An Introduction to the Operating Systems of the IoT

Elahe Jalalpoor and Parham Alvani
Amirkabir University of Technology

elahejalalpoor@gmail.com
parham.alvani@gmail.com

Jun 29, 2015

Outline

- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

Outline

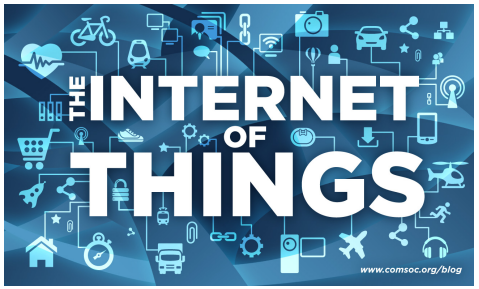
- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

Introduction

- Introduction
  - What is the IoT

What is IoT...

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.



What is the IoT

- A thing, in the Internet of Things, can be any natural or man-made object can be assigned IP address.
- So far, the Internet of Things has been most closely associated with machine-to-machine (M2M) communication.
- Although the concept wasn't named until 1999, the Internet of Things has been in development for decades.

Outline

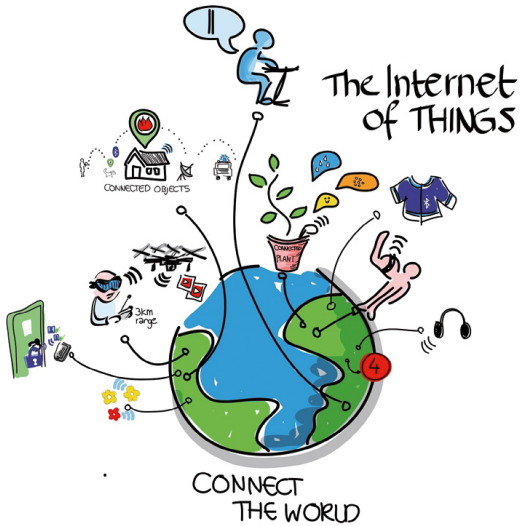
- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

IoT Requirements & Challenges

- IoT Requirements & Challenges
  - Effect of the requirements on OS

IoT Requirements & Challenges

- Scalability
- Modularity
- Connectivity
- Reliability
- Real-time behavior
- Run on a wide spectrum of hardware
- Holistic Security



Modularity

- IoT device will require a modular operating system that separates the core kernel from middleware, protocols, and applications.

Outline

- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

IoT OS

- IoT OS
  - General OS vs IoT OS
  - What are the main requirements in IoT OS
  - What are the main components in IoT OS

Outline

- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

Existing OSs

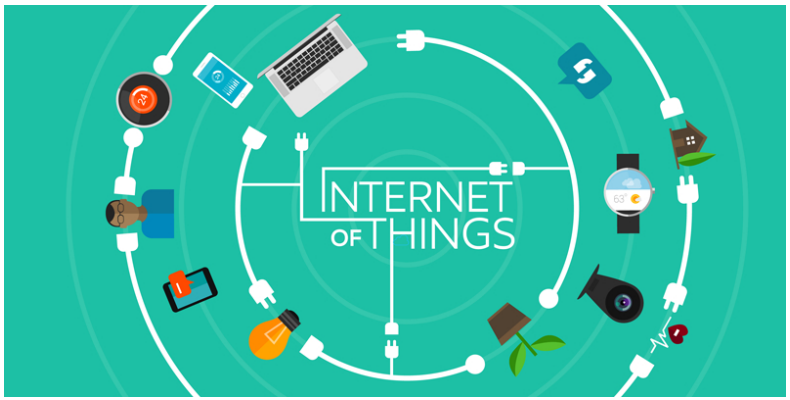
- Existing OSs
  - OS Classification
  - Overview of Open Source OSs
  - Overview of Closed Source OSs
  - Why Not Linux?

OS Classification

- Design Aspects for an IoT OS
  - Monolithic fashion
  - Layered approach
  - Microkernel architecture

Overview of Open Source OSs

- FreeRTOS
- RIOT
- Contiki
- TinyOS
- Embedded Linux
- OpenWSN



FreeRTOS

- FreeRTOS is designed to be small and simple.
- The kernel itself consists of only three or four C files.
- It provides methods for multiple threads or tasks, mutexes, semaphores and software timers.
- Key features are very small memory footprint, low overhead, and very fast execution.

