The friendly operating system for the IoT

by Thomas Eichinger (on behalf of the RIOT community)

OpenIoT Summit NA 2017

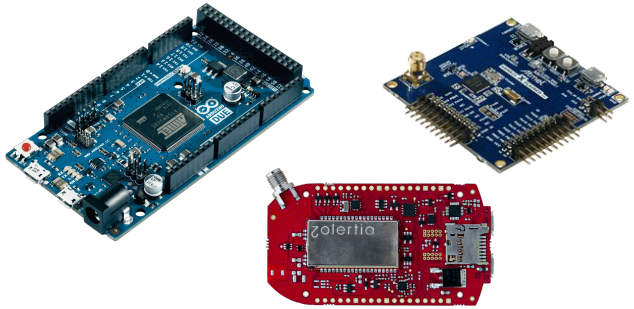
Why?
How?
What is RIOT?

Why?

How?

What is RIOT?

Why a software platform for the IoT?



- Linux, Arduino, … bare metal?

- But as IoT software evolves …
  - More complex pieces e.g. an IP network stack
  - Evolution of application logic
- … non-portable IoT software slows innovation
  - 90% of IoT software should be hardware-independent
    → this is achievable with a good software platform (but not if you develop bare metal)

Why a software platform for the IoT?

- ✓ faster innovation by spreading IoT software dev. costs
- ✓ long-term IoT software robustness & security
- ✓ trust, transparency & protection of IoT users' privacy
- ✓ less garbage with less IoT device lock-down

Why?

How?

What is RIOT?

How to achieve our goals?

Experience (e.g. with Linux) points towards

- Open source
- Free core
- Driven by a grassroot community

Indirect business models

Geopolitical neutrality

Main Challenges of an OS in IoT

Low-end IoT device resource constraints

- Kernel performance
- System-level interoperability
- Network-level interoperability
- Trust

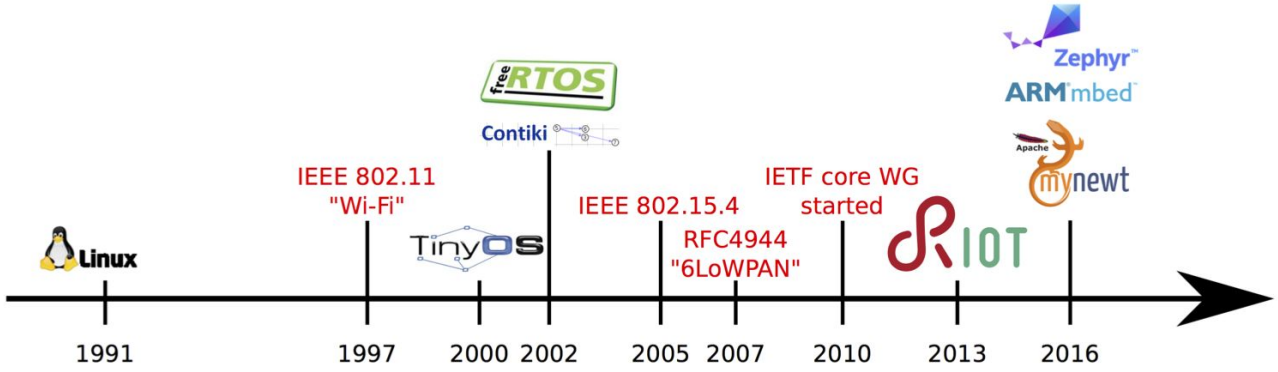
SW platform on low-end IoT devices

- The good news:
  - No need for advanced GUI (a simple shell is sufficient)
  - No need for high throughput performance (kbit/s)
  - No need to support dozens of concurrent applications

- The bad news:
  - kBytes of memory!
  - Typically no MMU!
  - Extreme energy efficency must be built in!

