Loosely Coupled Actor Systems

for the Internet of Things

Raphael Hiesgen
Internet Technologies Group
Hamburg University of Applied Sciences

Agenda

Introduction

Where We Are

Next Steps

Risks and Conclusion

The Internet of Things (IoT)

- Network of nodes
  - Connected through Internet standards
  - Perform machine-to-machine communication
    - Built from often constrained embedded devices
- Sensors and actuators
- Platform for distributed applications

Problem Statement

- Highly distributed applications design
- Development requires specialized knowledge
  - Communication, synchronization and scalability
    - Usually in low-level languages (such as C)
    - Error-prone & hard to debug
- Deployment is platform-specific

Approach

- Actors as base entities
  - Run concurrently & in isolation
  - Can spawn new actors
- Distributed runtime environment
  - Network transparent message passing
  - Distributed error-handling
- Network of actors as a design candidate for the IoT
  - Programm distributed applications

General Relevance

- The IoT is everywhere
  - Fitness trackers (FitBit, Health Kit, Google Fit, …)
  - Smart watches (Pebble, Android Wear, …)
  - Home automation (Home Kit, Nest, …)
  - Emerging development tools (ARM mbed [3], …)
- Number of participating devices increases

Relevance of the Research

- Ease application development
- Reduce the development overhead
- Professionalization, generalization and standardization
  - Reusability
  - Robustness
  - Portability
- Provide tools to test and deploy software

Research Question

- Can we efficiently link low-level protocols to an abstract communication between actors?

- Can we meet efficiency expectations regarding hardware resources?

- Is the actor model suitable to design and develop applications for the IoT?

- Is the actor model well suited to express typical application scenarios?