RIOT

- RIOT is a real-time multi-threading operating system.
- RIOT implements a microkernel architecture
- RIOT is based on design objectives including:
  - Energy-Efficiency
  - Reliability
  - Real-Time Capabilities
  - Small Memory Footprint
  - Modularity
  - Uniform API Access
    independent of the underlying hardware
    (this API offers partial POSIX compliance)

Contiki

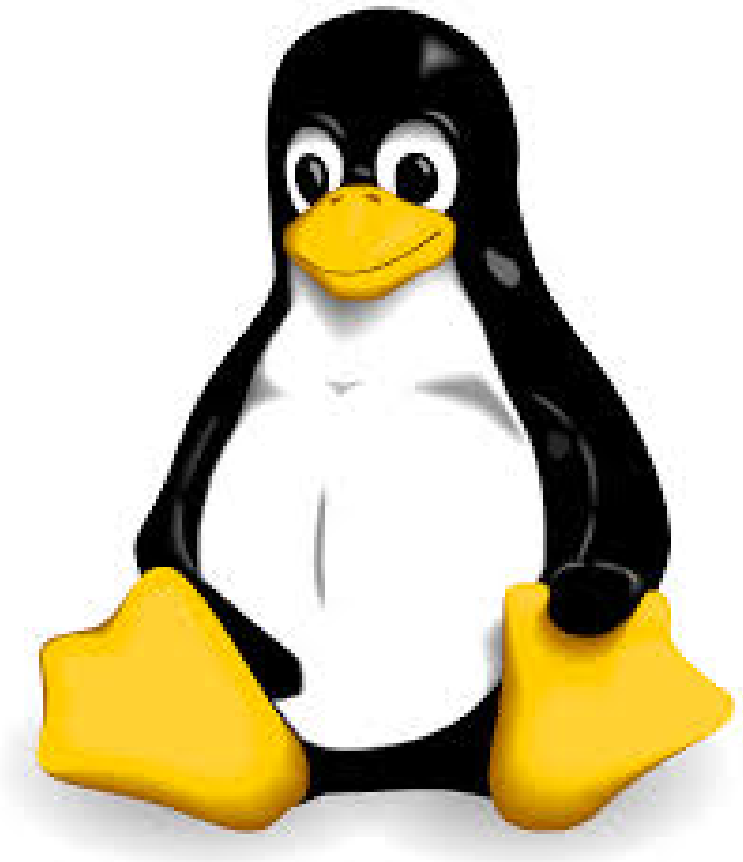
- Contiki is an open source operating system for networked, memory-constrained systems
- Contiki provides three network mechanisms:
  - The uIP stack, which provides IPv4 networking,
  - The uIPv6 stack, which provides IPv6 networking,
  - The Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks.

TinyOS

- TinyOS is a component-based operating system and platform targeting wireless sensor networks.
- TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes.

Embedded Linux

- Embedded Linux is created using OpenEmbedded, the build framework for embedded Linux.
- OpenEmbedded offers a best-in-class cross-compile environment.

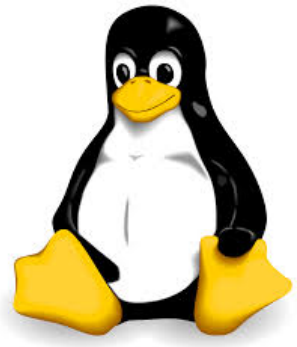


OpenWSN

- The goal of the OpenWSN project is to provide open-source implementations of a complete protocol stack based on Internet of Things standards, on a variety of software and hardware platforms.

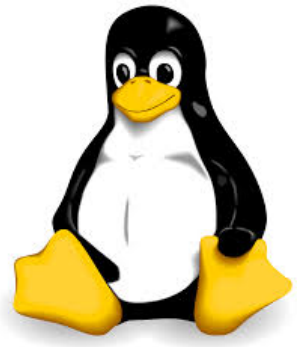
Comparison

OS	Min RAM	Min ROM	C Support	C++ Support
Contiki	$< 2kB$	$< 30kB$	Partial support	No support
Tiny OS	$< 1kB$	$< 4kB$	No support	No support
Linux	$\sim 1MB$	$\sim 1MB$	Full support	Full support
RIOT	$\sim 1.5kB$	$\sim 5kB$	Full support	Full support



