RIOT
an Open Source Operating System for Low-end IoT Devices

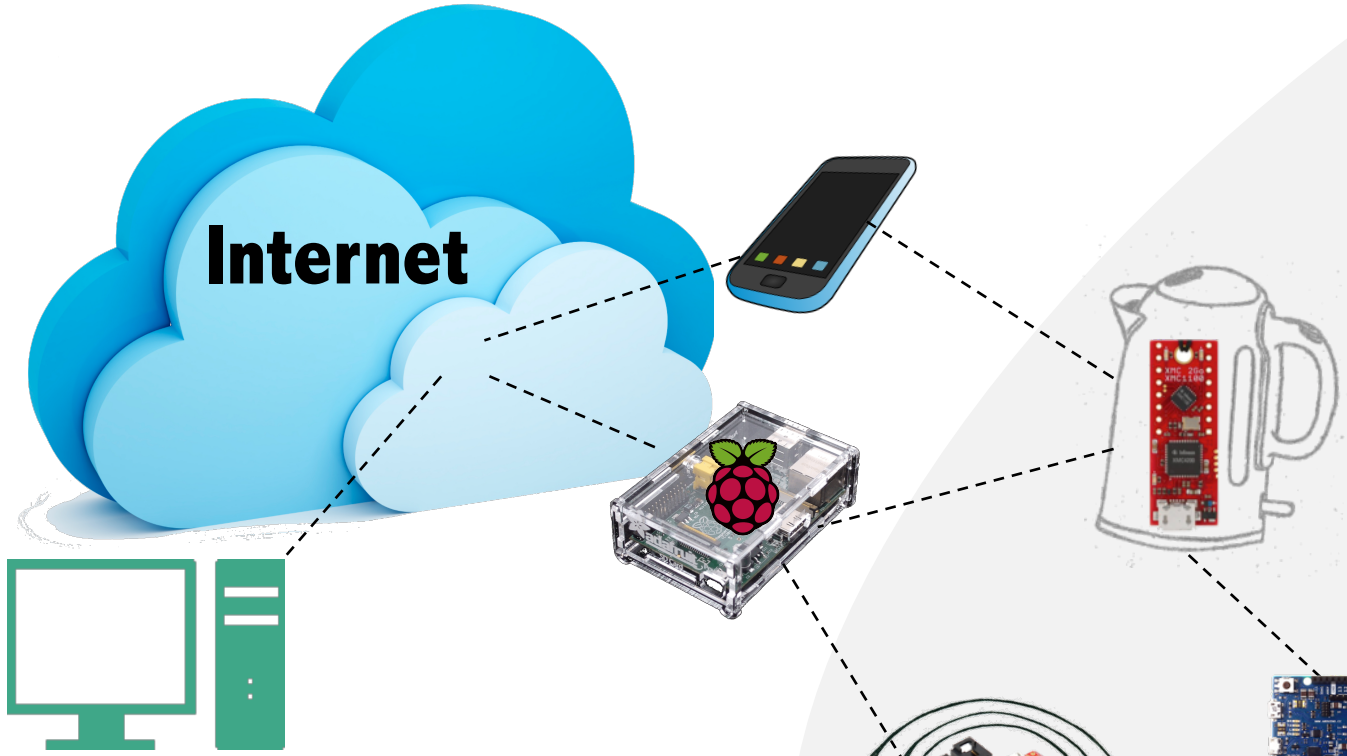
Emmanuel Baccelli

Inria

Agenda

- Context
- RIOT
- IoT Energy Efficiency
- IoT Security
- RIOT & SILECS

IoT Hardware



Low-end IoT devices

- 1000x less energy than RaspberryPi
- kBytes instead of GBytes of memory

Microcontrollers e.g.
- AVR (8-bit)
- MSP430 (16-bit)
- Cortex M (32-bit)
- MIPS
- …

Low-end IoT Devices : Polymorphism

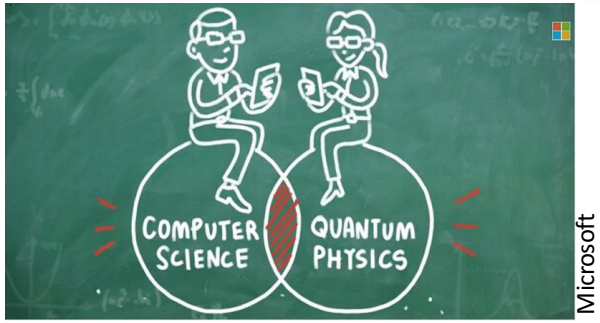
- Various vendors
- Various architectures (8-bit, 16-bit, 32-bit)
- Various low-power communication technologies (BLE, 802.15.4, DECT…)

Context

Biff Tenon

- Ubiquitous computing & connectivity is upon us
  - Giant cyberphysical robot
  - High-end IoT vs Low-end IoT

- Extreme computing power becomes average
  - Now: pooled power, from NSA to botnets & everything in between (e.g. Coinhive covert mining ads)
  - Later: quantum computing?



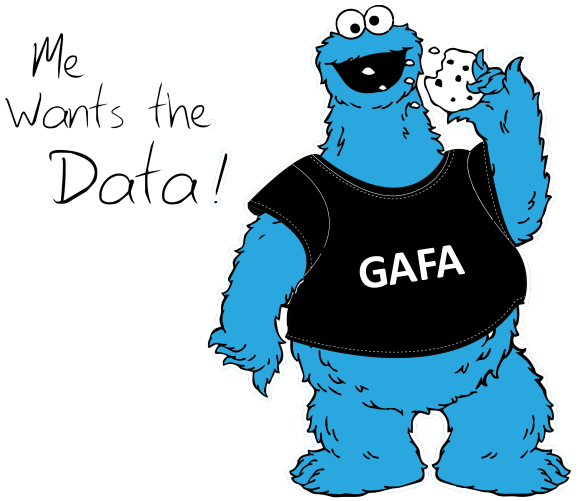
Microsoft

Context

- World War III is upon us (online)
  - geopolitically-driven (state-driven)
  - profit-driven (pirates, zero-day attacks)

Anonymous

- Personal data-hungry Behemoths are upon us
  - captive users, walled gardens, not necessarily secure
  - EU fightback: General Data Protection Regulation (GDPR)



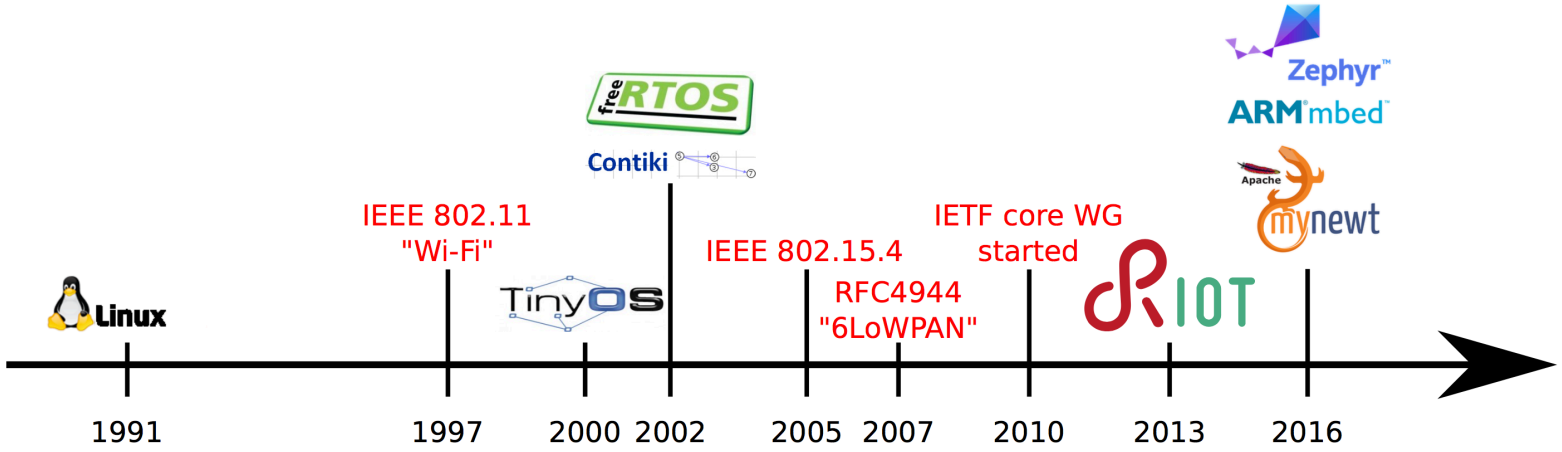
reality.ai

You don’t control what you can’t program

- Software platform on usual machines: **Linux**
  - ✓ provides programmability & full control for developers & users

- On low-end IoT devices?
  - Old : Closed-source, vendor-locked software
  - Next : **RIOT** & new open-source IoT operating systems

O. Hahm et al. "*Operating Systems for Low-End Devices in the Internet of Things: a Survey*," IEEE Internet of Things Journal, 2016.



Agenda

- Context
- RIOT
- IoT Energy Efficiency
- IoT Security
- RIOT & SILECS

RIOT

- Kernel Characteristics
- System-level Interoperability
- Network-level Interoperability
- Open-source Community Aspects

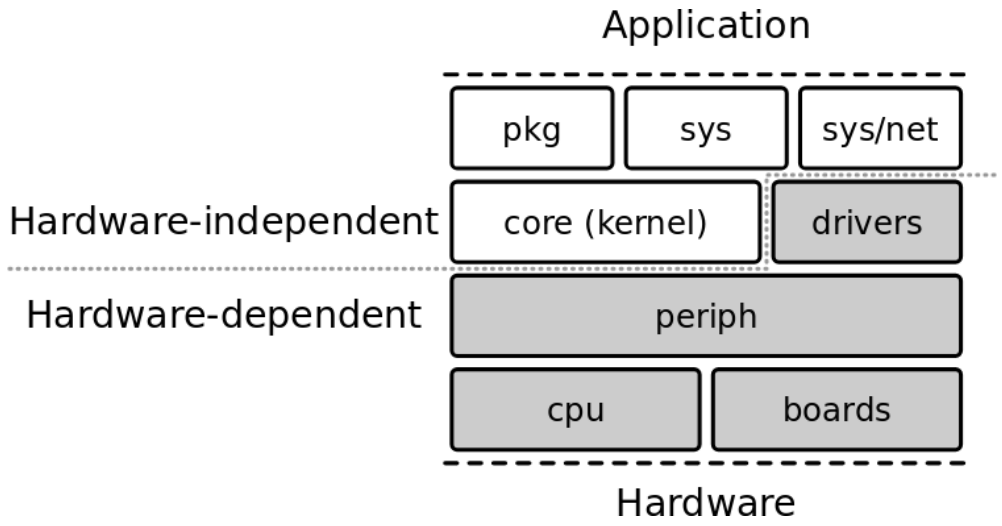
Principles for General Purpose OS for IoT

- ✓ **Unified APIs** – across all hardware, even for hardware-accessing APIs;
  Brings code portability and minimizes code duplication;
- ✓ **Vendor & Technology Independence** – Vendor libraries are avoided;
  Design decisions don't tie RIOT to a particular technology;
- ✓ **Modularity** – building blocks, to be combined in all thinkable ways;
  Caters for versatile use cases & memory constraints;
- ✓ **Static Memory** – extensive use of pre-allocated `structs`;
  Caters for reliability, simplified verification, and real-time requirements;

RIOT: Kernel Overview

Modular architecture based on a minimal, real-time kernel

- Multi-threading, thread synch. with mutexes, semaphores and IPC
- Separate thread context & memory stack (but single memory space)



How?