Agenda

Introduction

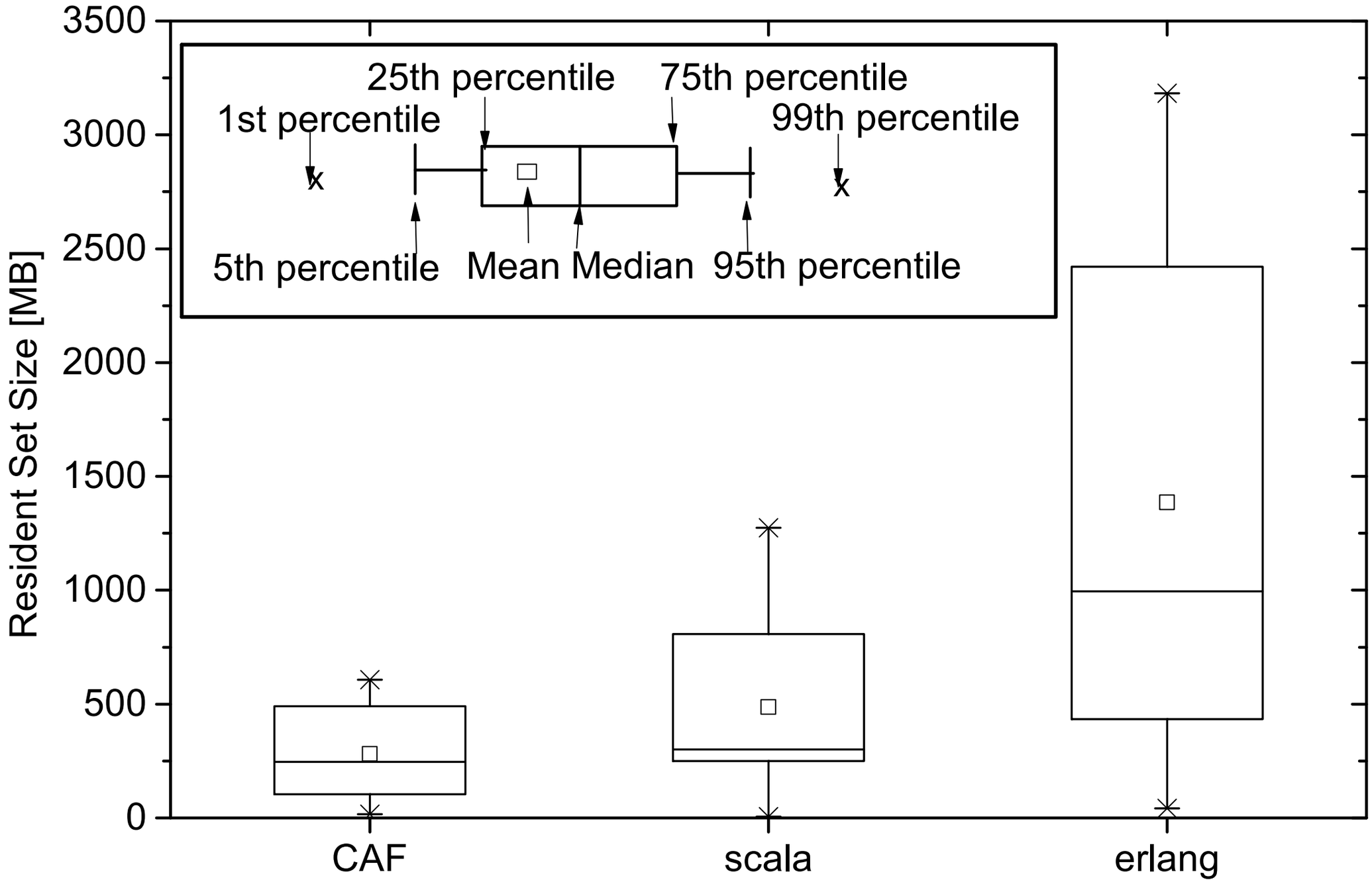
Where We Are

Next Steps

Risks and Conclusion

The C++ Actor Framework

- Open source implementation of the actor model [1]
- Native development in C++
- Small memory footprint
- Different runtime implementations
  - Memory management & scheduler
- Static type-checking
- Runtime inspection tools