SW platform on low-end IoT devices

- Contiki
- RIOT
- TinyOS
- myNewt
- FreeRTOS
- mbedOS (ARM)
- Zephyr (Intel)
- LiteOS (Huawei)
- ...
- and closed source alternatives



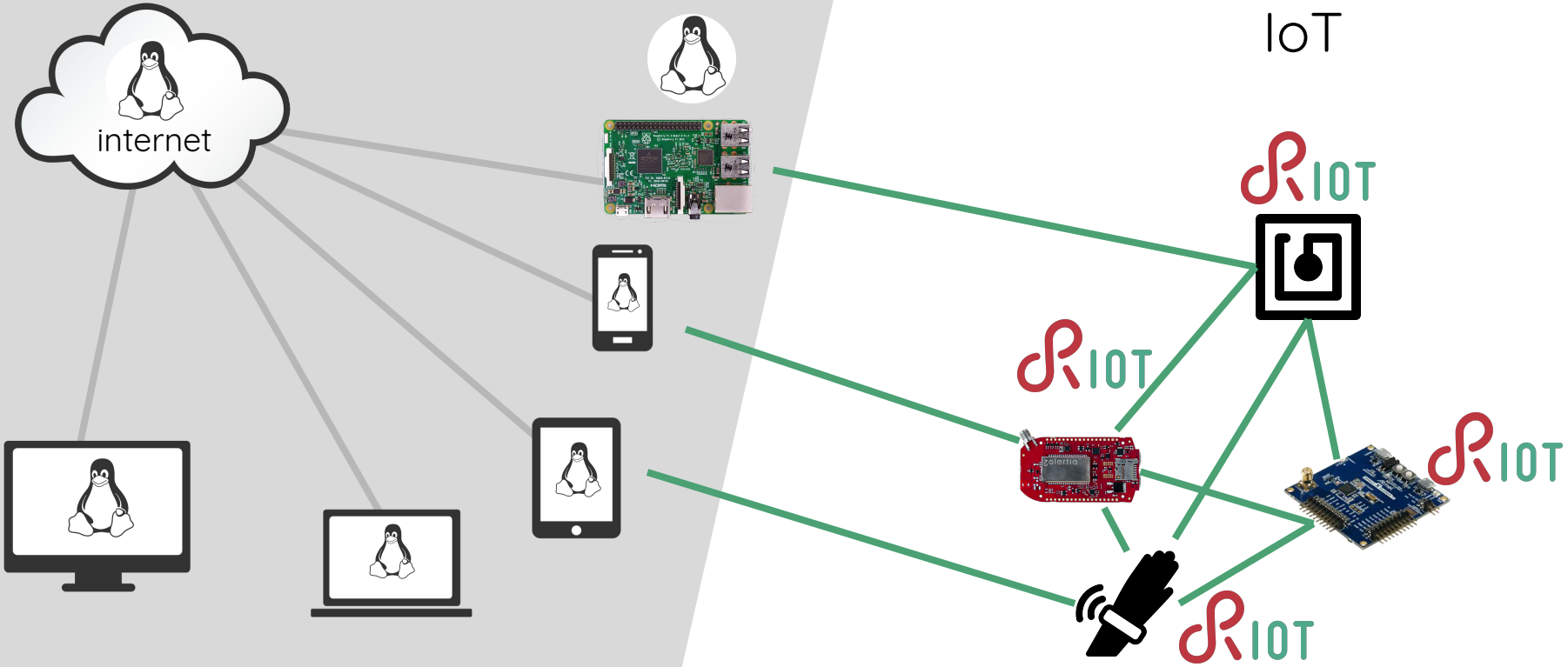
Reference: O. Hahm et al. "*Operating Systems for Low-End Devices in the Internet of Things: A survey*," IEEE Internet of Things Journal, 2016.

Why?

How?

What is RIOT?

RIOT: an OS that fits IoT devices



Meet RIOT

- Free, open source (LGPLv2.1) operating system for the IoT
  - Write your code in ANSI-C or C++
  - Providing some POSIX features like pthreads and sockets
  - No IoT hardware needed for development
    - Run & debug RIOT as native process on Linux