✓Small Thread Control Block (TCB)

✓Minimized Stack Usage

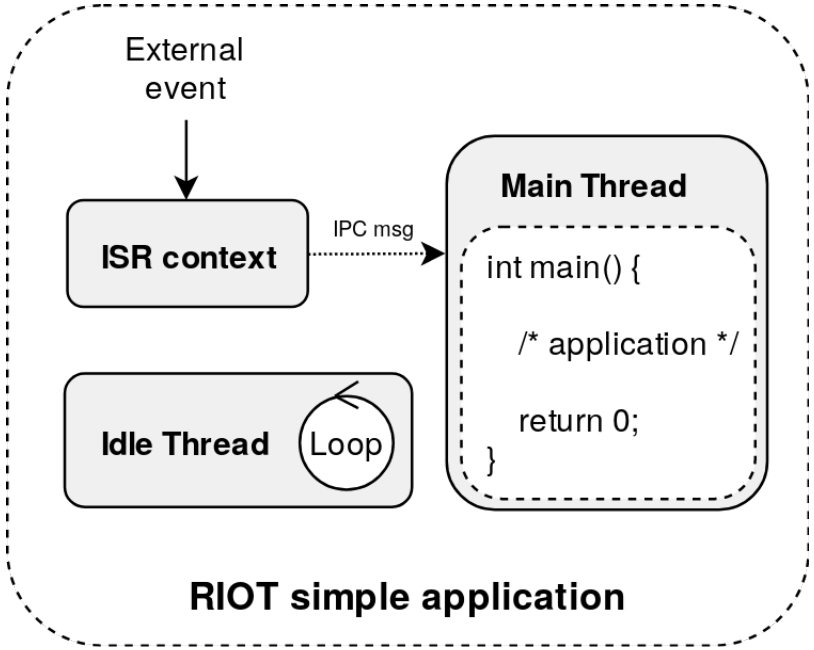
✓Small & fast Inter-Process Communication (IPC)

✓Tickless, O(1) scheduler w. priorities & preemption

(Note: Multi-threading is optional)

E. Baccelli et al. "*RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT*," IEEE Internet of Things Journal, 2018 (to appear).

RIOT: Kernel Overview



Default configuration, 2 threads:

- `main` thread running the main function
- `idle` thread: lowest priority
  - ✓fallback when other threads are blocked/done
  - ✓the `idle` thread **automatically switches to the lowest possible power mode**
- (ISR context handles external events and notifies threads using IPC)

RIOT: Kernel Overview

Some numbers…

On a low-end IoT device (16-bit, 8 MHz):

- ✓ Min. TCB: 8 bytes; Min. Stack Size: 96 bytes
- ✓ IPC: up to 16,000 Messages/s (~ 10k Packets/s for 802.15.4)
- ✓ Kernel: requires only **1kB RAM and 2kB Flash**

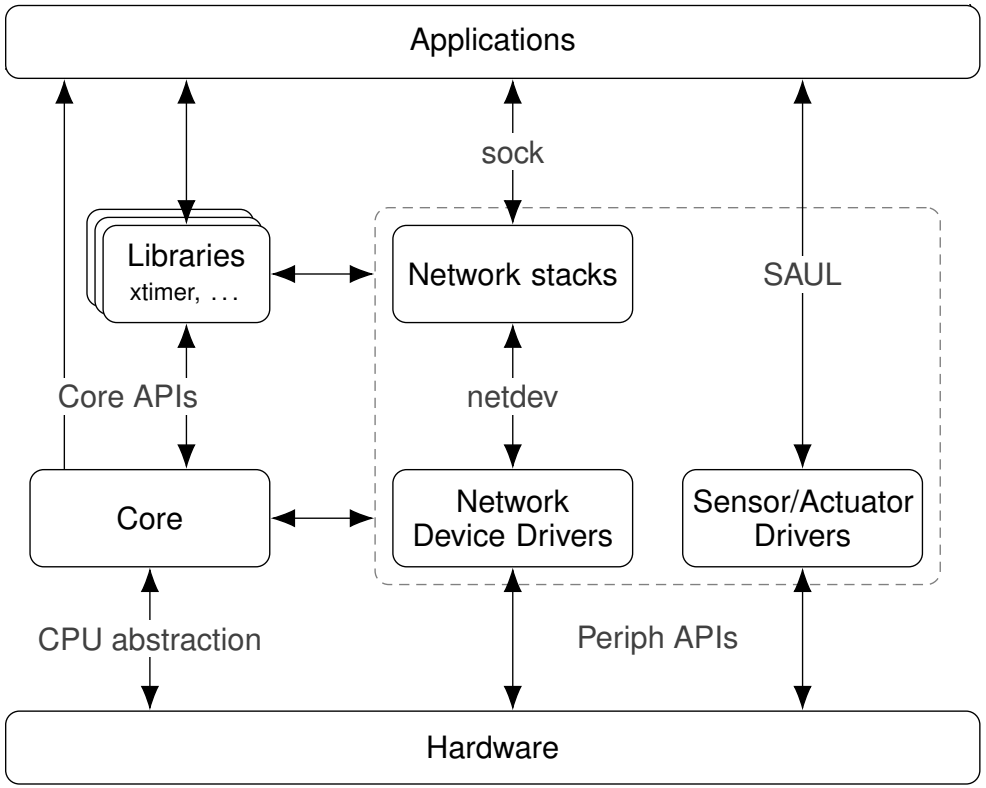
More numbers…

- ✓ On Cortex-M (32-bit): Kernel is 2,8kB RAM and 3.2kB Flash
- ✓ On AVR (8-bit): Kernel is 800B RAM and 4.2kB Flash

RIOT

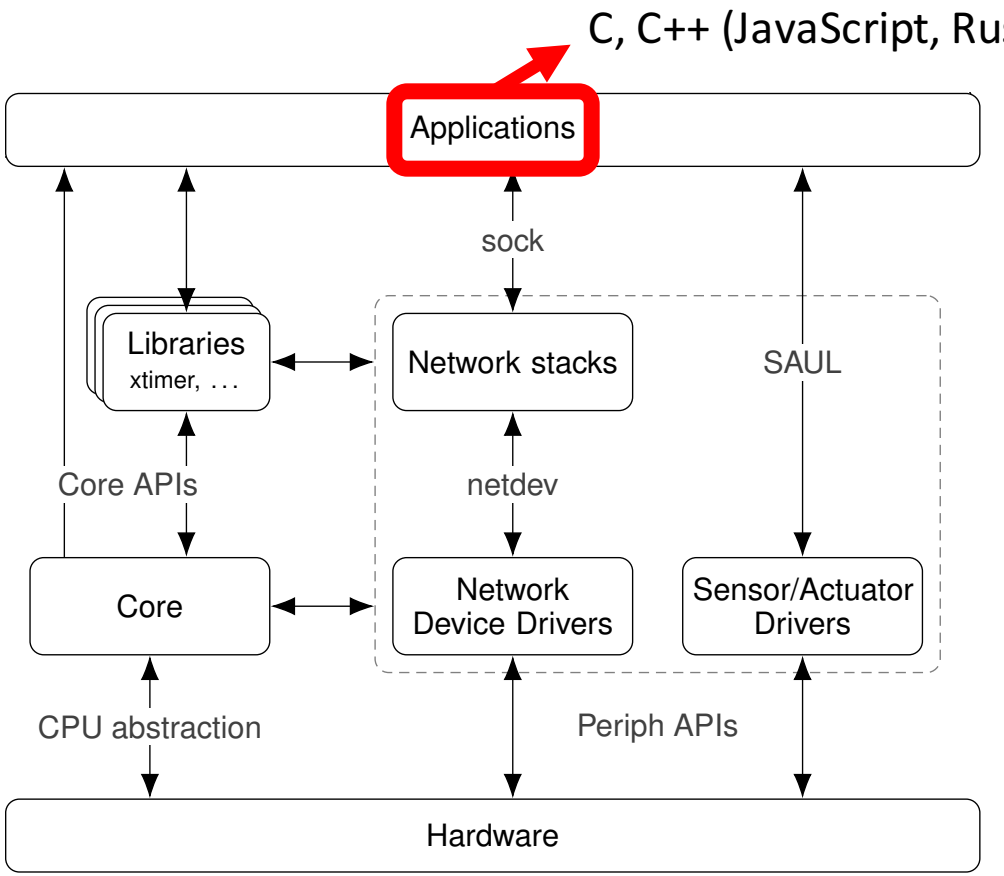
- Kernel Characteristics
- System-level Interoperability
- Network-level Interoperability
- Open-source Community Aspects

RIOT: System-level Interoperability



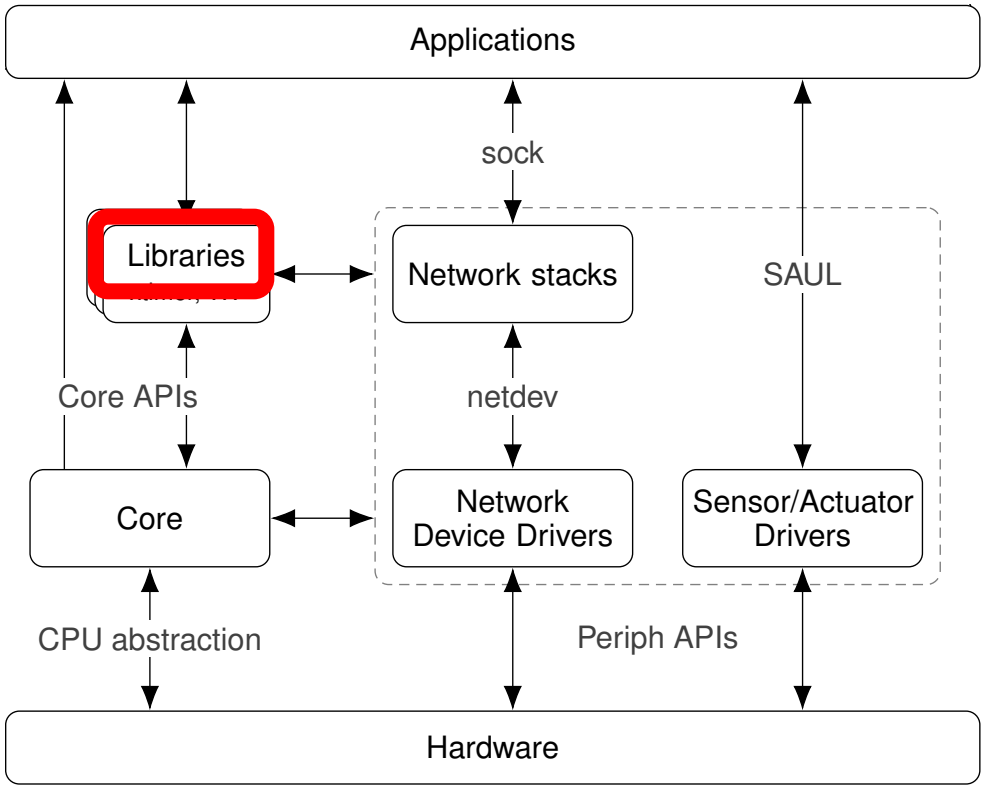
- Core of RIOT in (ANSI) C
- No macro 'magic'
- Board/cpu/drivers abstraction layer
- Well defined APIs, consistent across all supported hardware
  - Cherry-picking POSIX

