Loosely Coupled Communication in Actor Systems

AW2 - Raphael Hiesgen

- Introduction
- Paper 1
- Paper 2
- Paper 3
- Next Steps

Introduction

- Loosely Coupled Communication
  - Handle unreliable connections
  - Non-hierarchical error-propagation model
  - Implement secure communication
  - Transparent breach of NATs and firewalls
- Use-cases
  - Internet of Things (IoT) (Project 1)
  - Internet-wide Systems

- Introduction
- Paper 1
- Paper 2
- Paper 3
- Next Steps

Why is the Web Loosely Coupled?
A Multi-Faceted Metric for Service Design

- C. Pautasso and E. Wilde
- ‘Loose coupling’ is often quoted as desirable
  - Impact of change is limited
  - Services can evolve independently
- Specific definition is missing

Origins

- First appeared in 1967
- Software engineering
    - Principle of modularity
    - Affects evolution of a system
- Distributed systems
    - Shared memory vs. message passing
    - Publish / subscribe paradigm

Facets

- Discovery
  - Central registration vs. decentralized referral
  - Web uses search engines
- Interaction
  - Synchronous vs. asynchronous
- Interface Orientation
  - Horizontal (API) vs. vertical (protocol)

Facets

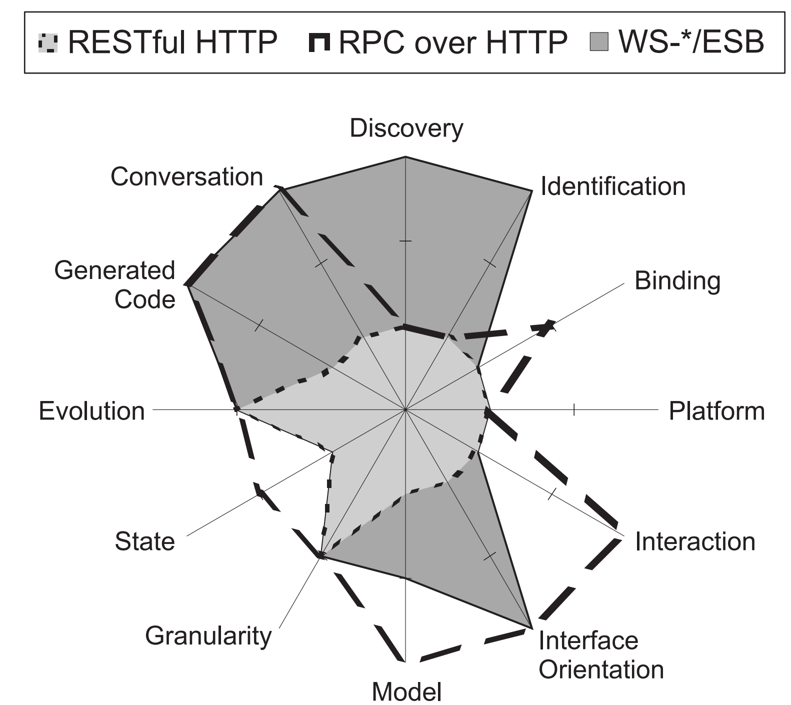
- Model
  - Specified data model vs. self contained messages
- State
  - Requires management (establishment, recovery, …)
  - Stateless design keeps ‘state’ in messages
- Conversation
  - Predefined exchange vs. dynamic discovery

Facets

- Identification
  - Central identification services vs. specified identification scheme
- Binding
  - Resolving names into identifiers
- Platform
  - Programming language requirement, …
- Granularity
  - coarse-grained vs. fine-granular interfaces
- Evolution
  - compatibility vs. fragmentation
- Generated Code
  - Code needs to be regenerated if the description changes

Analysis

	RESTful HTTP	RPC over HTTP
Discovery	Referral	Referral
Identification	Global	Global
Binding	Late	Early/Late
Platform	Independent	Independent
Interaction	Asynchronous	Synchronous
Interface Orientation	Vertical	Horizontal
Model	Self-Describing Messages	Shared Model
Granularity	Fine/Coarse	Fine/Coarse
State	Stateless	Stateless/Shared, Stateful
Evolution	Compatible/Breaking	Compatible/Breaking
Generated Code	None/Dynamic	Static
Conversation	Reflective	Explicit



Coupling in Web services [1].