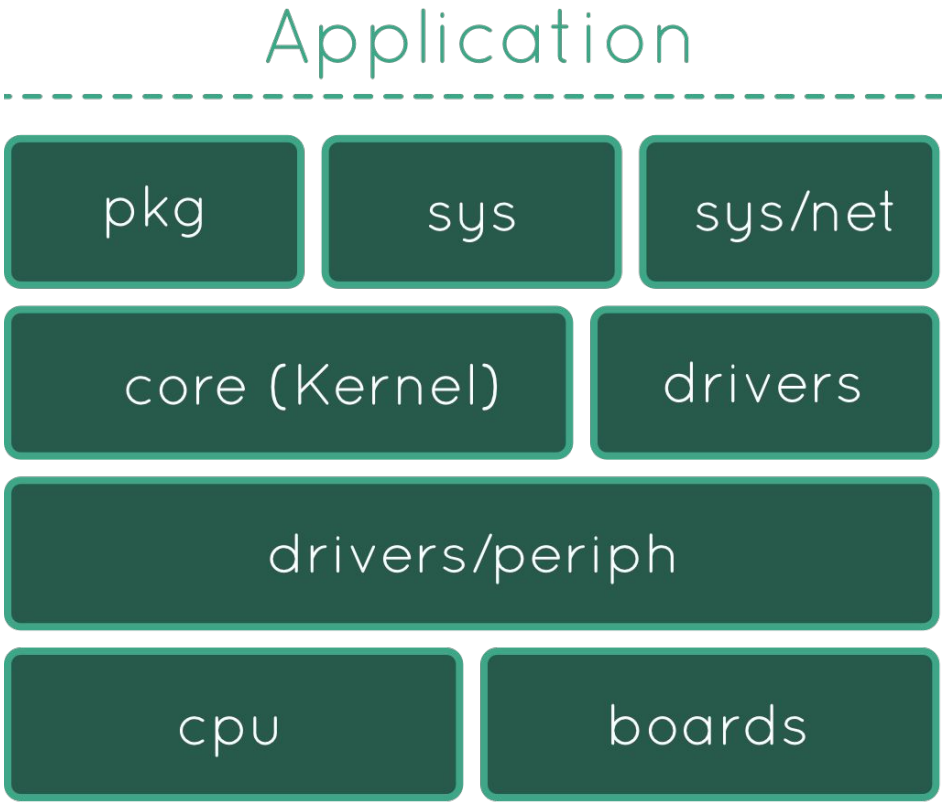
Why?
How?
What is RIOT?