RIOT: System-level Interoperability



- Drawbacks:
  - some memory overhead, but still fits low-end IoT devices memory budget
  - some more work because re-implement from scratch (behind vendor header files)
- Advantages:
  - **Efficient & highly reusable code** across all supported hardware
  - **Emulation** of RIOT as a Linux process
  - Reusability of well-known **3rd-party tools** such as `gdb`, `valgrind`, `gprof`…

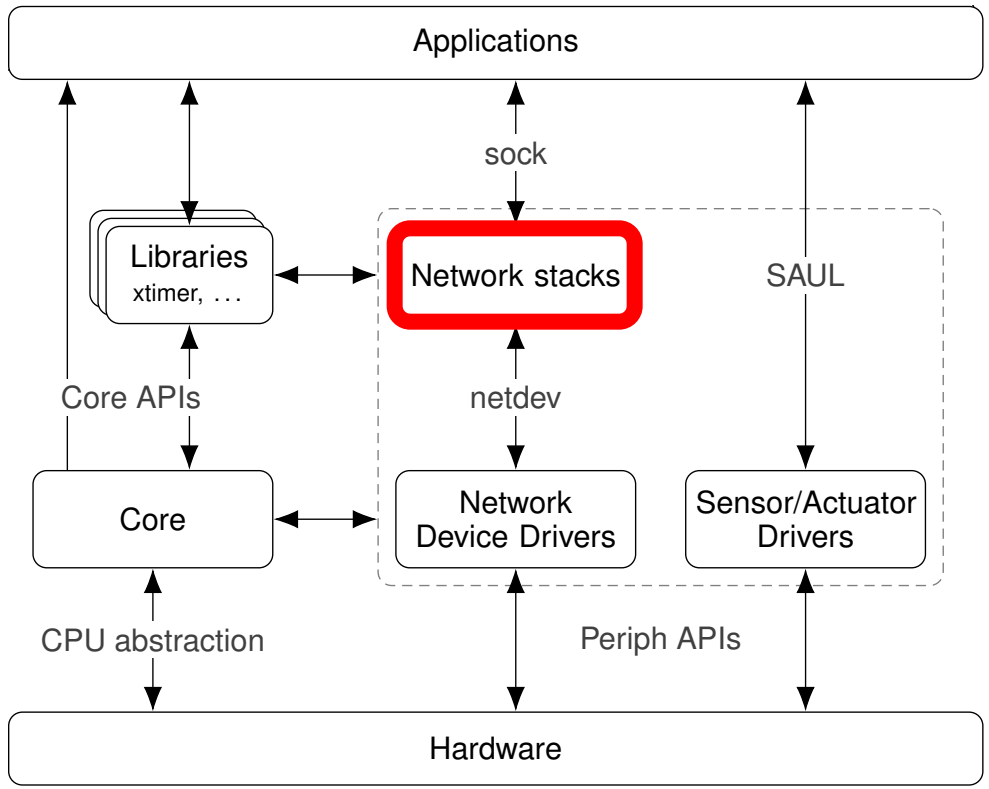
RIOT: System-level Interoperability



- **Packages**: bundling 3rd-party libraries
  - Integrated on-the-fly at build–time
  - Easy to add: just requires 2 Makefiles
  - Patches (if needed) are typically minimal

Package	Overall Diff Size	Relative Diff Size
ccn-lite	517 lines	1.6 %
libfixmath	34 lines	0.2 %
lwip	767 lines	1.3 %
micro-ecc	14 lines	0.8 %
spiffs	284 lines	5.5 %
tweetnacl	33 lines	3.3 %
u8g2	421 lines	0.3 %

RIOT: Network-level Interoperability



Wired & Bus
- CAN
- Ethernet

Low-power wireless LAN & WAN
- IEEE 802.15.4
- LoRa package
- BLE (work-in-progress)

IP Protocols Stacks
- Default stack (GNRC)
- Thread (package)
- lwIP (package)
- OpenWSN (in progress)

Experimental stacks
- CCN-lite (package)
- NDN-RIOT (package)

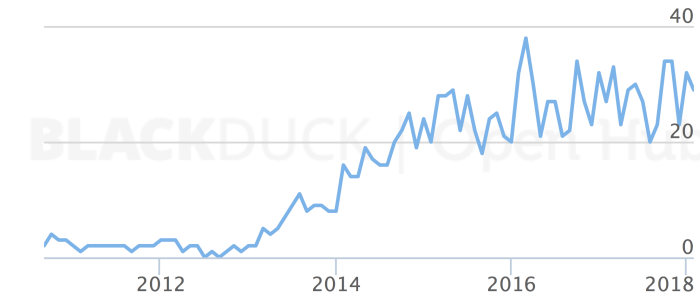
RIOT

- Kernel Characteristics
- System-level Interoperability
- Network-level Interoperability
- Open-source Community Aspects

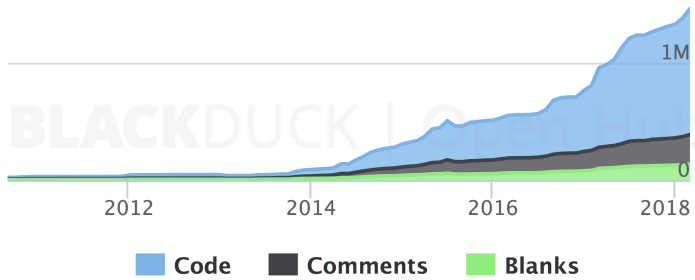
RIOT: Large Open-Source Community

GitHub

github.com/RIOT-OS/RIOT



Monthly contributors



Lines of code

Source: www.openhub.net/p/RIOT-OS

- 2013: started as French-German research project
- 2018: more than 180 contributors worldwide
  - 17,000 commits and 7,200 Pull Requests
  - Support for 100+ types of hardware platforms
  - First products shipping RIOT last year
  - Hundreds of related scientific publications
  - Yearly RIOT Summit conference

    next in Amsterdam, Sept. 13-14, see http://summit.riot-os.org

RIOT: Large Open-Source Community

Grassroots governance by two bodies

- see https://github.com/RIOT-OS/RIOT/wiki/RIOT-Community-Processes

Maintainers

- Status acquired via meritocracy on Github
- Duties: review incoming code contributions
- Rights: can merge in the master branch

RIOT e.V.

Coordinators

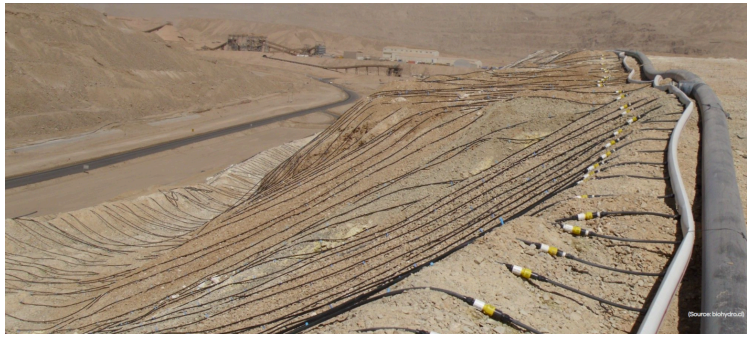
- Status granted by co-founders
- Duties: strategy steering
- Rights: some praise ;)

Inria INVENTORS FOR THE DIGITAL WORLD

Freie Universität Berlin

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Some Deployments & Products Using RIOT



- Telefonica (Chile): deployment in a copper mine
- Cisco (USA) and Huawei (USA): test deployments of ICN-IoT
- Algramo (Chile): deployment in fleet of cereal vending machines

- Hamilton IoT (USA) : environment sensing hardware + cloud services
- Unwired devices (Russia): LoRa communication module
- Eisox (France): smart heating

- Upcoming products
  - OTAKeys / Continental (Belgium): On-Board diagnostics for connected cars
  - Sleeping Beauty (Germany): GPS tracking device with an integrated GSM modem

