_data ( buffer );
   if (( Frame_id % BRP) == 0) {
      bgnd = set_background ( buffer
);
   } else {
      image = reconstruct_image
(bgnd , buff );
   }
   Frame_id ++;
}
```



Error-resilient packet

RFID

System Design



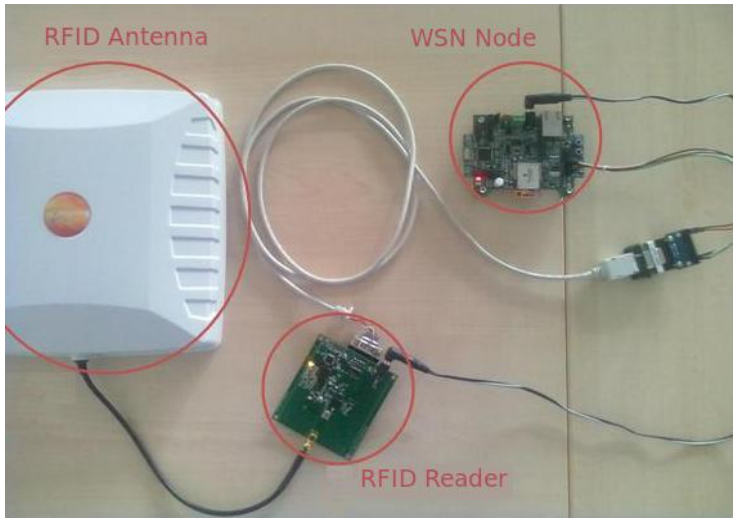
Hardware Components 1/2

- Board Seed-eye:
  - CPU Microchip PIC32, 80 MHz
  - IEEE 802.15.4 transceiver

- Tag RFID:
  - Chip IDS sl900a
  - Serial peripheral interface
  - Compliance with EPC standard

Hardware Components 2/2

- Reader RFID:
  - RF Power: 500 mW (27 dBm)
  - Distance in reading tag memories is close to 5m
  - Compliance with EPC standard

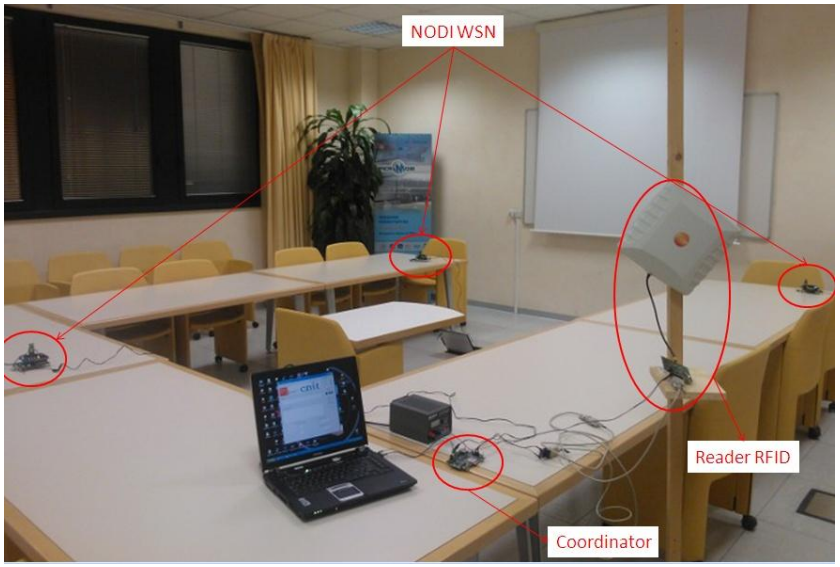


Software Components

- Erika RTOS
- Features developed:
  - Communication with the tag reader (through serial)
  - RFID tags memory access
  - User identification
  - Smart activation
  - Integration with WSN functionality

System validation

- Performance evaluation in a real scenario:
  - Correct accesses versus number of attempts
  - Response time
- Off-line analysis:
  - Power consumption

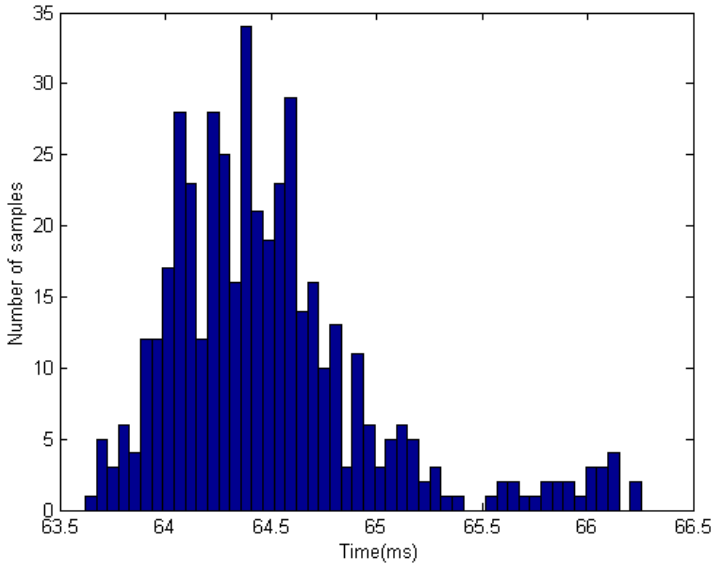


Performance Evaluation

