Advanced Internet and IoT Technologies

- Introduction to the Web of Things -

Prof. Dr. Thomas Schmidt

http://inet.haw-hamburg.de | t.schmidt@haw-hamburg.de

Agenda

Why do we need a Web of Things?
What are its main objectives?

Information processing in the Web of Things

A Data-centric Web of Things

Device management in the Web of Things

MOTIVATION

IoT

Internet of Things and the Web of Things

Things are proxies for physical, real-world objects on the Internet

WoT

Assessing the things via standard Web technologies

Internet of Things and The Web of Things

Things are proxies for physical, real-world objects on the Internet

Web of Things Objectives

Web of Things is an application layer for the IoT – with the main objectives:

- Discovery
- Identification
- Integration into platforms
- Interpretation of information
- Interoperability across platforms
- Security and privacy

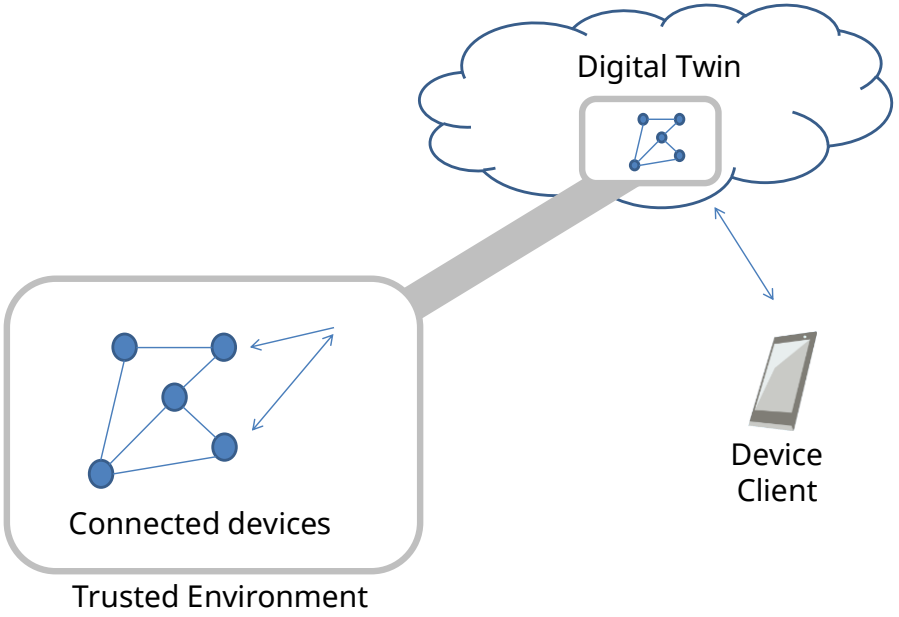
Use Case: Digital Twins

Virtual representation of a (group of) devices on an edge or cloud server

Useful to simulate new configurations or services prior to deployment

Usable proxy, even if real-world devices are offline

Accessible from the public Internet, even if the devices are not



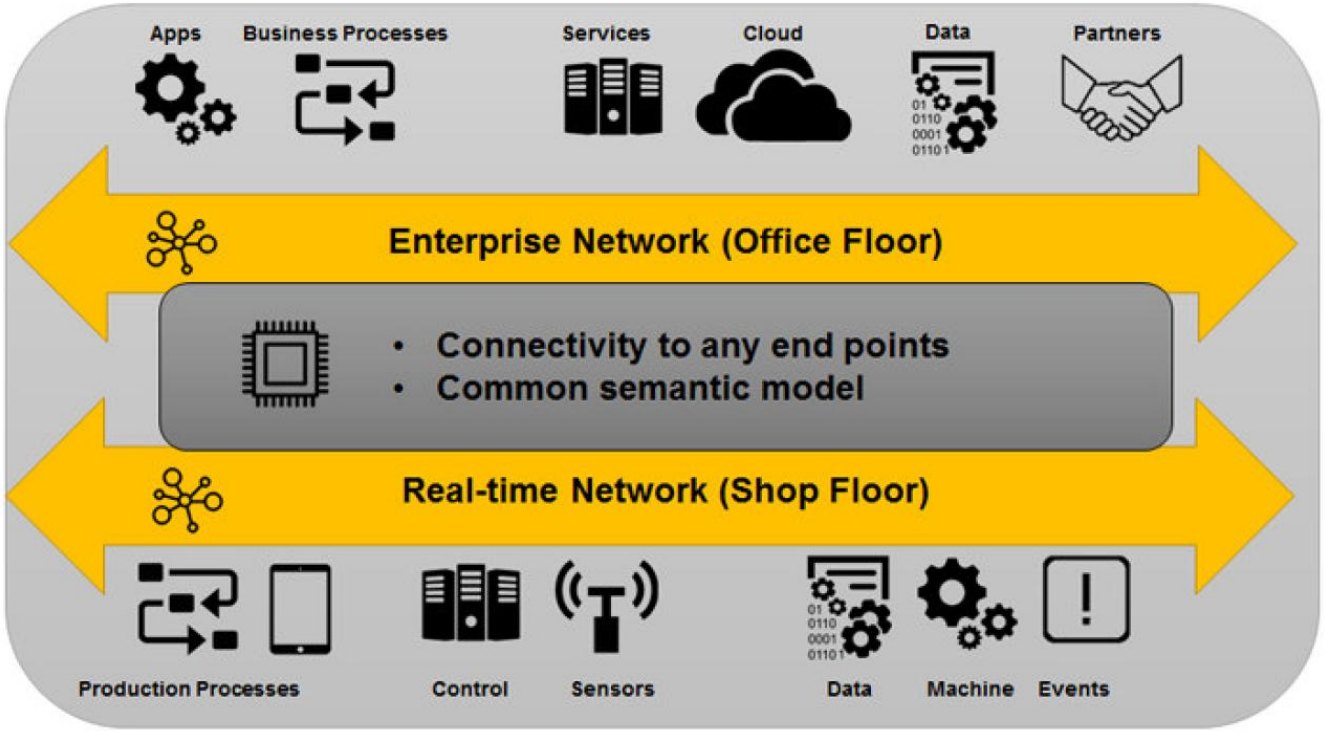
Source: https://www.w3.org/TR/wot-architecture/images/wot-use-cases/digital-twin.svg