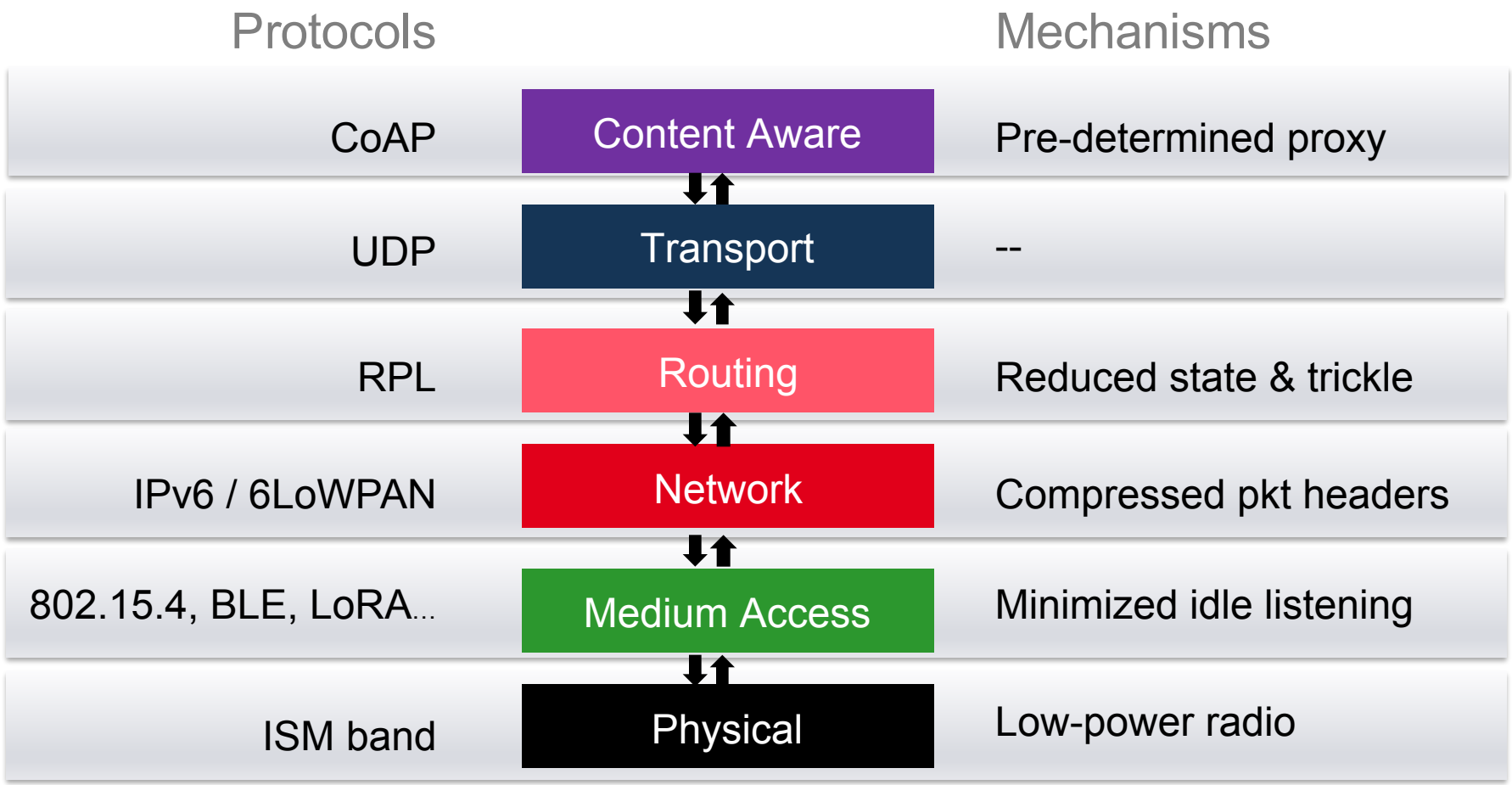
Agenda

- Context
- RIOT
- IoT Energy Efficiency
- IoT Security
- RIOT & SILECS

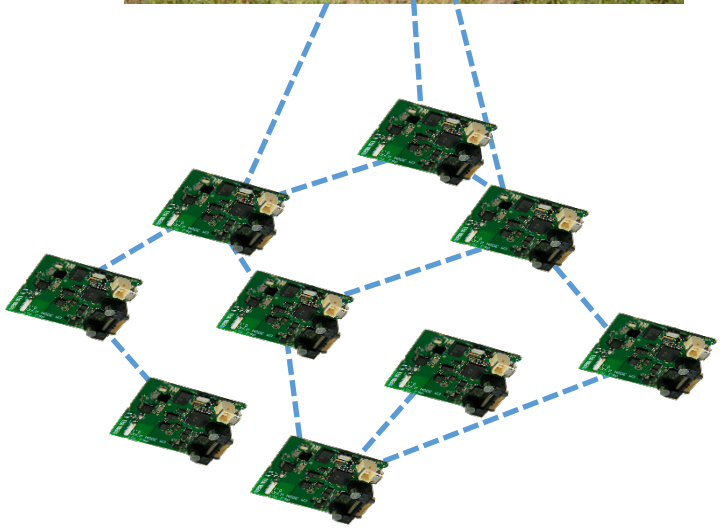
IoT Energy Efficiency

- Motivation
- Cooperative caching with ICN-IoT
- Experimental results

Standard IoT energy efficiency



IoT: Connecting the Physical World & Internet



Nature Monitoring

Intermittent connectivity

Industry 4.0

Micro Satellites

Problem statement

Standard approach breaks if proxy/gateway is unavailable most of the time

- ✓ Nature monitoring

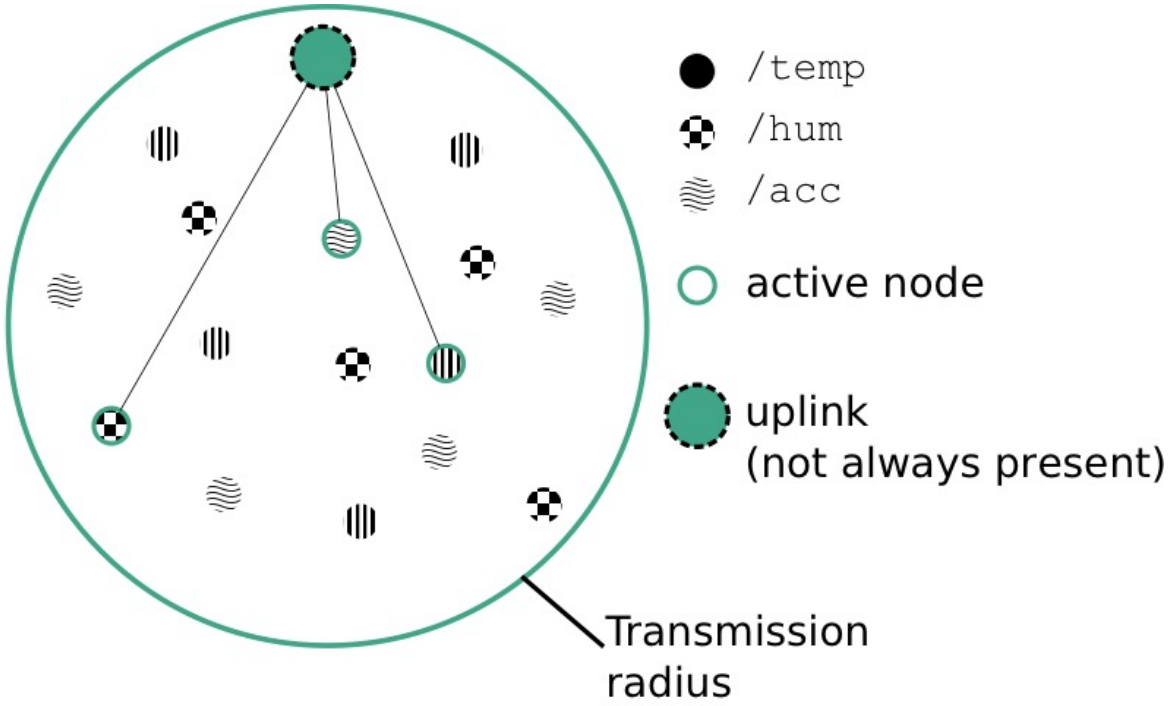
Periodic sensor data gathering via an UAV/drone

- ✓ Factory/storage monitoring

Periodic state check by roaming employees

- ✓ Micro satellites

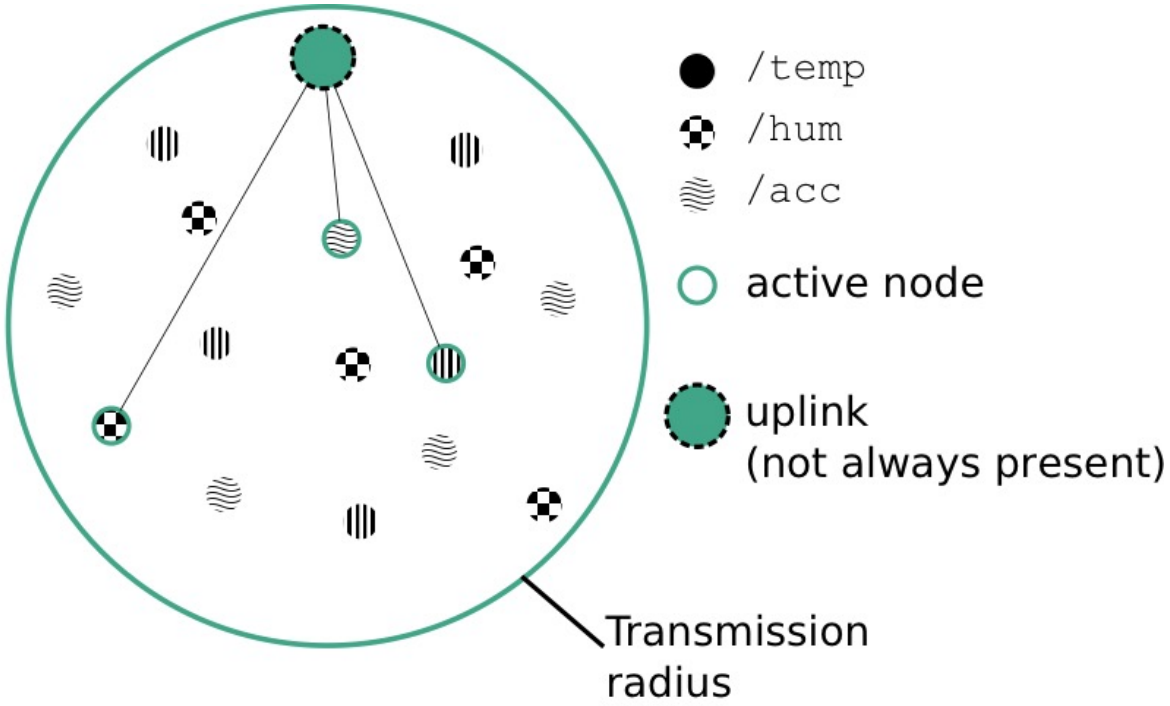
Periodic data gathering via regular satellites



→ **Trade-off:** data availability *vs* energy efficiency

Relevant IoT Data Metrics

- **Diversity**:
  Data from all sources is available.
- **Freshness**:
  The most up-to-date data is available.
- **Lifetime**:
  A sensor value is irrelevant after time L.

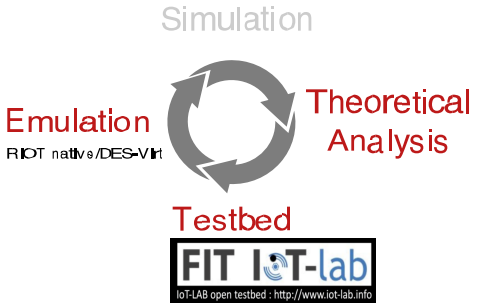


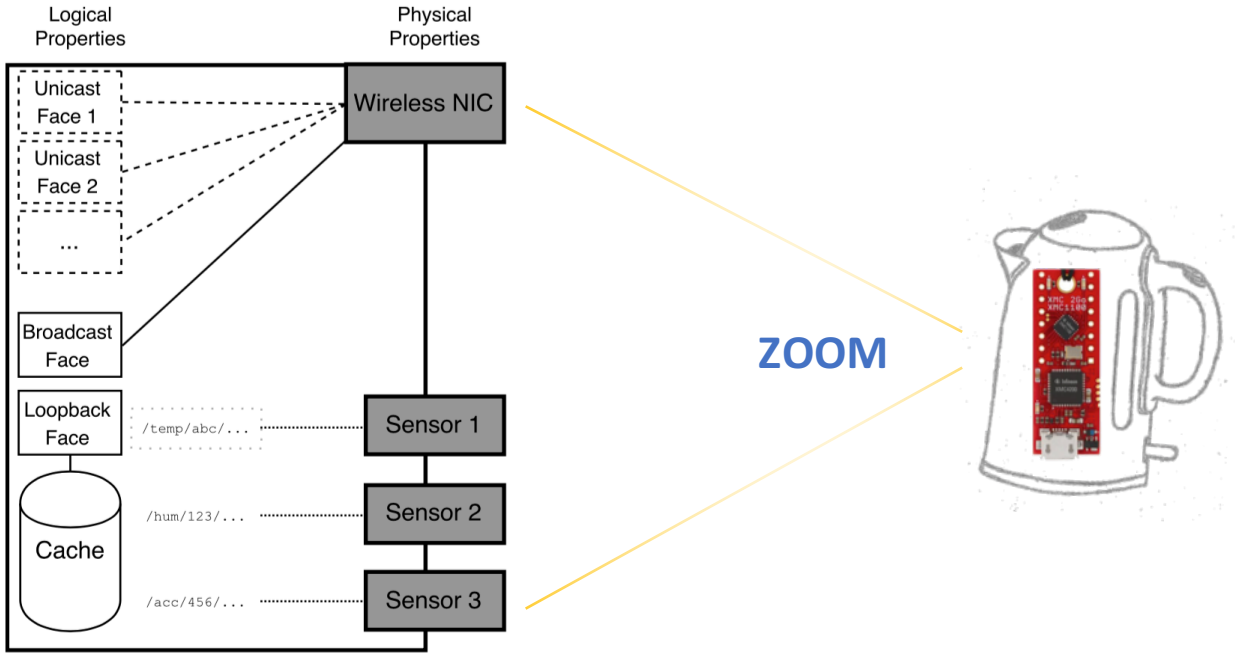
IoT Energy Efficiency

- Motivation
- Cooperative caching with ICN-IoT
- Experimental results

Information Centric Networking (ICN) for IoT ?