- Correct accesses versus number of attempts

Number of attempts	Correct access [%]
1	89.2
2	97.4
3	99.8
4	100.0

- Response time

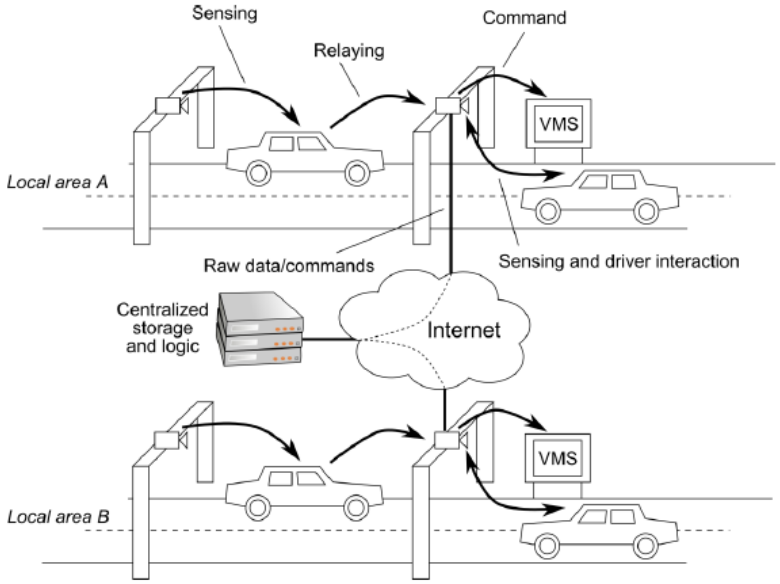


- No missing acceptances for number of attemps greater than 4
- Mean equal to 67 ms (access is authorized after the first attempt)

ITS USE CASE

Typical ITS before ICSI

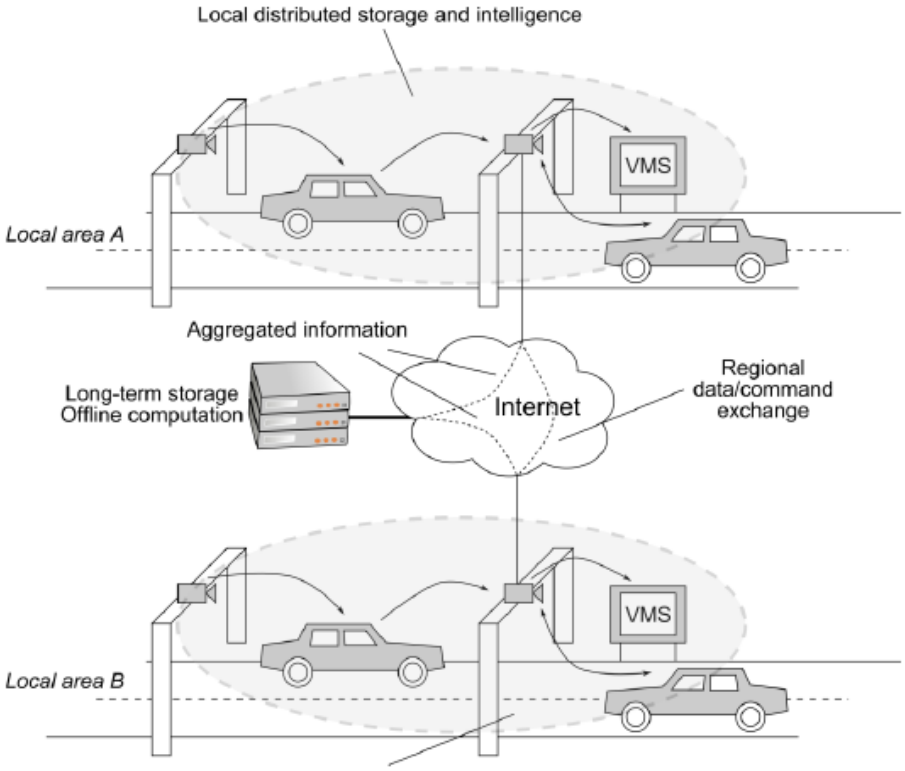
- The current ITS (Intelligent Transportation Systems) approach is purely hierarchical
  - Sensed data flow from the leaves (road-side or vehicle sensors) to the root (traffic management centre)
  - Not scalable nor flexible
  - Made of several sub-systems working in isolation to provide dedicated functions (often closed)
  - Fusion of data provided by different sub-systems operating in the same area is very difficult in practice



INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

ICSI Project

- ICSI proposes a new architecture for ITS
  - Sensing and actuation distributed over Gateways
  - Faster and more accurate reaction



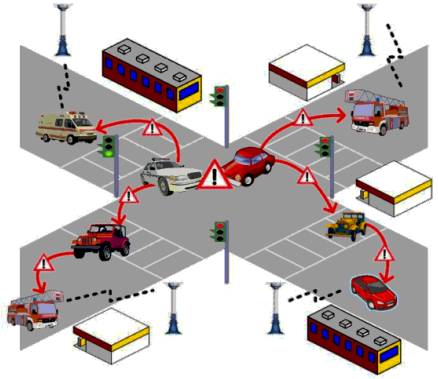
- **Main Project Goals**
  - Provide accurate real-time traffic monitoring
  - Reduce car accidents