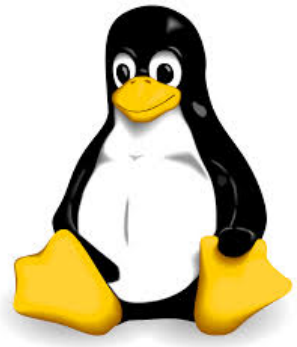
Comparison

OS	Multi-Threading	Modularity	Real-Time
Contiki	Partial support	Partial support	Partial support
Tiny OS	Partial support	No support	No support
Linux	Full support	Partial support	Partial support
RIOT	Full support	Full support	Full support



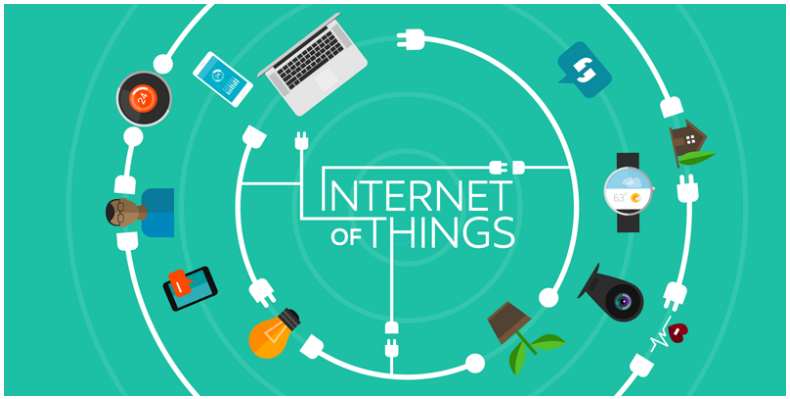
Operating Systems Availability

OS	Wsn430 Node	M3 Node	A8 Node
Contiki	Full support	Full support	No support
Tiny OS	Full support	No support	No support
Linux	No support	No support	Full support
RIOT	Full support	Full support	No support



Overview of Closed Source OSs

- ARM mbed
- Huawei LiteOS
- Google Brillo

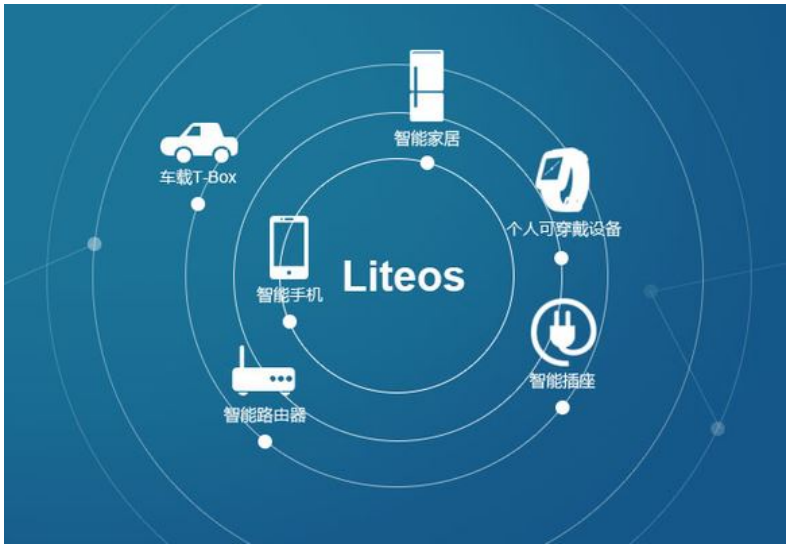


ARM mbed

- Automation of power management
- Software asset protection and secure firmware updates for device security & management
- Connectivity protocol stack support for Bluetooth low energy, Cellular, Ethernet, Wi-fi, Zigbee IP, Zigbee NAN, 6LoWPAN

Huawei LiteOS

- The company says that its LiteOS is the lightest software of its kind and can be used to power a range of smart devices



Google Brillo

- Brillo is derived from Android but polished to just the lower levels.
- It supports Wi-Fi, Bluetooth Low Energy, and other Android things.

Why Not Linux?

Real-Time Linux

Controlling a laser with Linux is crazy, but everyone in this room is crazy in his own way. So if you want to use Linux to control an industrial welding laser, I have no problem with your using PREEMPT_RT.

- Linus Torvalds

Why Not Linux?