- Introduction
- Paper 1
- Paper 2
- Paper 3
- Next Steps

Constrained Application Protocol (CoAP)

- Developed by the IETF (currently a draft)
- Designed for M2M communication
- Request-response model adapted from HTTP
- Works asynchronously over UDP
- Implements reliable messages
  - ‘Confirmable’ message answered with ‘ACK’

Congestion Control in Reliable CoAP

- A. Betzler, C. Gomez, I.Demirkol and J. Paradells
- Limited hardware and link capacities
- Basic CoAP vs. CoCoA
- Performance with parallel transactions

Basic CoAP vs CoCoA

- Retransmission after timeout
  - Lossy links
  - Congestion
  - Long processing
- Default interval [2s, 3s]
- Counteract congestion:
  - Binary exponential back-off timer

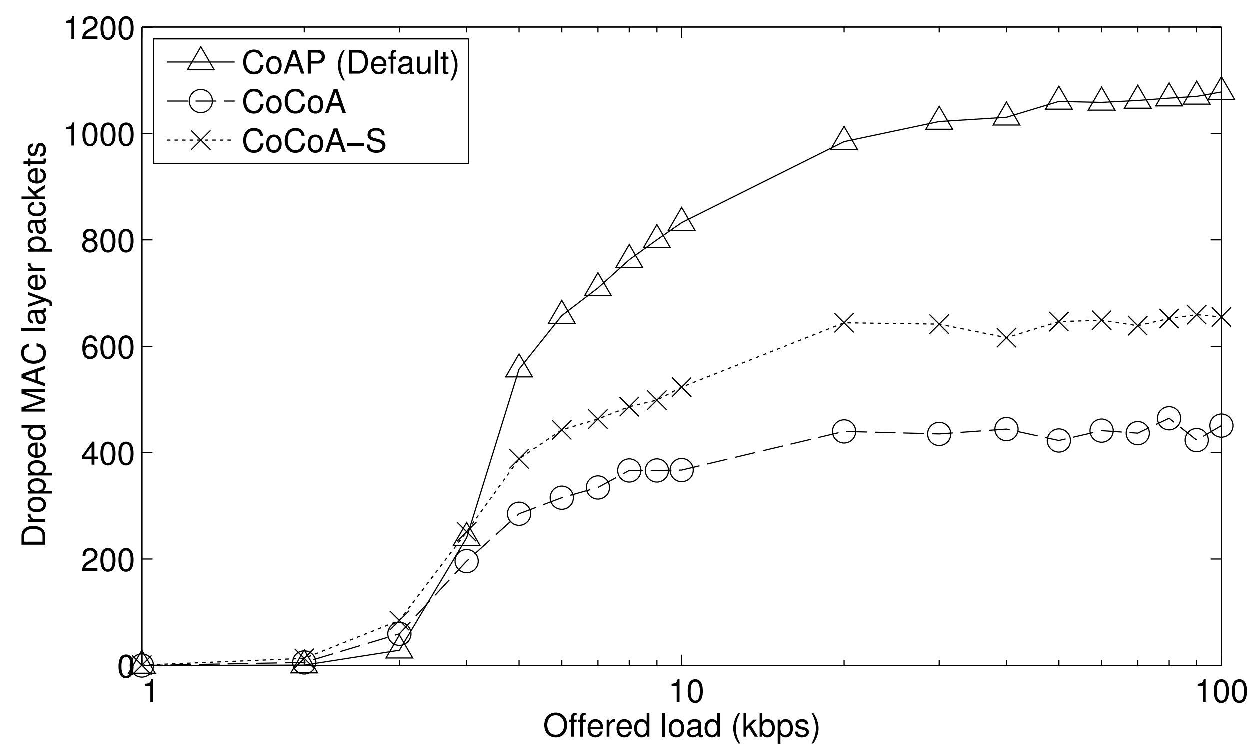
- Two separate timeout values
- RTT until ACK is received
- Estimators
  - Strong: no retransmission
  - Weak: retransmission
- Weighted averages (init: 2s)
- Third approach uses the strong estimator (CoCoA-S)