  - Reduce travel time
  - Enhance context-aware personal services

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

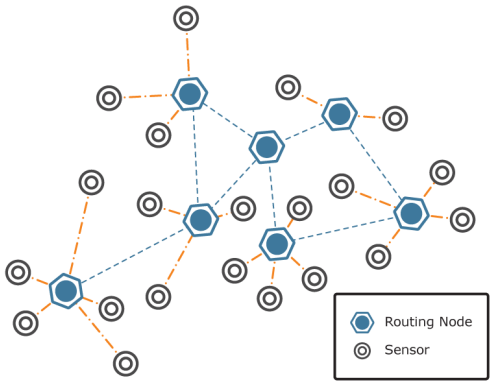
ICSI key technologies

- Vehicular ad hoc network – VANET
  - Vehicular communication
  - Road Side Unit communication
  - IEEE 802.11p



- Computer vision
  - Traffic flow analysis
  - Traffic and parking monitoring

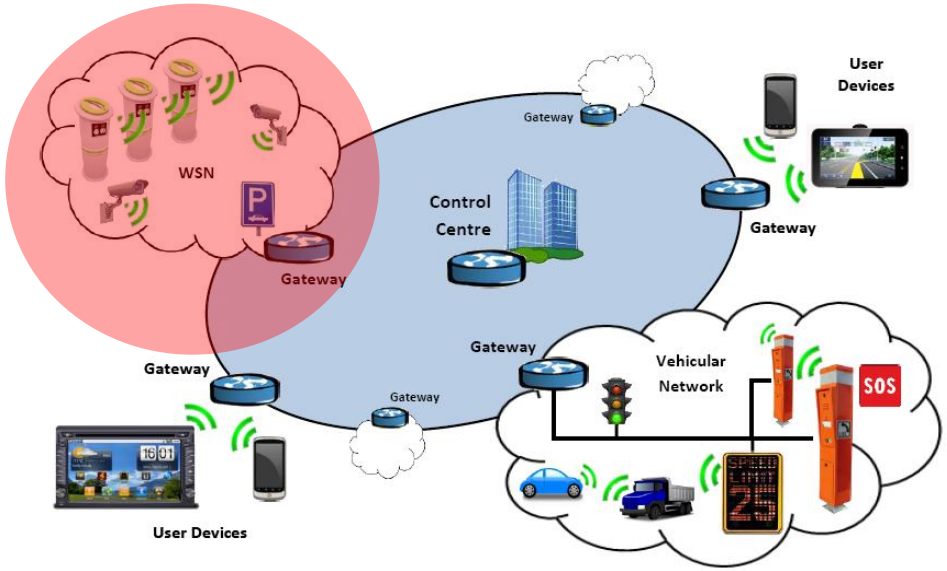
- Visual Wireless Sensor Network
  - Pervasive communication capabilities
  - Scalability to large scale of deployment
  - Heterogeneity of nodes
  - Multimedia sensors (video)



INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

ICSI Event-based Middleware 1/2

- ICSI ITS system is composed of distributed WSN segments
- The WSN collects and aggregates ITS events
- Traffic flows / Parking slots monitoring using Smart Camera sensors



ICSI event-based middleware to manage data and resources in the pervasive WSN

ICSI Event-based Middleware 2/2

Main features:

1. Event composition, subscription, and notification of simple or composed events
2. Remote software update and configuration