- Kernel performance
- Connectivity
- Portability
- Trust

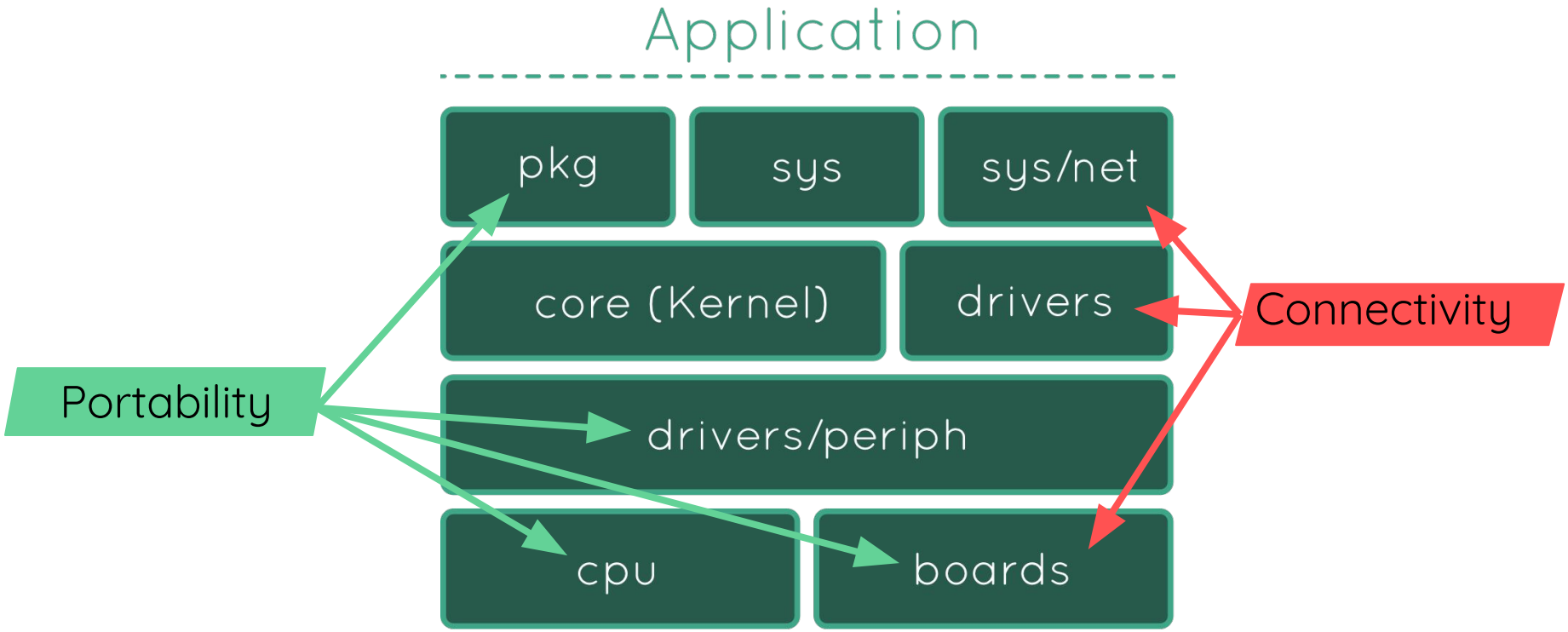
RIOT Specs

- Microkernel architecture (for **robustness**)
  - The kernel uses ~1.5K RAM on 32-bit architectures
- Tickless scheduler (for **energy-efficiency**)
- Deterministic O(1) scheduling (for **real-time**)
- Low latency interrupt handling (for **reactivity**)
- Modular structure (for **adaptivity**)
- Preemptive **multi-threading** & powerful **IPC**

Architecture



Architecture



Why?
How?
What is RIOT?

Kernel performance

Connectivity

Portability

Trust

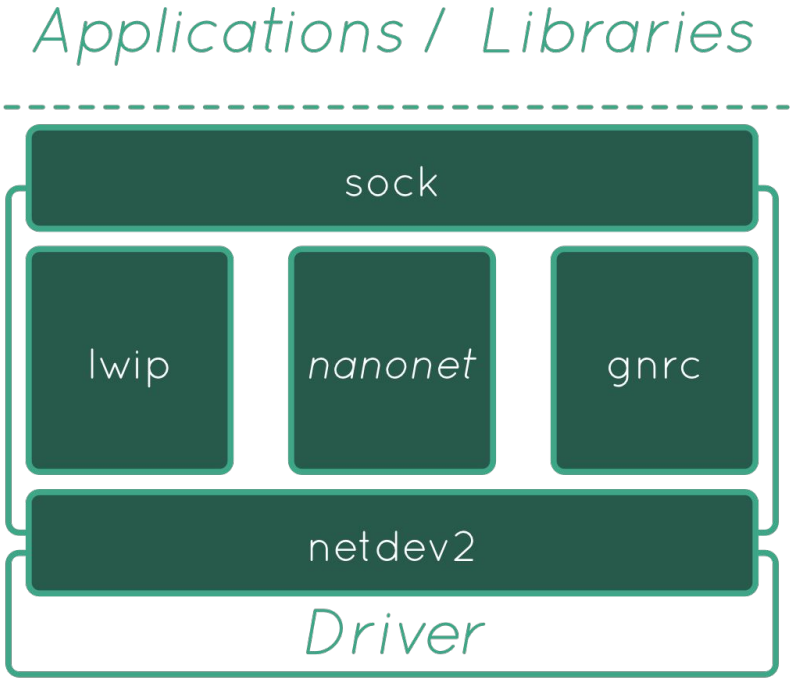
Connectivity

- Use what you need

  Flexible module based stack

  ✓ Many different PHY technologies (IEEE802.15.4, IEEE802.3, Bluetooth, NFC, serial, CAN bus)
  ✓ Interoperability tested IETF 6lo implementation
  ✓ IPv6
  ✓ UDP, TCP
  ✓ COAP, MQTT-SN (in the making)

Connectivity cont'd



- 3rd-party packages
  - lwIP stack
  - uIP (emb6) stack
  - Thread (OpenThread) stack
- Experimetal stacks
  - CCN-Lite
  - NDN-RIOT

Why?
How?
What is RIOT?