- Experimental research enabled by RIOT



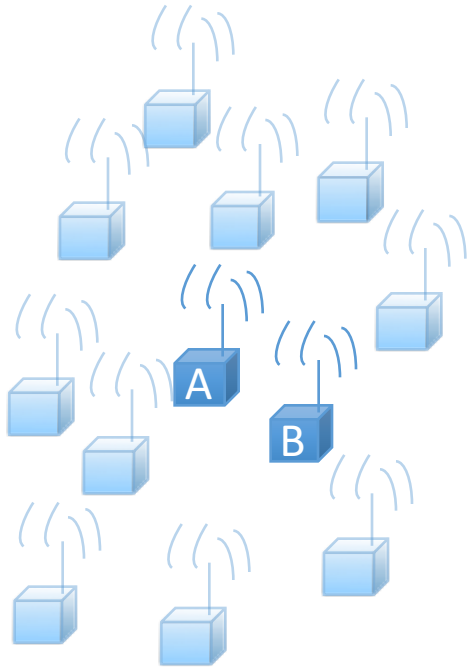


- ICN principle: instead of focusing first on connecting machines, directly **focus on accessing (named) data**.
- ICN aim: **simplified network stack** with only two types of packets:
  - **Interest**
  - **Data** (chunks)

E.Baccelli, C. Mehlis, O. Hahm, TC Schmidt, M. Wählisch, *''Information centric networking in the IoT: Experiments with NDN in the wild,''* in ACM ICN, Sept. 2014.

Idea: NDN+CoCa (Cooperative Caching)

- Intuition: few active devices = no energy *vs* availability trade-off



Interplay of **sleeping** and **ICN caching** & **replication**?

Smart Interplay of IoT protocols, OS & HW

- Design & study of strategies for ICN caching & replication
- Experiments with low-power IoT hardware & OS capabilities
  - ✓ not only radio duty cycling, but make also the MCU sleep
  - ✓ consider both newer & legacy IoT hardware

O. Hahm et al. ’’ *Low-power Internet of Things with NDN and Cooperative Caching*,’’ in ACM ICN, Sept. 2017.

Needed NDN Adaptations

✓Support for content push

- mixture of Interest-Interest, and Immediate Broadcast

✓Support for name wildcards

- in PIT & in Interests, of the shape /* or <prefix>/*
- Still: one content chunk sent per Interest

NDN+CoCa strategies

Sleeping Strategies

- **Baseline**: Radio Duty Cycling
- Uncoordinated Sleeping
- Coordinated Sleeping: Deputy on Watch

Caching Strategies

- **Baseline**: Random Caching
- Max-Diversity Most Recent (MDMR)
- P-MDMR: ‘hardwired’ name pref.

Replication Strategies

- **Baseline**: single broadcast
- Rebroadcast for selected content

Simple analysis of NDN+CoCa with MDMR

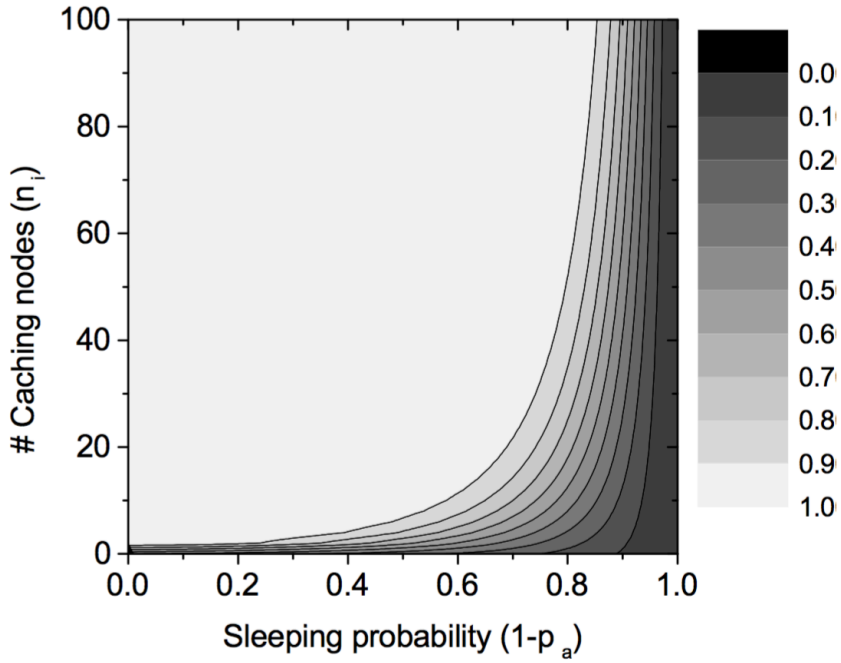
Simplification: no radio interference

Uncoordinated sleeping

→ independent events

→ Bernoulli trials

availability of content for L=1



$|S|$:= Number of sensor sources

n_i := Number of designated caching nodes for content i

L := Lifetime of data

p_a := Probability of being awake

$$\mathbb{E}[collectable\ content] = |S|(1 - (((1 - p_a)(1 - p_a + p_a(1 - p_a)^L)^{n_i - 1}))$$

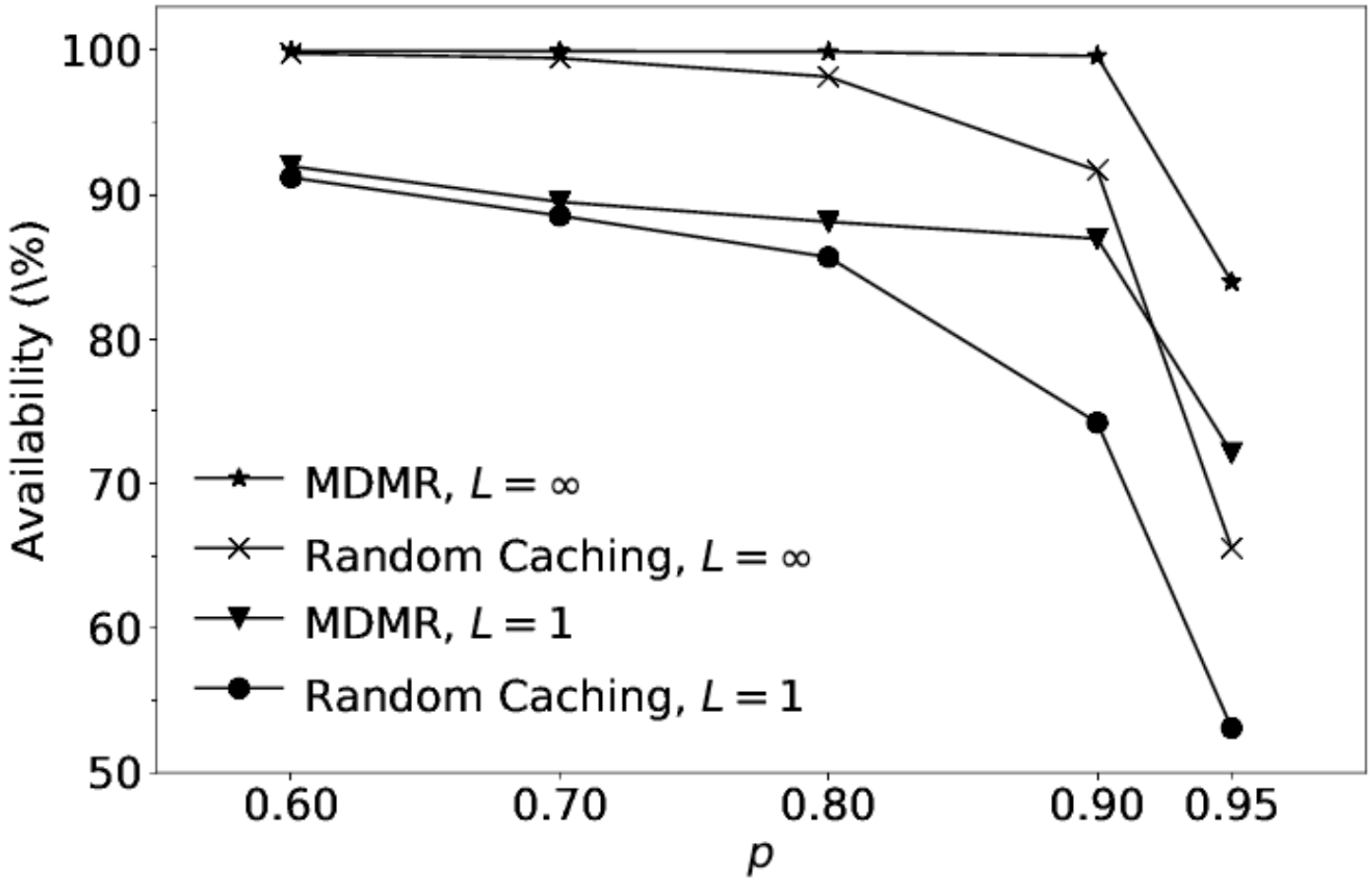
IoT Energy Efficiency

- Motivation
- Cooperative caching with ICN-IoT
- Experimental results

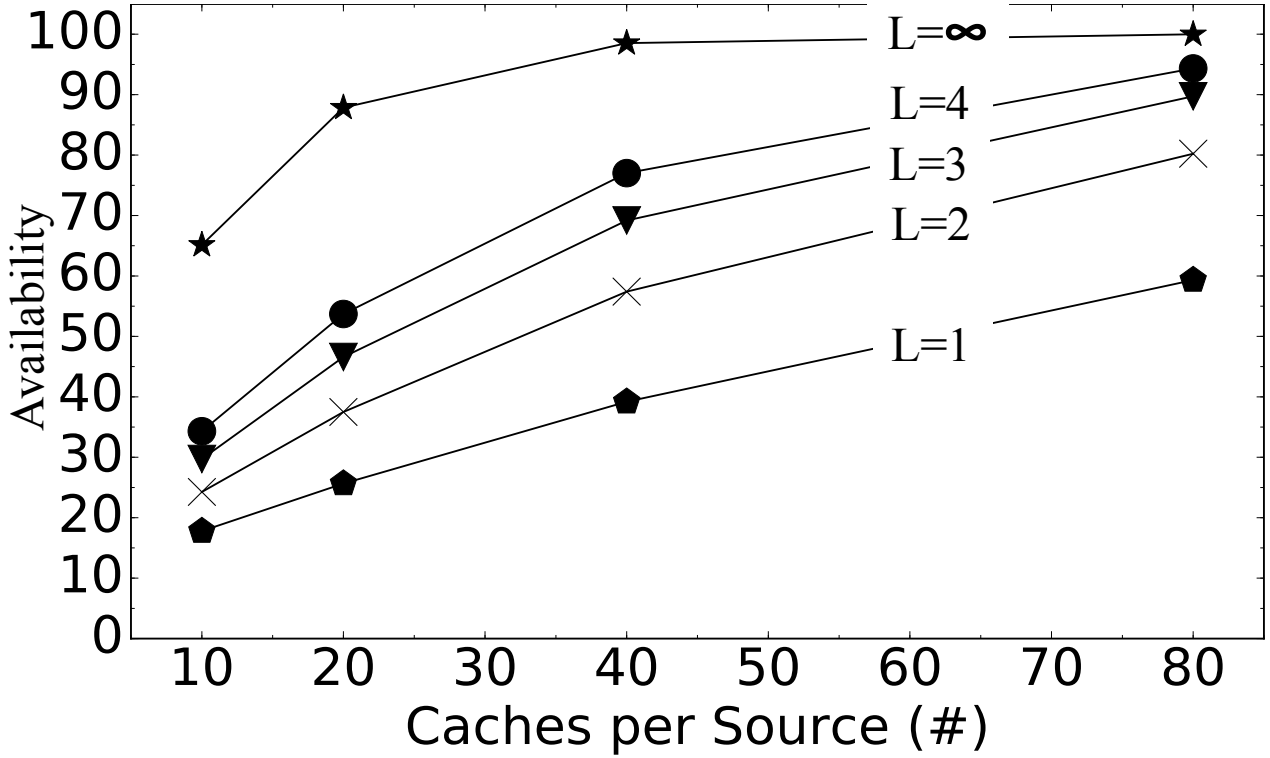
Experimental approach

- ✓Implementation in RIOT
  - Using ICN pkg: CCN-lite
- ✓Used same code for:
  - Emulation on RIOT emulator
    - virtual network with configurable topologies
    - up to 1000 emulated IoT devices on a standard PC
  - Experiments on FIT IoT-lab (IoT-LAB open testbed : http://www.iot-lab.info)
    - heterogeneous hardware in various physical topologies
    - up to 350 IoT devices at one site