3. Data transmission strategies for monitoring applications

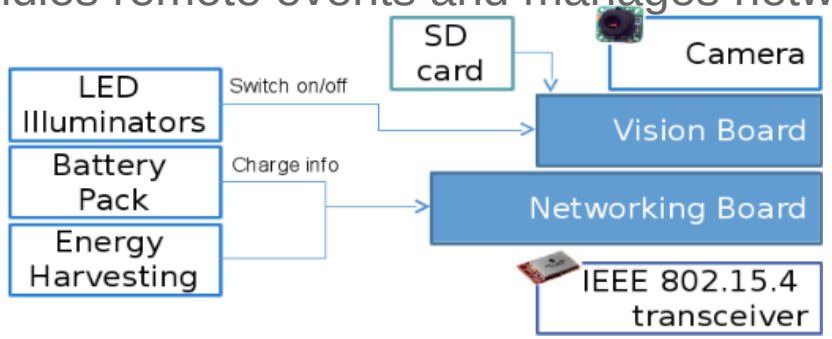
Parking monitoring use-case

- Multiple sensors watching the same slots
- Different confidence on the measure
- In-network aggregation

Structure of the ICSI Middleware 1/4

The ICSI visual sensor node decouple the vision and networking capabilities

- Vision platform
  - Acquires and processes images
  - Generates events for the Networking platform
- Networking platform
  - Connected with the vision board through an intern bus
  - Handles remote events and manages networking functionality



The middleware is designed to be run on the networking board, by supporting the vision board requirements

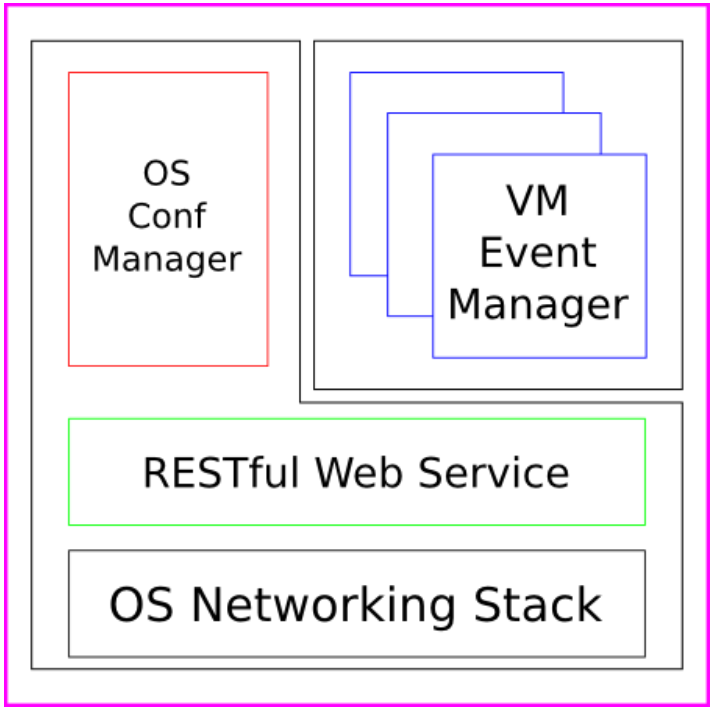
Structure of the ICSI Middleware 2/4

The operated system stack proposed for the Networking platform is composed by:

- An **Operating System** for networking and configuration services
- A **Virtual Machine** for re-configurable event managers
  - Instantiate a controlled execution environments for tasks and easily manage re-programmability
- Publish/Subscribe model instantiated atop a Virtual Machine Based design