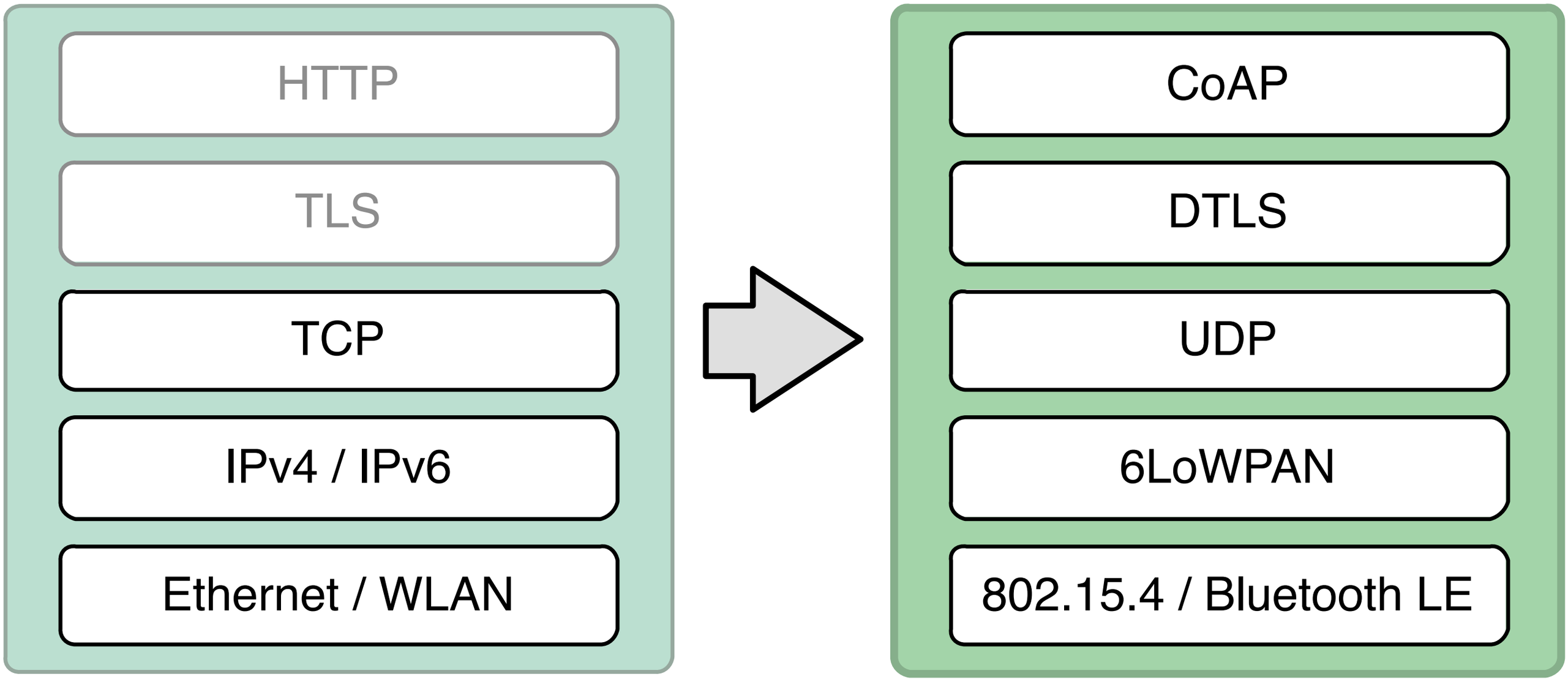
Memory Consumption

Spawning 2^{20} actors.

Adaption to the IoT

- Communication protocols
  - Lossy links are common
  - Handle infrastructure failure
- Nodes may contain private data
- Secure wireless communication
- Requires suitable messaging layer