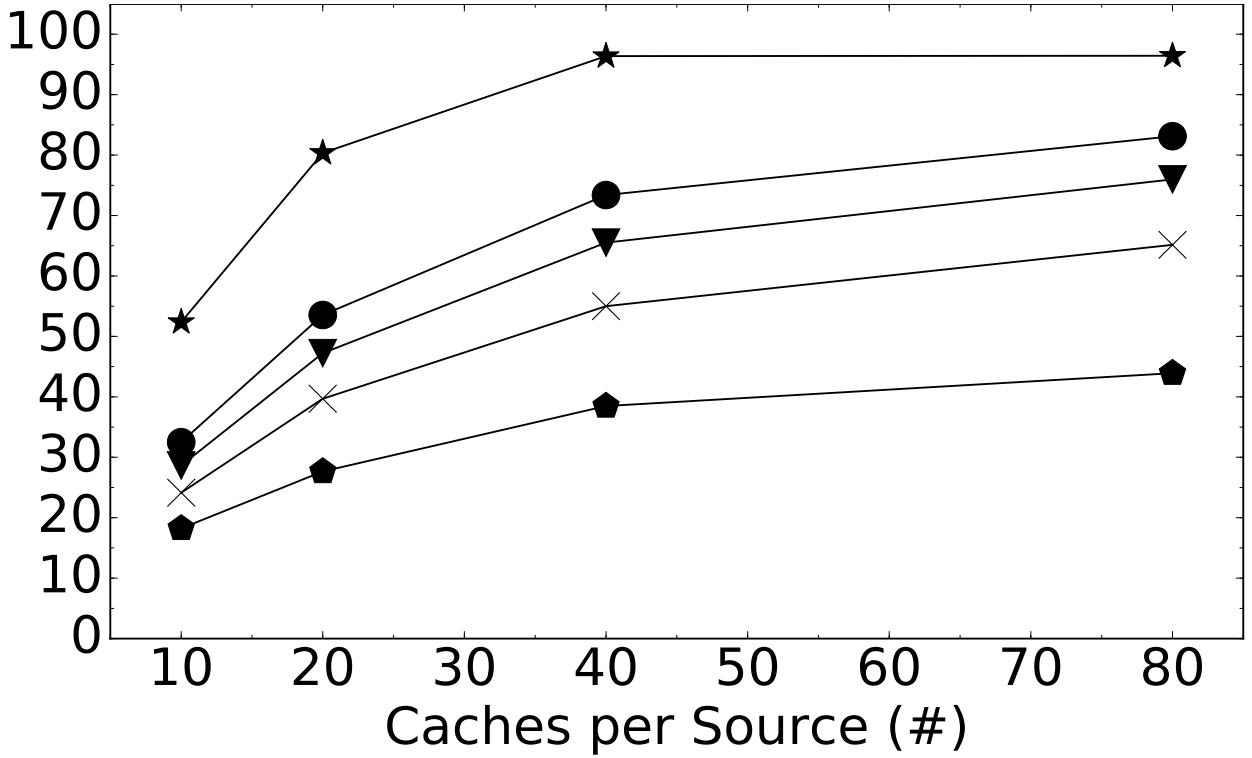
Comparing CoCa (MDMR) to random caching



Matching model & experiment (with RIOT on IoT-Lab)

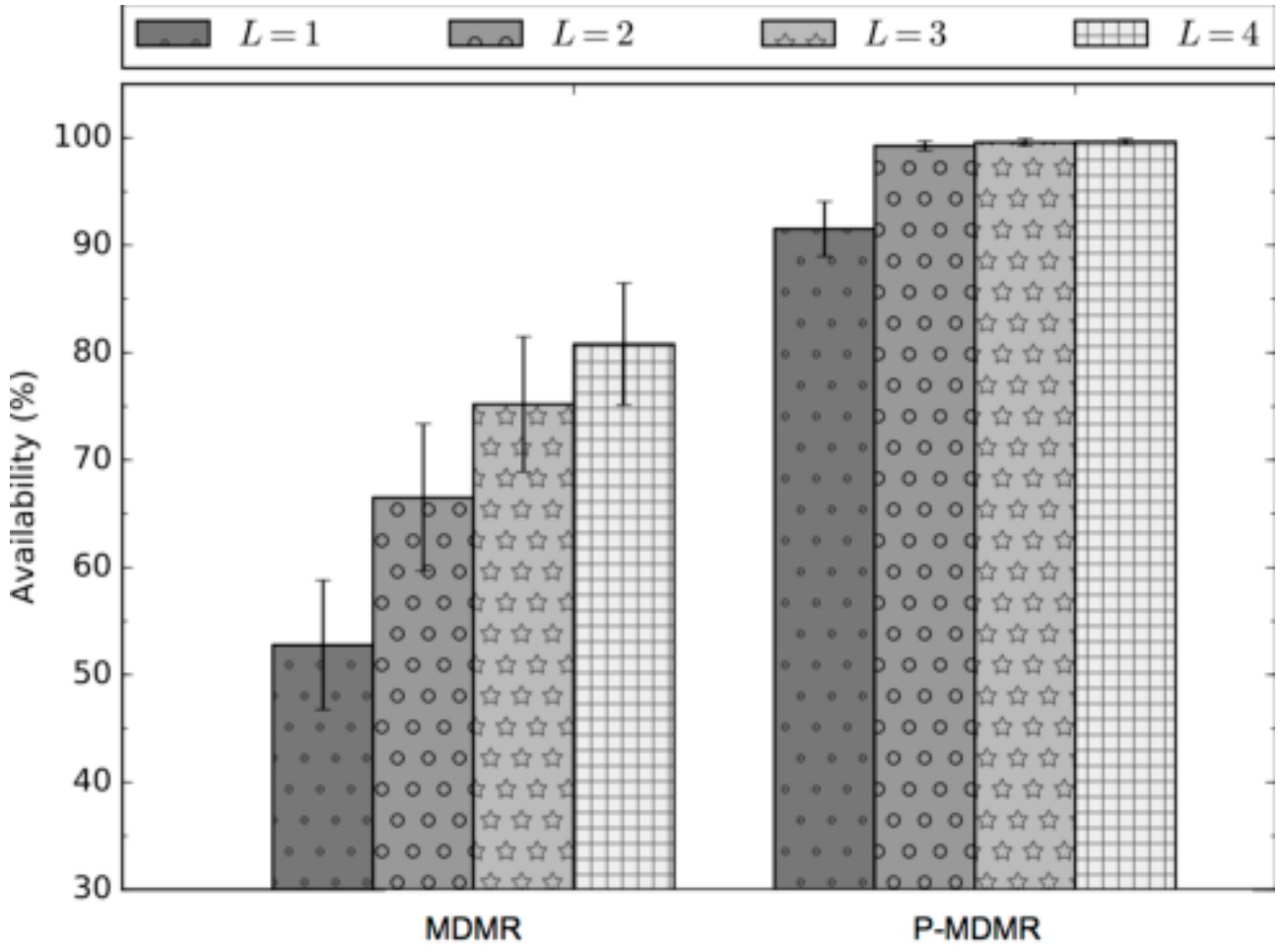


Theoretical Analysis



Testbed results: IoT-Lab in Lille

Scaling up



RIOT emulation with 1,000 nodes

Summary

- Significantly less energy consumption is possible with ICN
  - Compliant with basic NDN
  - Works also on very low-memory IoT devices
  - Works on top of any broadcast-capable network layer

✓ 90% less energy with 90% availability!

Agenda

- Context
- RIOT
- IoT Energy Efficiency
- IoT Security
- RIOT & SILECS

IoT Security

- IoT Attack Surface
- RIOT Security
  - Today
  - Next level

IoT Bottleneck: Security

- Today's IoT: **not an acceptable tradeoff** w.r.t. functionalities *vs* risks
  - B. Schneier: Internet of (Unsecure) Things*

- **Dimensions of the work** needed to change that?

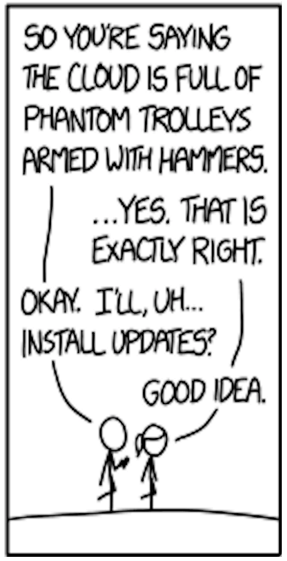
  - Improving functionality
    - Better IoT hardware
    - Richer IoT software

  - Mitigating risks
    - More IoT security

* https://www.rsaconference.com/blogs/bruce-schneier-talks-about-securing-the-world-sized-web-at-rsac-apj-2016

Traditional IT Attack Surface

- **Human vector**
  - Misconfiguration, phishing, social engineering
  - 95% security incidents involve human error*
- **Hardware vector**
  - e.g. Spectre & Meltdown **vulnerabilities** on recent processors: leaks breaking isolation
  - … and **backdoors** from NSA & co ?

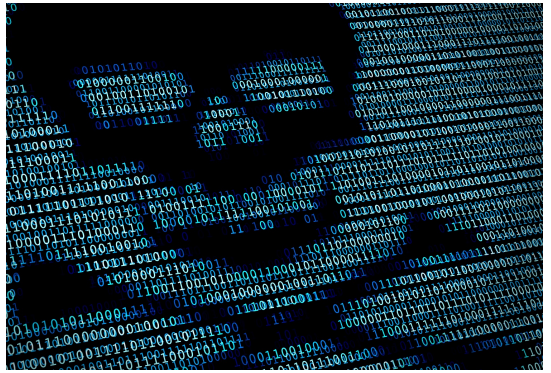


xkcd.com No 1938

* Cybersecurity Intelligence Index, IBM, 2014.