Kernel performance
Connectivity
Portability
Trust

Portability

- Code your application once & run everywhere
  - Various 32-bit platforms, but 16-bit and 8-bit platforms are supported too (ARM, x86, MSP430, MIPS, AVR…)
  - Independent from hardware vendors and their specific solutions
  - gcc standard toolchain, but llvm is usable too
- Use existing libraries
  - libcoap
  - libfixmath
  - lwip
  - micro-ecc
  - relic

Portability cont'd

- Easy porting of RIOT to new hardware
  - `periph` Interfaces
    - Porting is a matter of hours or days
    - E.g. support for new ARM Cortex-M boards is `trivial`
  - Reusable `*_common` modules
    - Reduce code duplication

sys/*

- Posix sockets, pthreads
  (use familiar concepts)
- Shell
  (interact with your board via shell, use `ps` and `ifconfig`)
- Crypto & hashes
  (aes, 3des, md5, sha1, sha256, ...)
- C++11
- Arduino
  (run your arduino sketch on RIOT)
- Cbor
- SenML

Why?
How?
What is RIOT?

- Kernel performance
- Connectivity
- Portability
- Trust

Trust

- if secured & understood,
  - IoT is positively groundbreaking
- else
  - IoT may be one of the greatest threads in human history

Privacy?

Security?

Reliability?

Some level of trust?

Combining RIOT & Linux, IoT is possible with

- End-to-end open source
- End-to-end secure & open communication standards
- From anywhere in the Internet all to the way to (low-end) IoT devices

RIOT Roots & Evolution

- 2008 - 2012
  - Ancestors of kernel stem from research projects (FireKernel, uKleos)
- 2013 - 2017
  - Branding of RIOT started, source code moved to Github, major development of the network stack and the OS as such
- Speed evolution
  - Of the codebase
  - Of the community

RIOT in Numbers

- 3690 commits in in 2016
- ~150 contributors (~30 maintainers)
- 60+ boards
- 35+ MCUs
- 25+ Sensors
- 1 RIOT Summit
- 1 RIOT Foundation

RIOT Community Work

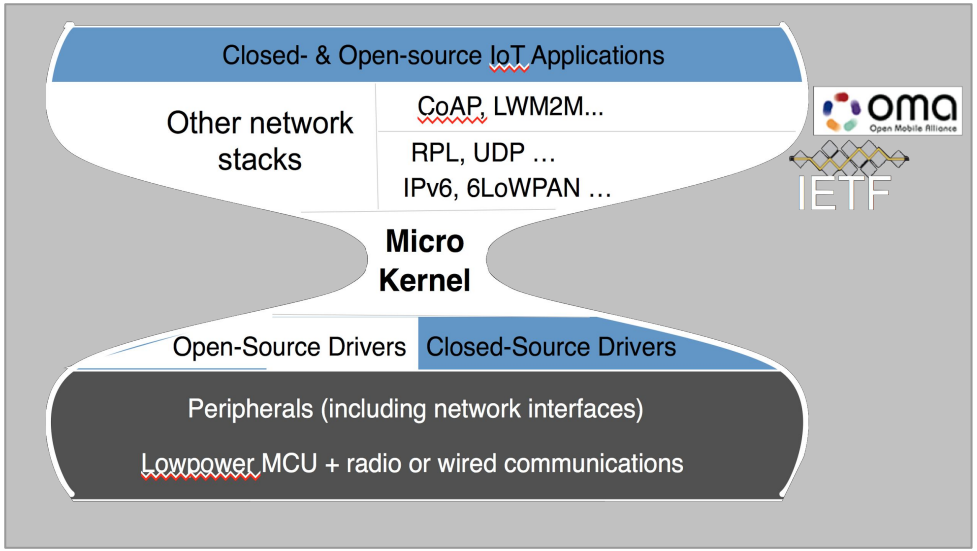
- Time based release model (3 months cycles)
- Roadmap, to help focusing on specific topics
- Task Forces (to work on specific topics)
- Open development process (github)
- Monthly Hack&Ack sessions
- Mailing lists
- IRC channel

RIOT in a nutshell

Free, open source platform for portable IoT software

RIOT offers a platform functionally equivalent to Linux, based on:

- Open source,
- Open-access protocol stacks
- Community driven development



Thanks for your interest!

News: **https://twitter.com/RIOT_OS**

For cooperation questions: **riot@riot-os.org**

For developer questions: **devel@riot-os.org**

Support & discussions on IRC: **irc.freenode.org**
#riot-os


IOT