Parallel Transactions

- Defined through NSTART (default = 1)
- Parallel transactions lead to higher congestion
- Overhead through additional state
- Examined for four parallel transactions

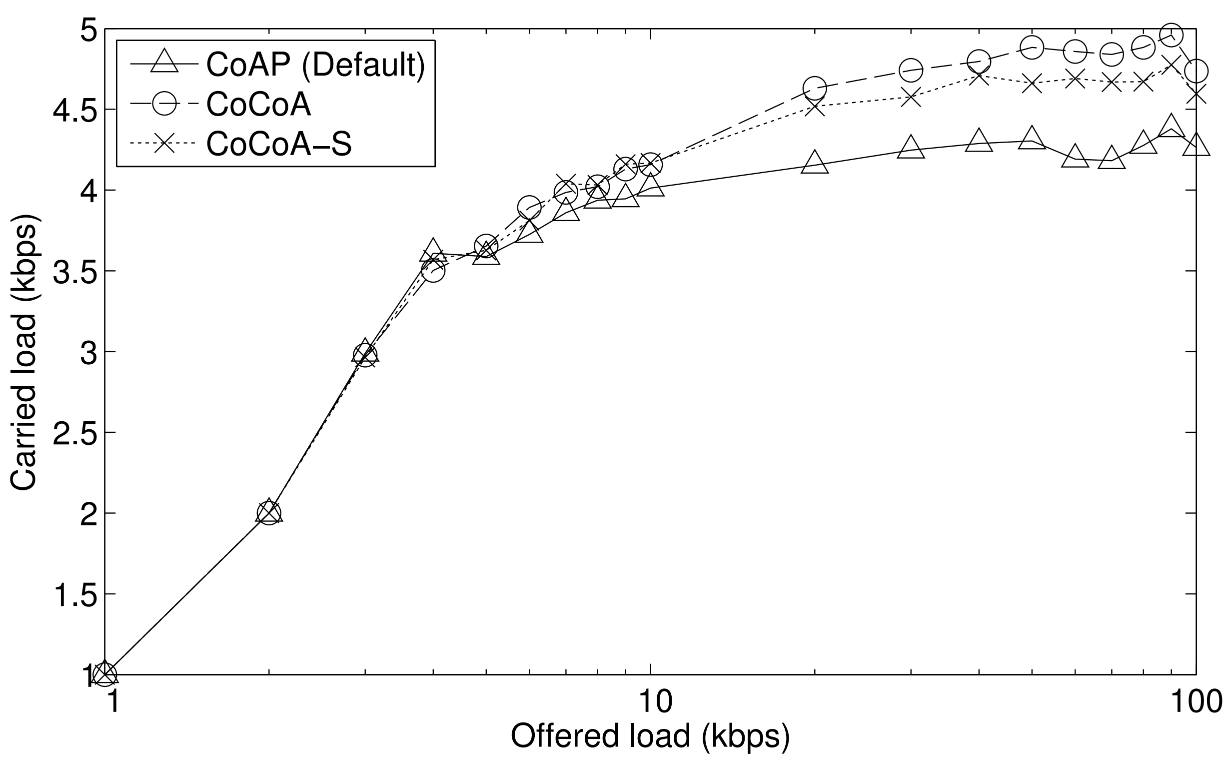
Analysis

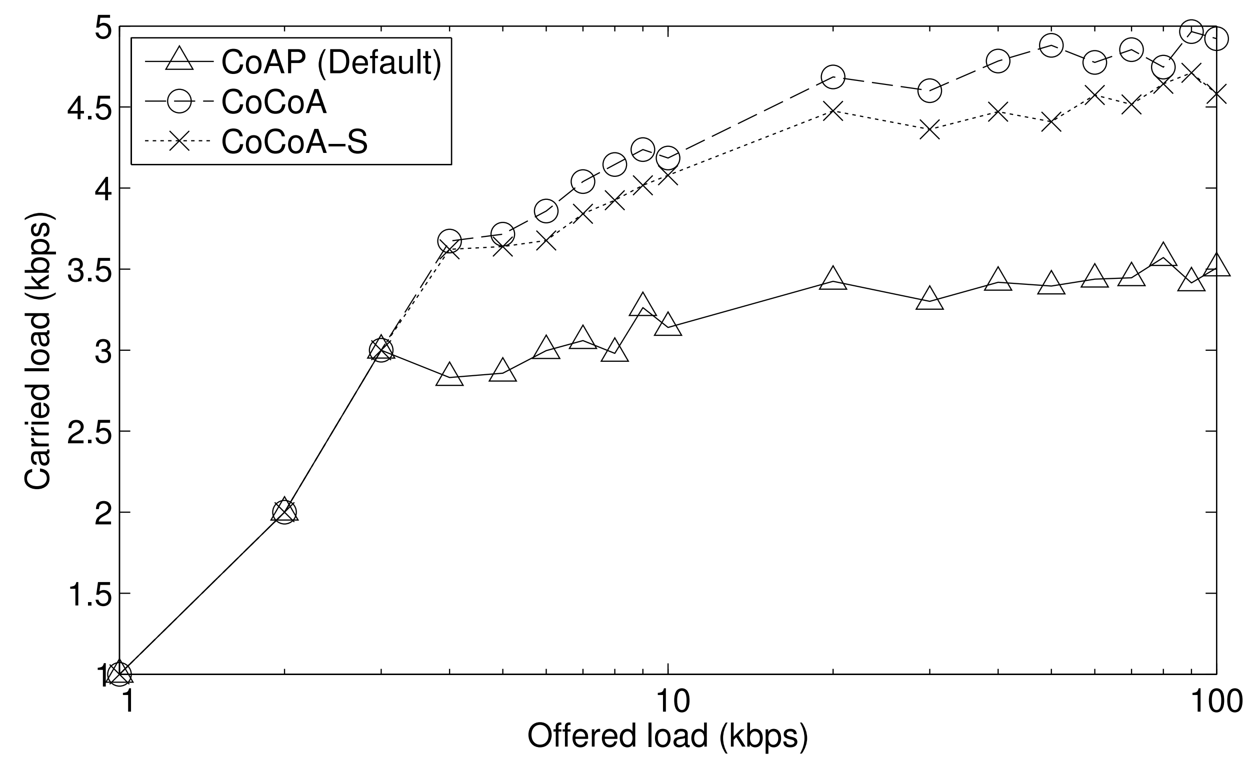
- Stack: 802.15.4, 6LoWPAN, UDP and CoAP
- RPL Routing
- Different topologies
  - Chain, grid and dumbbell
  - Influenced by routing characteristics



Dropped MAC layer packets in the chain topology [2].

Throughput





Achieved throughput in the chain topology,
NSTART=1 (left) and NSTART=4 (right) [2].

- Introduction
- Paper 1
- Paper 2
- Paper 3
- Next Steps

Drop the Phone and Talk to the Physical World: Programming the Internet of Things with Erlang

- A. Sivieri, L. Mottola and G. Cugalo
- Most embedded systems are developed in low-level languages, such as C
- Leaves a lot of responsibility to the developer
- Difficult to test, maintain and port
- Solution: a high-level programming model for the IoT

Erlang

- Actor-like concurrency model (masking distribution)
- Functional core (dynamic typing, pattern-matching)
- Embedded system support (pattern-matching on bit streams)
- Code can be hot-swapped

ELIoT

- Library for constrained, distributed environments
    - Many-to-many syntax, not based on TCP
- Interpreter without unnecessary features
    - Smaller memory requirements (few MB)
- Simulator to validate implementation
    - Transparent migration to real hardware

Analysis

- Implementation of three routing protocols
  - Flooding, Tickle and CTP
  - 62 simulated devices and 2 real ones
- Compare lines of code to TinyOS and Contiki

Algorithm	TinyOS	Contiki	ELIoT
Opportunistic flooder	495	187	100
Trickle	219	194	61
CTP	2169	1470	303

Lines of code comparison [3].