C++ Actor Framework

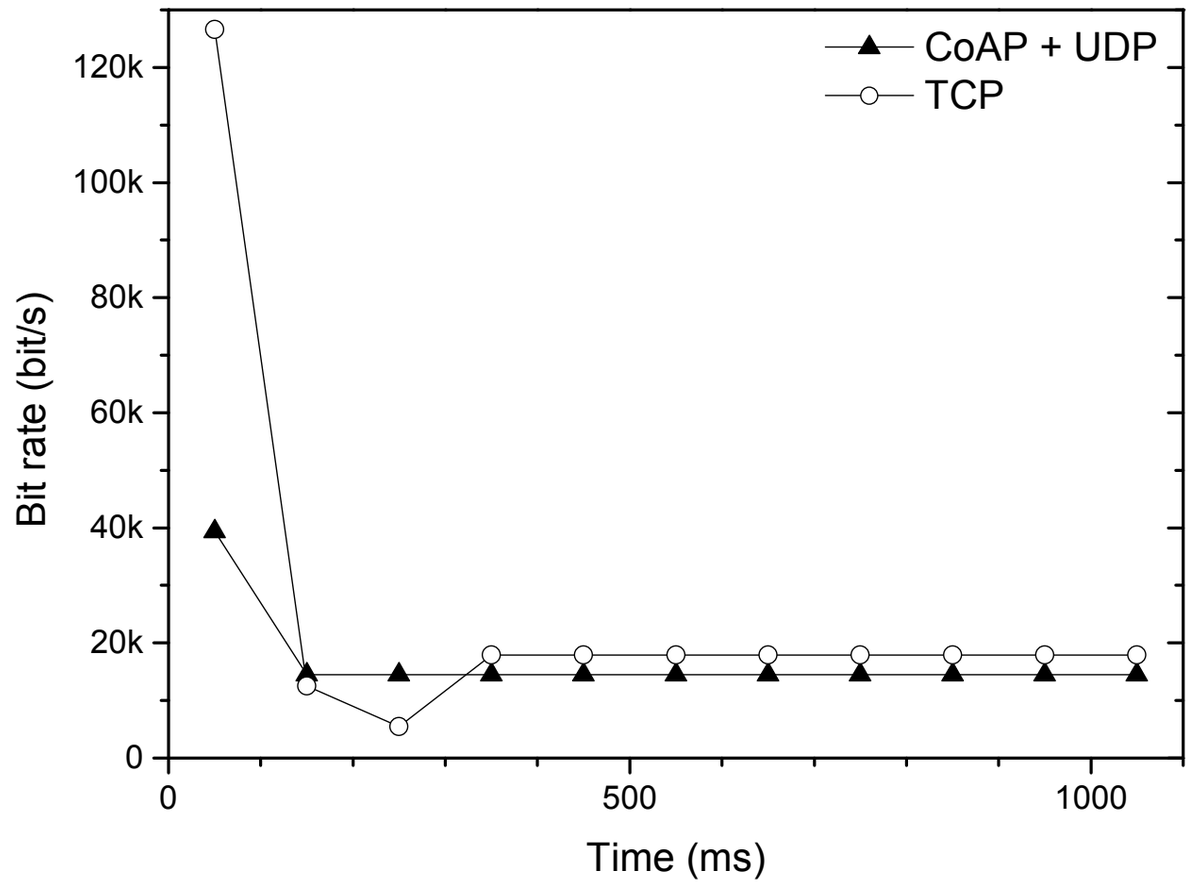


Network Stack

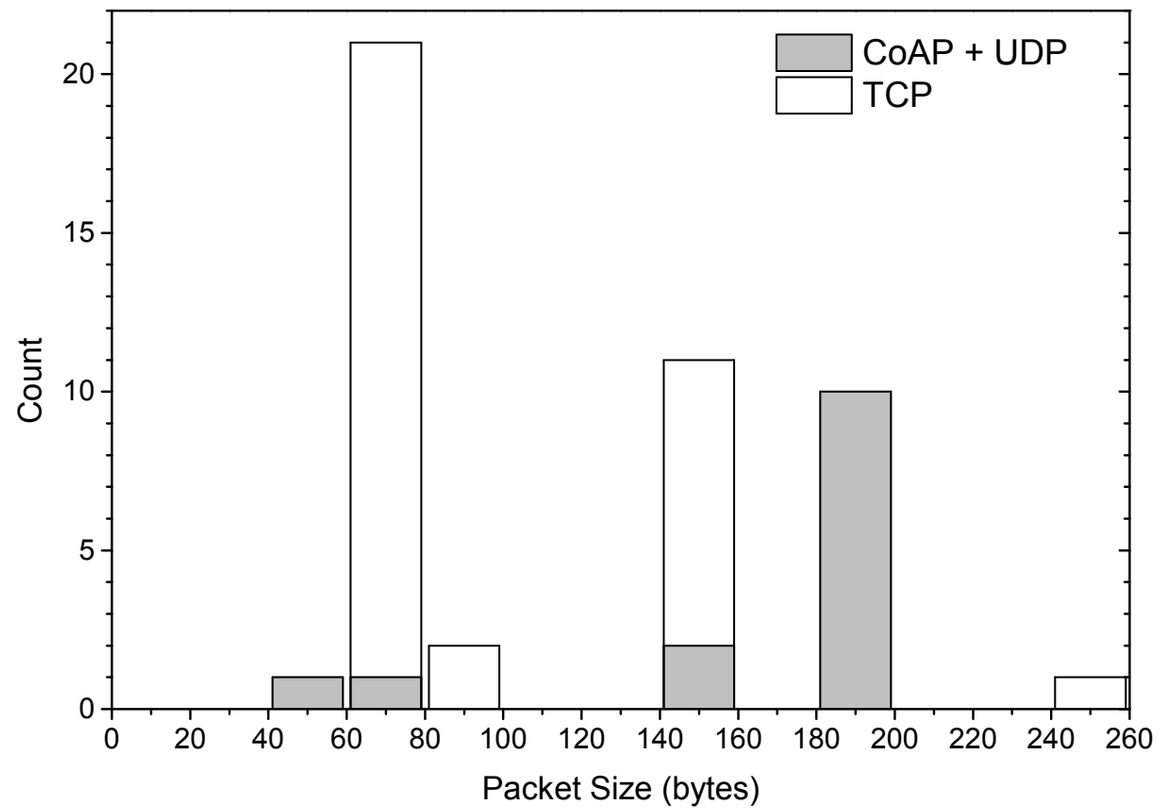
Built upon open Standards

Proof-of-Concept

- Data collection at source
- Based on Ethernet, UDP and CoAP
- Compared to TCP-based impl.