- Linux certainly is a robust, developer-friendly OS
- Linux has a disadvantage when compared to a real-time operating system:
  - Memory footprint
  - It simply will not run on 8 or 16-bit MCUs
- Linux will certainly have many uses in embedded devices, particularly ones that provide graphically rich user interfaces.
- There are thousands of applications for which Linux is ill suited.

Outline

- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

Protocol stack

- Protocol stack
  - Traditional Stack
  - IoT Requirements
  - IoT Stack
  - Comparison

Protocol stack

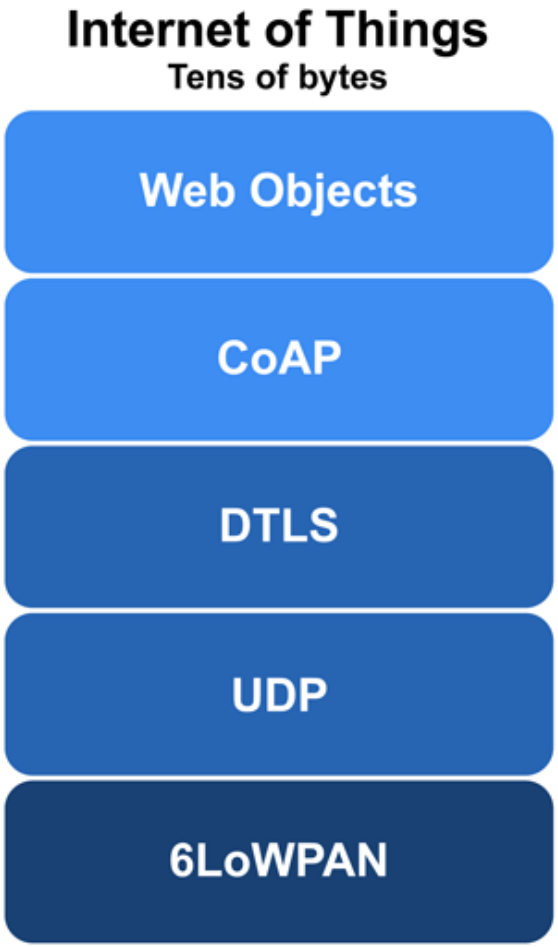
- Can you build an IoT system with familiar Web technologies?

Protocol stack

- Can you build an IoT system with familiar Web technologies?
- Yes you can, although the result would not be as efficient as with the newer protocols.