Traditional IT Attack Surface

- **Low-level software vector**
  - e.g. **EternalBlue** vunerability on Windows <10 (NSA exploit turned bad, used in WannaCry)
  - e.g. **HeartBleed** OpenSSL (also on Linux!)
  - Fatal combination: exploit OS & network stack vulnerabilities to inject malicious code



itsecurityguru.org

- **High-level software vector**
  - e.g. malicious PDF exploiting Adobe Reader vulnerabilities

Traditional IT Attack Surface

- **Software supply-chain vector**
  - e.g. backdoor hacked into software updates of Ccleaner application*
  - attack laced legitimate software with malware (distributed by a security company!!!)

xda-developers.com

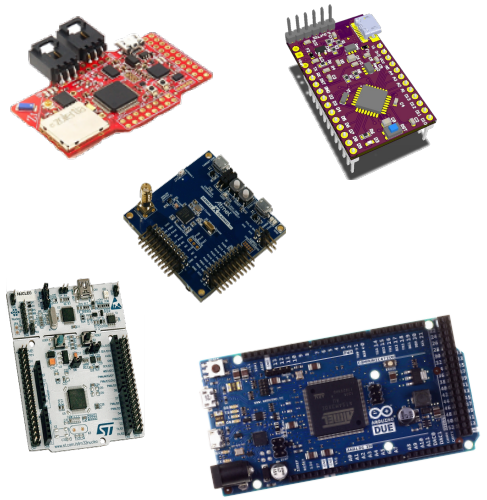
* www.wired.com/story/ccleaner-malware-supply-chain-software-security

IoT *vs* Traditional IT Attack Surface

- IoT～Machine-to-Machine: the **human factor is less important**

- Single binary systems (so far)
  - **no high-level software**

- **Low-end memory, CPU capacities**
  - kBytes of memory instead of Gbytes or more
  - MHz instead of GHz
  - mW or less, instead of W or more



IoT *vs* Traditional IT Attack Surface

- IoT∼giant cyberphysical robot: hacked system can cause direct physical harm
  ⇒ **acceptable risks are changed**

- Sensors everywhere, all the time
  ⇒ **scope of privacy breaches are changed**

- Industrial IoT applications*
  ⇒ required level of system availability is higher

* AR Sadeghi et al. "*Security and Privacy Challenges in Industrial Internet of Things*," in ACM DAC, 2015.

In a nutshell

- ~~Humans~~
- Hardware
- Low-level software
- ~~High-level software~~
- Software supply-chain

- Good news:
  - attack surface *might* be smaller than usual

- Bad news:
  - harsher contraints & potentially more impactful attacks
  - no human in the loop means its harder for some aspects (bootstrap…)

IoT Security

- IoT Attack Surface
- RIOT Security
  - Today
  - Next level

RIOT Security Today: Basics

- State-of-the-art crypto primitives
  - Symmetric crypto
  - Asymmetric crypto (elliptic-curve)
- 1000 eyes
  - Clean, open-source code
  - Low number of LoC (total ~500k LoC)
- Standard process in place to handle zero-day vulnerabilities
  - security@riot-os.org

Next-level Security in RIOT

Trends towards:

- Crypto primitives: higher speed, stronger security, smaller memory footprint
- Formally verified IoT software modules
- Secured IoT software updates & supply-chain
- Trusted execution

IoT Crypto Primitives

- Devices deployed now will last for years
  - Maybe decades!
    - MSP430 runs **12 years on an AA battery**.
    - Energy harvesting

- **Future-proof crypto** is thus crucial. Quantum resistance?
  - Trade-off: **key & signature size *vs* speed** *
    - ex. ECC 256b key *vs* McEliece 500kB key
    - ex. ECC 80B signature *vs* MQDSS 40kB signature

* S. Samardjiska '*Post Quantum Cryptography for the IoT*,' RIOT Summit, 2017

IoT Crypto Primitives

It is nevertheless possible to **prepare IoT crypto for post-quantum now**

- symmetric crypto needs upgrade but same security $=\sim$ double key size
- asymmetric crypto
  - some techniques would break entirely (e.g. RSA?)
  - some techniques are quantum-resistant (hash-based signatures…)

Considered IoT Crypto Primitives

- IoT Symmetric Crypto
- IoT Asymmetric Crypto
- Operations over Encrypted IoT Data

IoT Symmetric Crypto

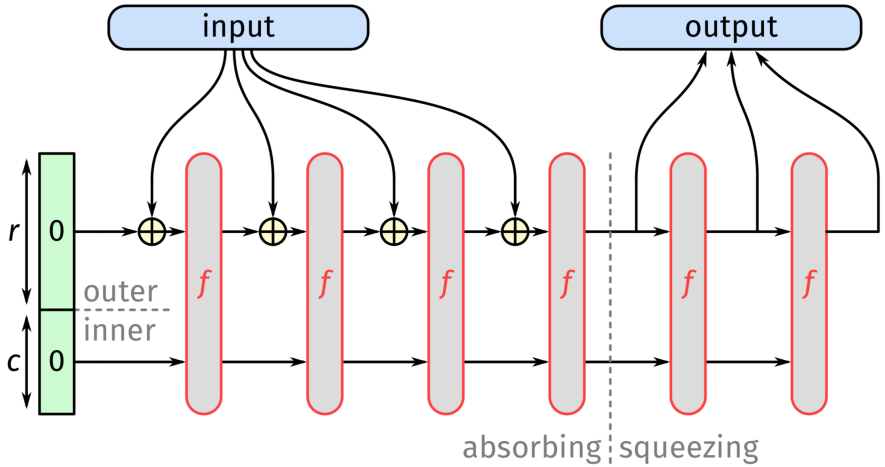
- Symmetric = same key everywhere.

More flexible primitives:

SHA-3's **sponge construction*** for hashing

- Easy to (re)configure security level
  - Just vary capacity c
- Shared code can provide various functions
  - Pseudo-random number generator
  - Message authentication code (MAC)
  - Stream encryption
  - (more with the duplex construction)

- Initiating experimental work to evaluate this prospect on top of RIOT



G. Van Assche *'Permutation-based cryptography for the IoT,'* RIOT Summit, 2017.

* G. Bertoni et al. *'Cryptographic sponge functions'*, 2011.

IoT Asymmetric Crypto

- Symmetric crypto:
  - communicating entities need a shared secret (the key)

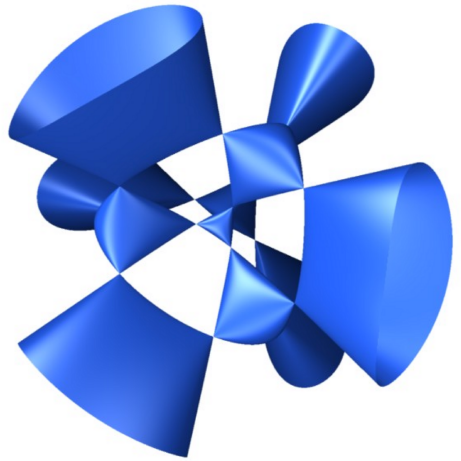
  $\Rightarrow$ key distribution problem, Pre-Shared Key (PSK) is norm on IoT

- Asymmetric crypto with public/private keys:
  - Public Key Infrascruture (PKI) solves key distribution
  - Public key allows digital signatures

IoT Asymmetric Crypto

Smaller & faster crypto

- more efficient implementation
  - **tweetnacl** (Bernstein et al.): Source funnily fits in 100 tweets, using curve25519
- more efficient algorithms (GRACE Inria)
  - **uKummer** *: Smarter use of algebraic geometry
    - software-only hyperelliptic cryptography on constrained platforms
    - demonstrated on AVR 8-bit and ARM Cortex-M 32-bit
    - up to 70% faster & 80 % smaller compared to using curve25519
  - **qDSA** **: even smaller stack & code size
    - a qDSA packge was merged into RIOT last week



* J. Renes et al. '*µKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers*', CHES, 2016.

** J. Renes, B. Smith '*qDSA: Small and Secure Digital Signatures with Curve-based Diffie–Hellman Key Pairs*', ASIACRYPT 2017.

Operations over Encrypted IoT Data

The cloud, or the server hosting IoT database may not be trusted

⇒ Nevertheless it may be required to (batch) process IoT data

- Use of **partially homomorphic** crypto
  - Talos and Pilatus prototype platforms *
  - **Allows some operations (range, sum) over encrypted data**
  - Using Elliptic-Curve ElGamal crypto-system (instead of Paillier)
  - Encryption by low-end IoT devices themselves (demonstrated on Cortex-M3)

* H. Shafagh et al. *'Secure Sharing of Partially Homomorphic Encrypted IoT Data,'* ACM SenSys, 2017

Operations over Encrypted IoT Data

- Body of work on **differential privacy** in the field of smart metering *
  - Followed seminal work from **Dwork** in 2006

- Principle: **add (some) noise to IoT data points**
  - Differential guarantee = analysts draws same conclusions about an individual whether the individual includes himself in the dataset or not
    - **Still able to extract coarse signal** from aggregate data (e.g. mean, average…)
    - **No privacy violation in practice**

* V. Rastogi et al. '*Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption*', ACM SIGMOD 2011.

Next-level Security in RIOT

Trends towards:

- Crypto primitives: higher speed, stronger security, smaller memory footprint
- Formally verified IoT software modules
- Secured IoT software updates & supply-chain
- Trusted execution

Humans (even if skilled) make buggy code

- **Formally verified crypto code**
  - HACL* library: written in F* programming language (PROSECCO Inria)
  - F* code formally verified (memory safety, mitigations against timing side-channels, and functional correctness)
  - F* code then compiled to readable C code
  - Elements of HACL* already in Firefox
  - HACL* currently being integrated into RIOT
- Next :
  - critical network stack elements (TLS 1.3) generated with F*
  - critical system modules (parts of the kernel) generated with F*

* JK Zinzindohoué et al. *'HACL*: A verified modern cryptographic library,'* ACM CCS, 2017

Towards RIOT Kernel Formal Verification?

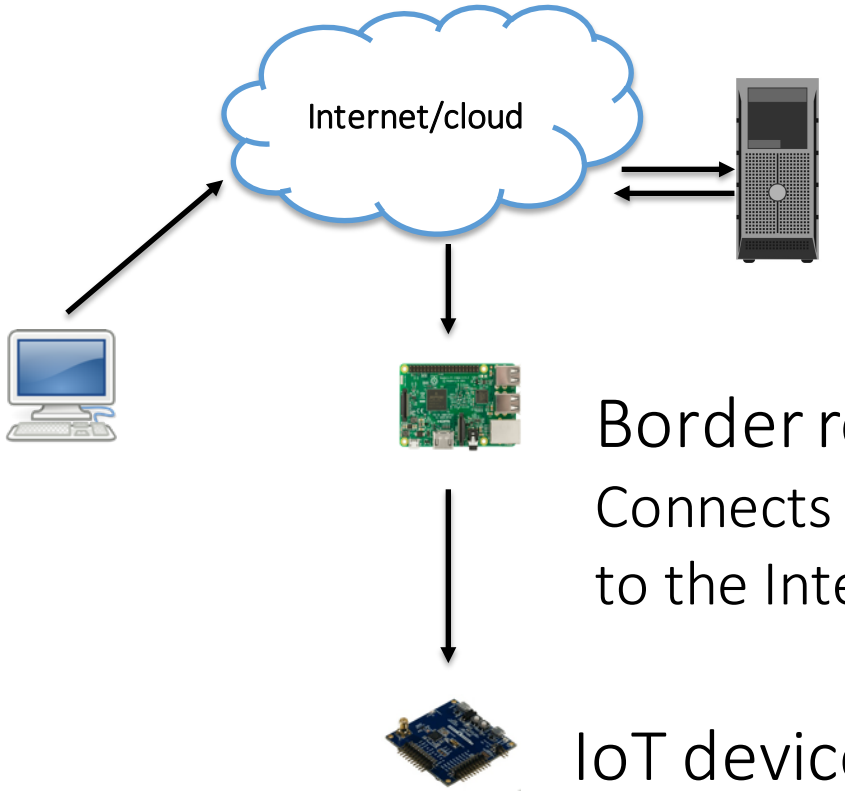
- RIOT kernel is small: 2600 lines of .c and 4200 lines of .h
- First step: using Frama-C, annotate the kernel and prove correctness
  - Method manipulating *functional specifications*, and then *proving* that the source code satisfies these specifications
- Next, consider full kernel reimplementation in F* ?