Structure of the ICSI Middleware 3/4

The Application logic is kept simple by supporting a precise:

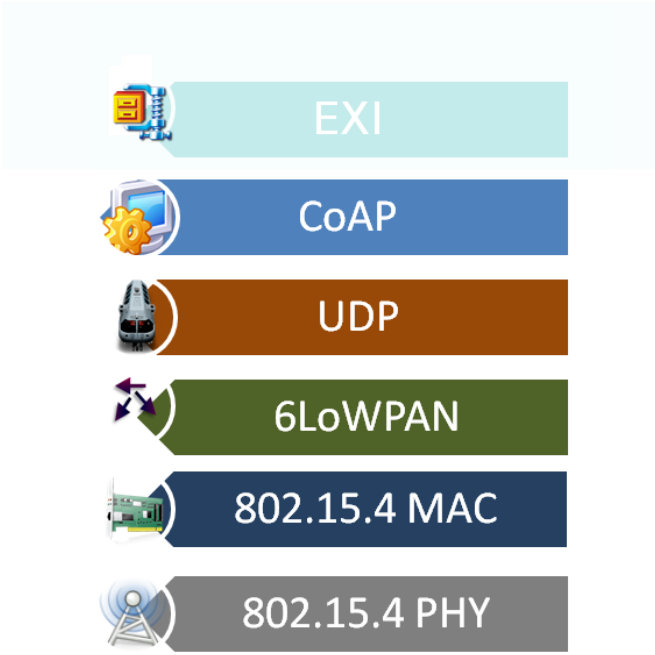


- RESTful Web Service
  - Handles network data inputs and outputs
  - Acts as resource directory service
- Event Managers (multiple)
  - Collect events and perform event processing
  - Take inputs from both vision platform (local events) and network messages (remote events)
- Configuration Manager
  - Handles software/firmware updates, energy policies, network and vision platform parameters configuration

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna

Structure of the ICSI Middleware 4/4

The network logic is build atop of a set of protocols specifically thought for constrained devices



- Efficient XML interchange – EXI
  - Binary XML format
    - Encodes XML documents in a binary data format
    - Reduces the verbosity of XML documents
    - Reduces the cost of parsing
  - Designed to support high-performance XML applications for resource constrained environments (WSNs)
  - To fully support the ETSI M2M paradigm

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES
Scuola Superiore Sant'Anna