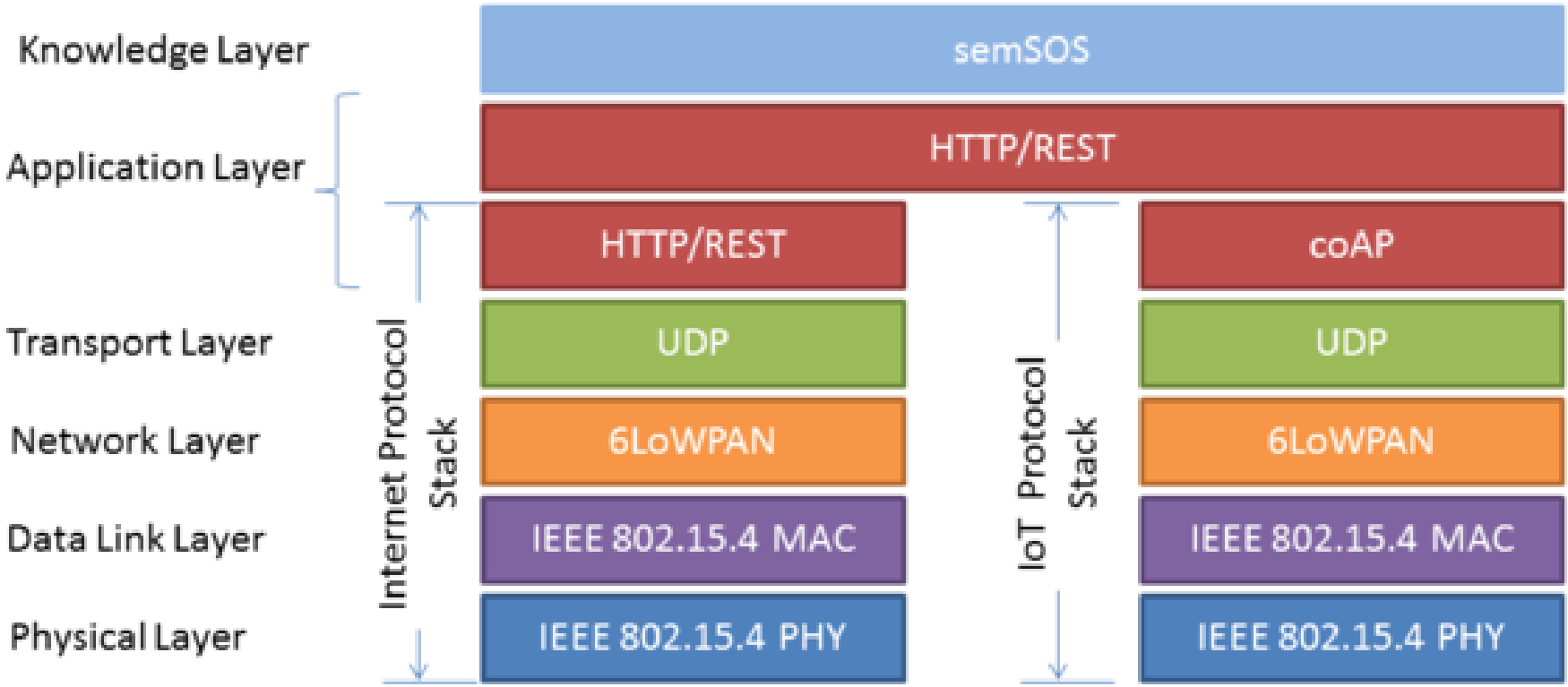
Comparison

- Inefficient content encoding
- Huge overhead, difficult parsing
- Requires full Internet devices



- Efficient objects
- Efficient Web
- Optimized IP access

Comparison


Knowledge Layer
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
semSOS
HTTP/REST
HTTP/REST
UDP
6LoWPAN
IEEE 802.15.4 MAC
IEEE 802.15.4 PHY
Internet Protocol Stack
coAP
UDP
6LoWPAN
IEEE 802.15.4 MAC
IEEE 802.15.4 PHY
IoT Protocol Stack

Internet Usage and Protocols for the IoT

Protocol	*Transport*	*Messaging*	*2G,3G,4G (1000's)*	*LowPower and Lossy (1000's)*	*Compute Resources*	*Security*	*Success Stories*	*Arch*
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility field area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High- Optional	Military	Bus
DPWS	TCP		Good	Fair	100Ks/RAM Flash ++	High- Optional	Web Servers	Client Server
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low- Optional	Smart Energy Phase 2	Client Server
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High- Optional	Network Monitoring	Client-Server
UPnP		Pub/Subscrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High- Manditory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subscrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High- Optional	CERN	P2P

Outline

- Introduction
- IoT Requirements & Challenges
- IoT OS
- Existing OSs
- Protocol stack
- Test & Development Environments
- Conclusion

Test & Development Environments

- Test & Development Environments
  - IoT Lab test
  - RIOT environment
  - Compilers
  - Development environment

IoT Lab test

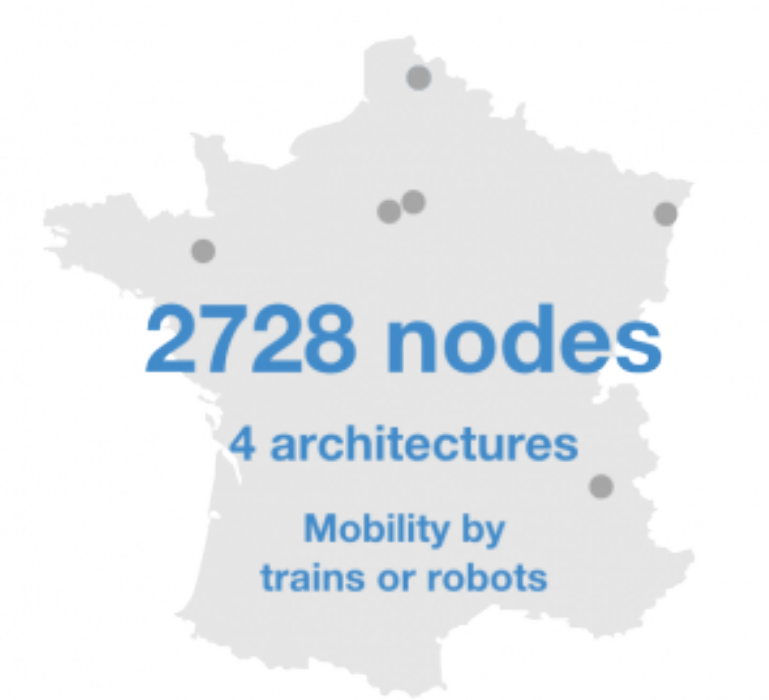
- A scientific testbed
- Different topologies and environments
- Different nodes
- A part of FIT

A scientific testbed

- IoT-LAB provides full control of network nodes and direct access to the gateways to which nodes are connected, allowing researchers to monitor nodes energy consumption and network-related metrics.