- Few lines of code for complex protocols

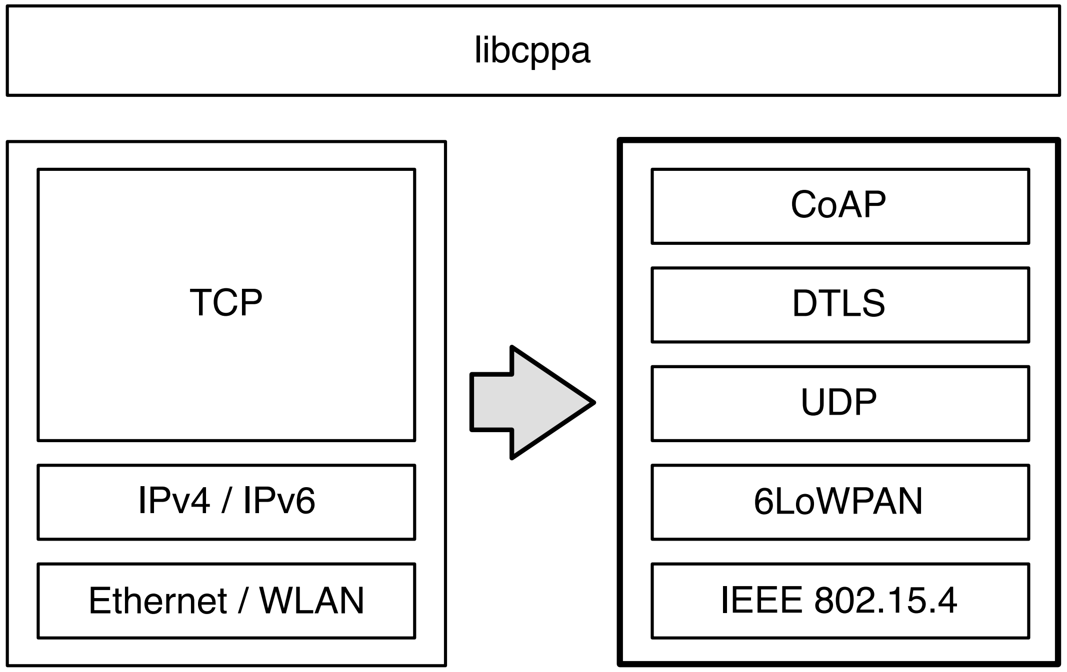
Further Questions

- Paper does not present the network stack
  - Why is message passing limited to a single-hop?
- Code has not been published
- Author is still active!

- Introduction
- Paper 1
- Paper 2
- Paper 3
- Next Steps

Next Steps

- Implement a network stack
  - Transaction based message passing
  - Use protocols for the IoT
- Future work
  - Setup test environment
  - Address Internet-wide systems (HTTP, NATs, …)
  - Encryption and authentication



Adapting the network stack of libcppa to the IoT.

Thank you!

Questions?

References

[1] Pautasso, Cesare and Wilde, Erik (2009). *Why is the Web Loosely Coupled?: A Multi-faceted Metric for Service Design*. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 911–920, New York, NY, USA. ACM.

[2] Betzler, August and Gomez, Carles and Demirkol, Ilker and Paradells, Josep (2013). *Congestion Control in Reliable CoAP Communication*. In Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems, MSWiM '13, pages 365–372, New York, NY, USA. ACM.

[3] Sivieri, A. and Mottola, L. and Cugola, G. (2012). *Drop the phone and talk to the physical world: Programming the internet of things with Erlang*. In Software Engineering for Sensor Network Applications (SESENA), 2012 Third International Workshop on, pages 8–14.

[4] Shelby, Z., Hartke, K., and Bormann, C. (2013). *Constrained Application Protocol (CoAP)*. Internet-Draft – work in progress 18, IETF.

[5] Bormann, C. (2014). *CoAP Simple Congestion Control/Advanced*. Internet-Draft – work in progress 01, IETF.