- Findings
  - Fewer packets send
  - Lower bandwidth used



Agenda

Introduction

Where We Are

Next Steps

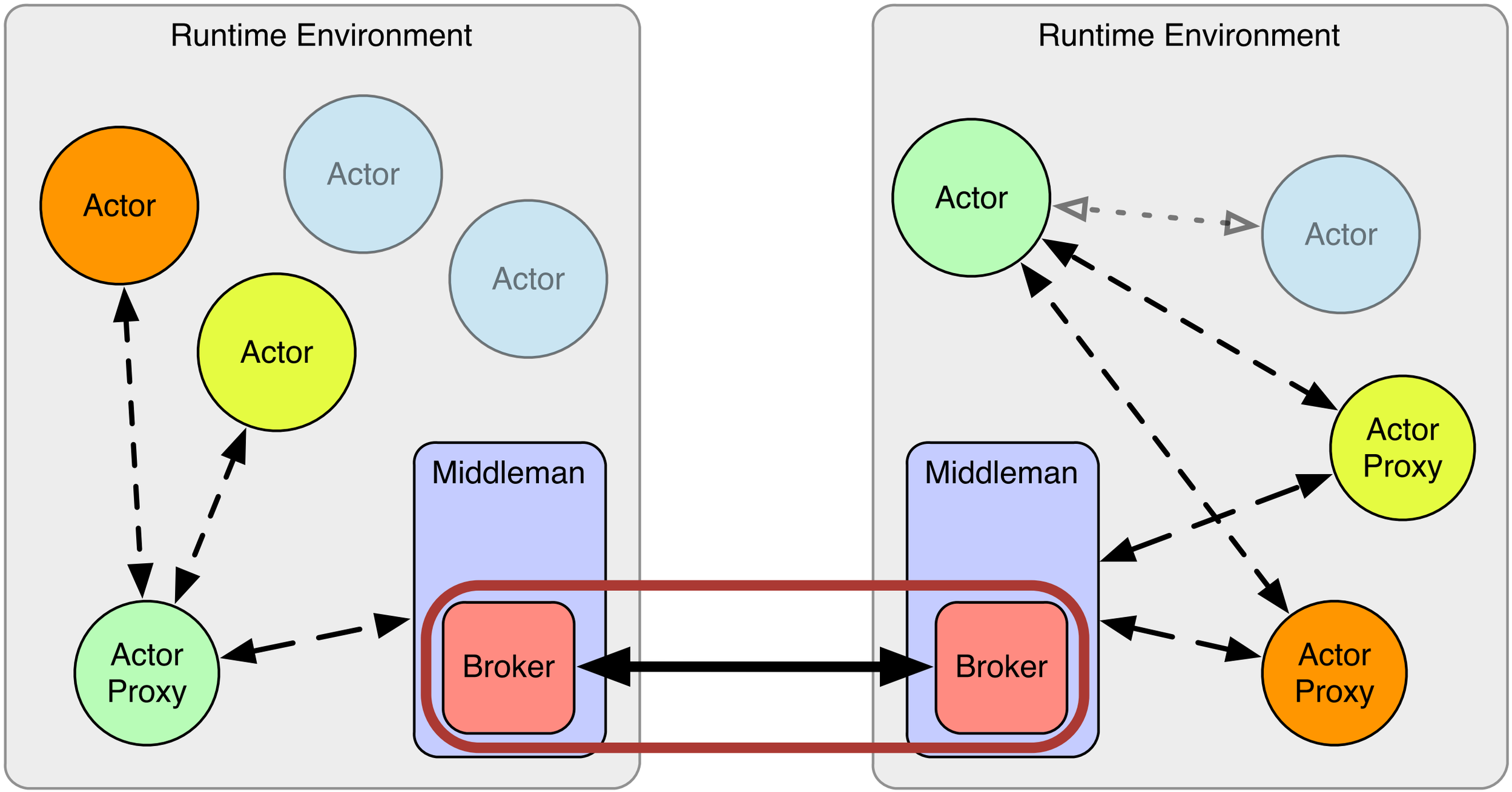
Risks and Conclusion

Transactional Layer

- 6LoWPAN
  - IPv6 compatibility
  - Header compression
- CoAP
  - Duplicate message detection
  - Reliable message transfer (transactions)
  - Fragmentation of large messages
- CAF
  - Message header compression
  - Error propagation

Security

- Authentication, authorization and encryption
- Challenges
  - Constrained power & energy
  - Nodes physically acquired
- Crypto is hard to do right



Concept (WIP)

What do we need to secure?

Support for Embedded OSs

- The friendly Operating System for the IoT [2]
- POSIX compliance
- Energy efficient
- Real-time capable
- Development in C or C++

Validation

- Protocol correctness
  - Packet loss & message sizes
- Application performance
- Reproducibility
  - Environment heavily impacts results (e.g., interference)
  - Unpredictable behavior in networks
- Requires experimentally driven design and testing

Test Environments

- From comfortable and fast to realistic and slow
- RIOT offers a native port
  - Quick tests, not a realistic environment
- Few nodes in our lab
  - 7 Raspberry Pis running Linux
  - USB dongles enable 802.15.4
  - Useful for a proof-of-concept

Raspberry Pi

Our local test-hardware.

However …

- Pis are not our target platform
  - Have lots of power and memory
  - Run Linux (like desktops)
- Only a few number of nodes