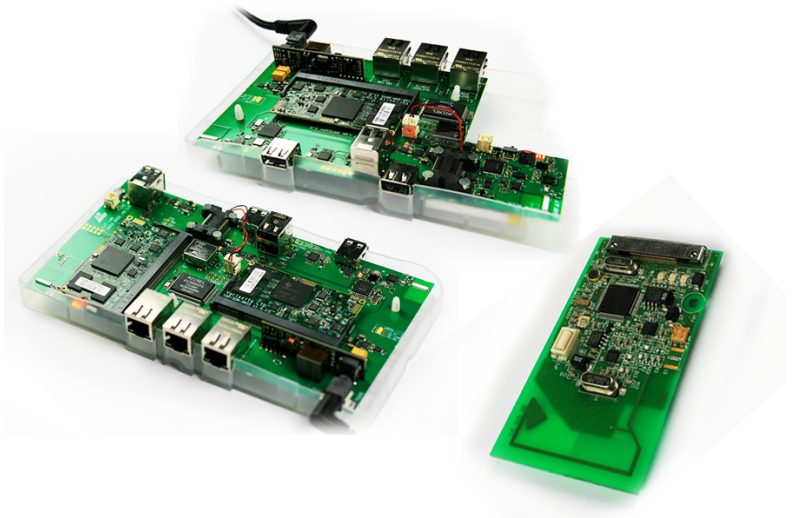
Different topologies and environments

- IoT-LAB testbeds are located at six different sites across France which gives forward access to 2728 wireless sensors nodes.



Different nodes

- The IoT-LAB hardware infrastructure consists of a set of IoT-LAB nodes.
- A global networking backbone provides power and connectivity to all IoT-LAB nodes and guaranties the out of band signal network needed for command purposes and monitoring feedback.



A part of FIT

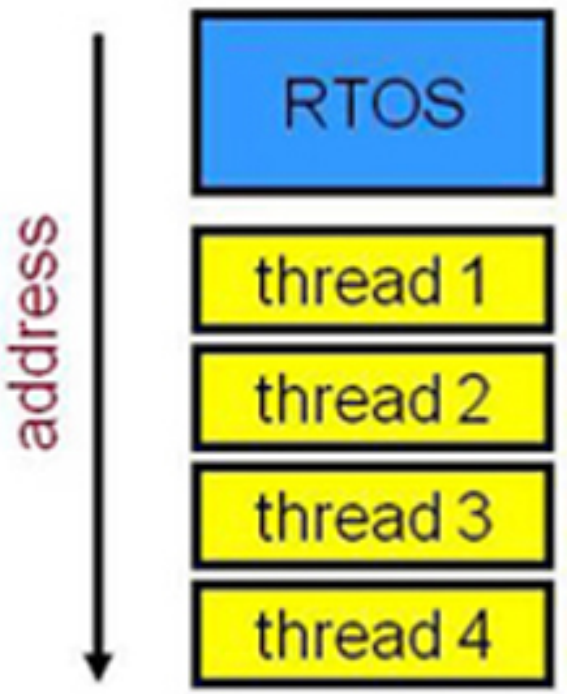
- IoT-LAB is a part of the FIT (Future Internet of the Things) platform.
- FIT is a set of complementary components that enable experimentation on innovative services for academic and industrial users.

RIOT environment

- RIOT features the native port with networking support.
- This allows you to run any RIOT application on your Linux or Mac computer and setup a virtual connection between these processes.

Multi-Tasking, Thread Model

- Most RTOS products on the market are thread model.
- Tasks are now called threads.
- All the tasks code and data occupy the same address space, along with that of the RTOS itself.



Event-Driven, Non-Blocking I/O Model

- Networking Event-Driven
- Non-Blocking I/O

Questions?