Next-level Security in RIOT

Trends towards:

- Crypto primitives: higher speed, stronger security, smaller memory footprint
- Formally verified IoT software modules
- Secured IoT software updates & supply-chain
- Trusted execution

You can’t secure what you can’t update



Update Server:
stores updates
& serves them
to IoT devices

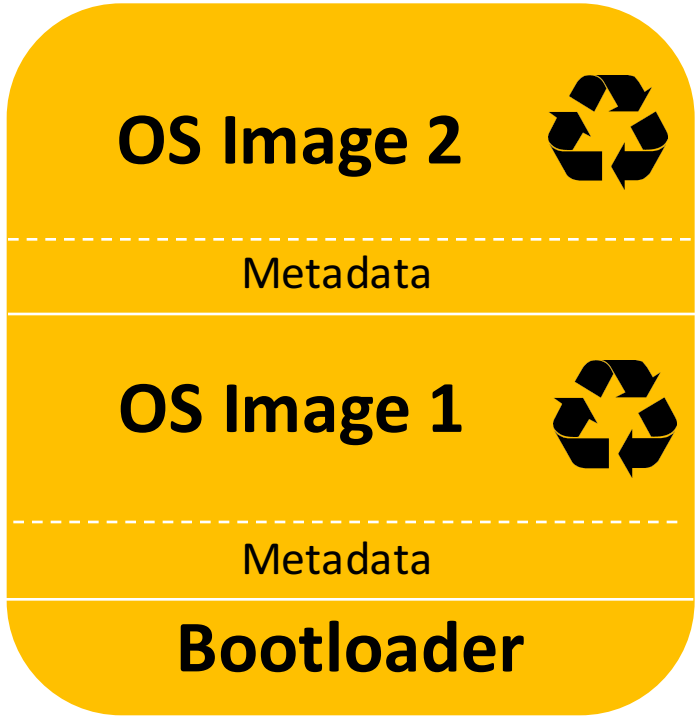
Developer:
Builds, signs &
uploads updates

Border router:
Connects device
to the Internet

IoT device:
Downloads, verifies
and boots updates

Full IoT Software Updates: Firmware Updates

You thought you were tight w.r.t. memory?



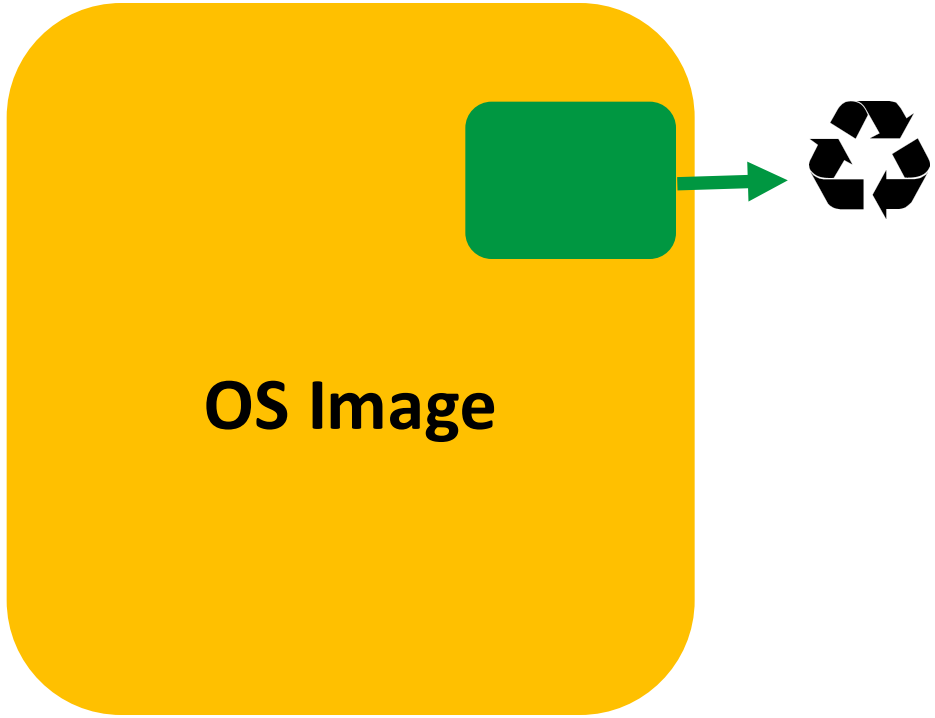
Memory must be further split :

- Bootloader
  - (e.g. minimalistic config or RIOT)
- Several OS Images
  - Typically need >2 for roll-back
- Metadata
  - IETF working group SUIT working on standardizing metadata for IoT firmware *

* https://datatracker.ietf.org/group/suit/documents/

Partial IoT Software Updates

Efficiency of firmware updates?
⇒ try partial software update



Various approaches:

- **differential updates** of arbitrary parts of the binary

⇒ Efficient but risky

- modular program & **dynamic binary module loading**

⇒ More robust but more complex

- use **interpreted languages** (instead of compiled)

⇒ Elegant but interpreter overhead

Runtime .js container demonstrated to work on Cortex-M based low-end IoT devices with RIOT *

⇒ next: explore sandboxing properties of this contruct?

* E. Baccelli et al. "*Scripting Over-The-Air: Towards Containers on Low-end Devices in the Internet of Things*," IEEE PerCom, 2018 (to appear).

Trusting IoT Software Supply

Need to verify legitimacy of software updates!

- SUIT working group at IETF: metadata format + signatures etc. for IoT firmware
  - https://datatracker.ietf.org/group/suit/documents/

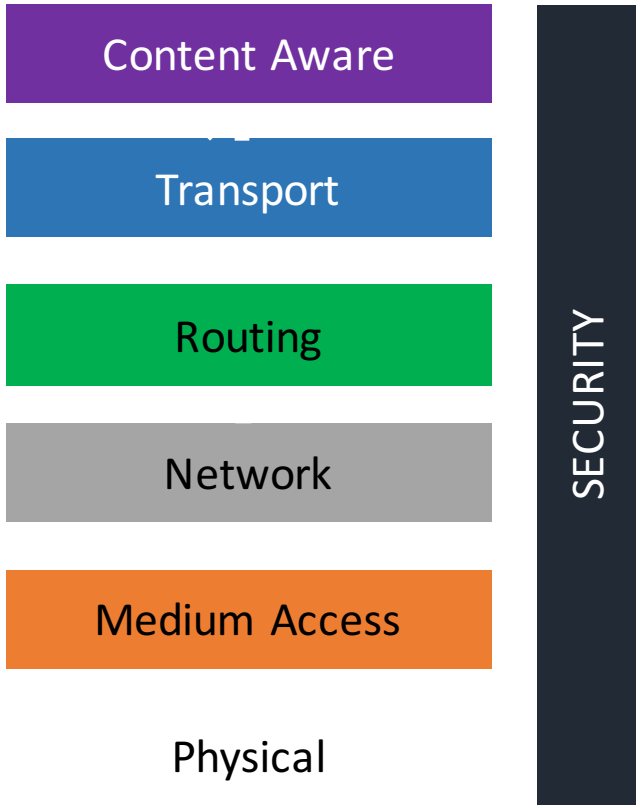
- CHAINIAC * **decentralized framework** to verify software update legitimacy
  - source code & binaries with **reproducible builds**
  - collective signing & **blockchain**-like decentralized consensus on **release history**
  - Series of work on this topic from J. Cappos group at NYU

Side note: enforcing legitimacy can turn bad -- beware of **Treacherous Computing** (R. Stallman)

* K. Nikitin et al. '*CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds,*' USENIX Security Symposium, 2017.

Trusting IoT Software Supply

- Of course you also need secure network protocols…



OSCORE end-to-end encryption, integrity, & replay protection for CoAP (header/payload)

DTLS (over UDP) for confidentiality, integrity, authenticity with public/private keys

(secure join, routing protocol security…)

Link-layer security sublayer for authentication, confidentiality, with symmetric keys

Next-level Security in RIOT

Trends towards:

- Crypto primitives: higher speed, stronger security, smaller memory footprint
- Formally verified IoT software modules
- Secured IoT software updates & supply-chain
- Trusted execution

Trusted Execution on IoT Hardware

- Principle: **secure area of a processor** for isolated execution, integrity of trusted applications & confidentiality of their assets

- **Sancus*** on MSP430 16-bit microcontrollers (OpenMSP430)
  - Prototype **isolating software components** via memory curtaining
    - Added MMU and crypto HW unit on openMSP430 (open source!)
    - Text/Data/ProgramCounter states monitoring/matching, per software component
  - **Remote attestation** & authenticates communication with software component
    - HW crypto enables key derivation per software component
  - Sancus2.0 tested in automotive context. Claims only 6% energy overhead
  - Project initiated by KU Leuven to integrate RIOT and Sancus

- Similar: **TrustZone** for popular ARM Cortex-M 32-bit microcontrollers
  - Upcoming Cortex-M33 and Cortex-M23 micro-controllers

* J. Noormans et al. '*Sancus 2.0: A Low-Cost Security Architecture for IoT Devices*', ACM Transactions on Privacy and Security, 2017

Trusted Execution on IoT Hardware

- **TEEP** working group at IETF *
  - Context: delete, **update applications** running in the TEE
  - Goal: **communication** between the TEE, a relay outside TEE & a remote server

    ⇒ Trusted execution environment protocol (TEEP)

Side note: installing new software in the TEE **increases attack surface**…

* https://datatracker.ietf.org/wg/teep/documents/

Agenda

- Context
- RIOT
- IoT Energy Efficiency
- IoT Security
- RIOT & SILECS

RIOT: Tools & Community Processes

- One release every 3 month: <year>.<month> (2018.01, 2018.04, etc)
  - In-depth code reviews
    - Including C99 compliance
  - Nightly builds
  - Online documentation generated with Doxygen

    ⇒ https://doc.riot-os.org

- Distributed Continuous Integration: Murdock https://ci.riot-os.org
  - Build for all targets (more than 15k configurations!)
  - Static tests (Cppcheck, Coccinelle, etc)

    ⇒ Hardware in the loop (WIP)

SILECS ?

FIT IoT-lab

IoT-LAB open testbed : http://www.iot-lab.info

Conclusion

RIOT is a natural extension to FIT IoT-Lab, allowing:

- within IoT-Lab, easy & advanced programming of IoT-lab nodes
- beyond IoT-Lab running the *same code* on 100+ other types boards

There is a large & lively community around RIOT:

- On-going academic research
- Industry is part-taking

There is lot still to do. Help & ideas are welcome!

Thanks! Questions?

GitHub

github.com/RIOT-OS/RIOT