  - Link interference hardly a problem
  - Applications may have more nodes

Larger Testbeds

- FU Berlin
  - 60 nodes distributed in several rooms and floors
  - CC1100 radio chips, 868 MHz CPU
- INRIA Technology Institute in France [4]
  - Connected through RIOT and Safest
  - 2700 nodes distributed through France

Agenda

Introduction

Where We Are

Next Steps

Risks and Conclusion

Risks

- We ask too much of the hardware
  - Power & energy consumption
  - Memory usage
  - Message sizes
- Security scheme
- Community adoption (CAF & RIOT are doing well)

Conclusion

- Development in the IoT required on specialized knowledge
  - Network communication and synchronization
  - Porting software to new hardware
- The actor model abstracts over distributed systems
  - Adapt CAF to the characteristics of the IoT
    - A transactional layer built from open standards
    - Authentication, authorization and encryption
    - Support for RIOT-OS
  - Requires experimentally driven testing

References

[1] D. Charousset, R. Hiesgen, and T. C. Schmidt, “CAF - The C++ Actor Framework for Scalable and Resource-efficient Applications,” in *Proc. of the 5th ACM SIGPLAN Conf. on Systems, Programming, and Applications (SPLASH ’14), Workshop AGERE!*, New York, NY, USA: ACM, Oct. 2014.

[2] RIOT-OS., “RIOT,” www.riot-os.org, November 2014.

[3] ARM Ltd., “ARM mbed IoT Device Platform,” https://mbed.org, November 2014.

[4] INRIA, “FIT/IoT-LAB,” https://www.iot-lab.info, November 2014.