Integration Use Case: Industrie 4.0

IT - Information Technology Level

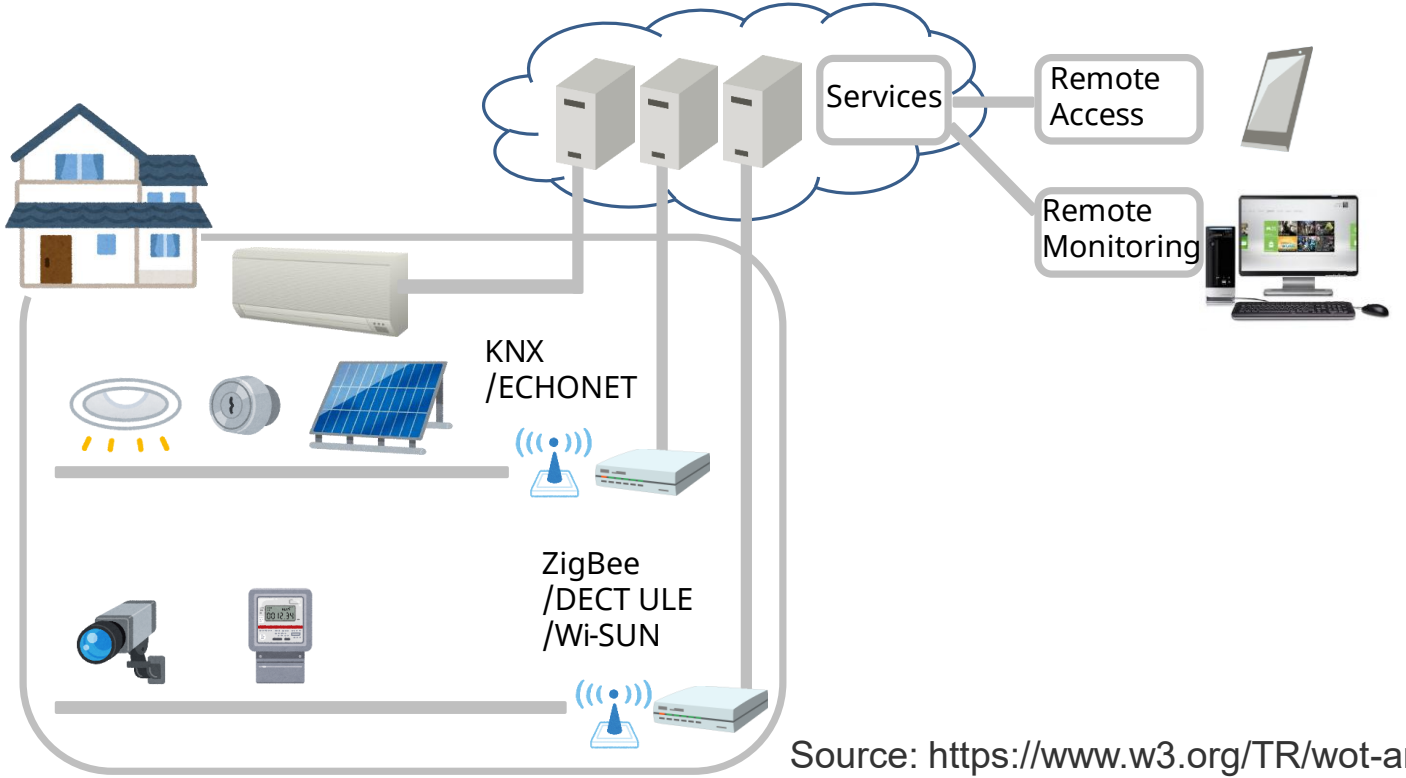
Integration

OT - Operational Technology Level



Source: Luca Foschini

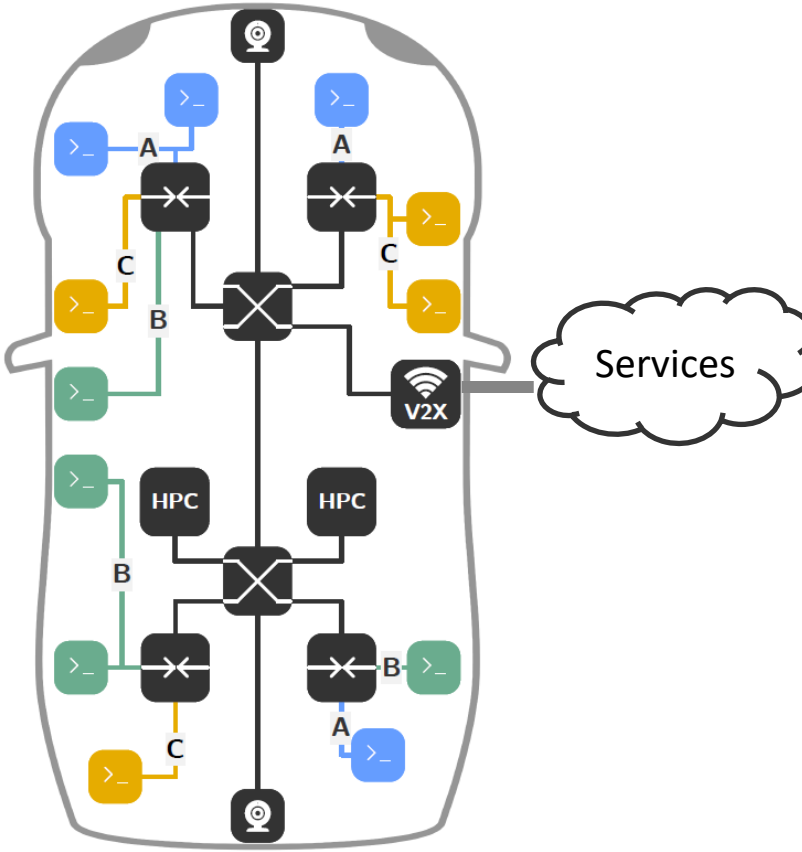
Use Case: Smart Home



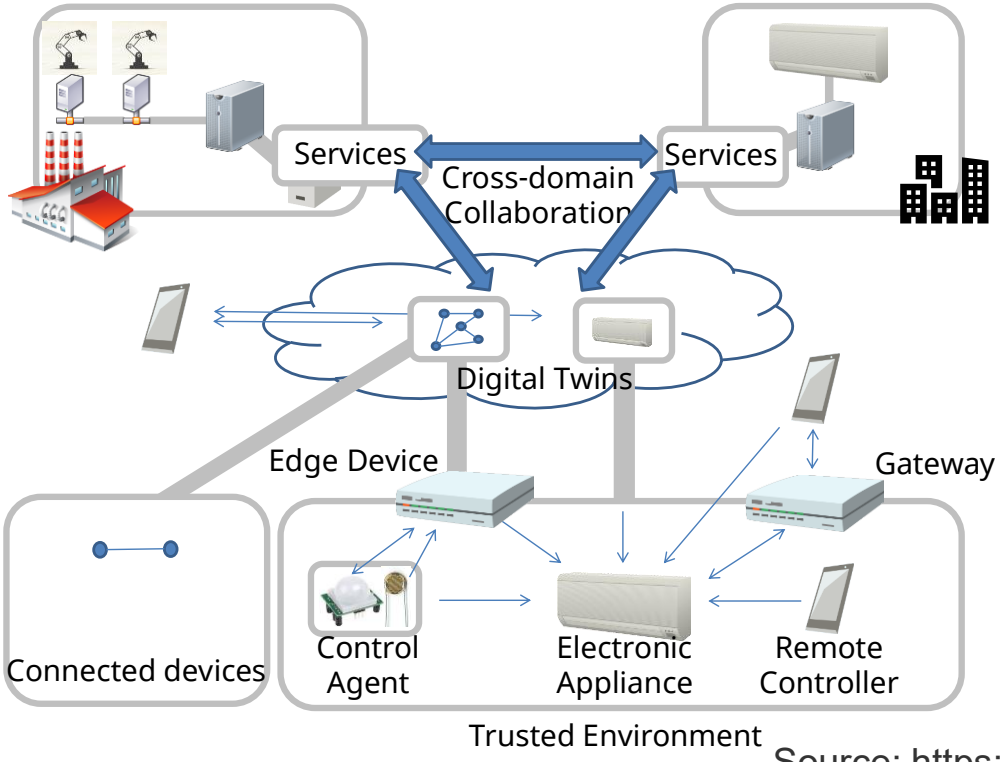
Source: https://www.w3.org/TR/wot-architecture/

Use Case: Smart Connected Car

- IEEE 802.1Q TSN –
  Time Sensitive Networking
- SOME/IP –
  IPv4+DiffServ/UDP/App
- Gateway to specialized
  cloud services



Cross-domain Collaboration



Source: https://www.w3.org/TR/wot-architecture/

WoT Requirements

Interoperability - must be possible to connect a WoT enabled device with a cloud service from different manufacturers out of the box

Compatibility – must bridge between existing and developing IoT solutions including upwards compatibility with current standards

Flexibility – shall cover a wide variety of device configurations and IoT implementations

Scalability - must scale for IoT solutions that incorporate thousands to millions of devices

WoT Requirements

The WoT shall enable mutual interworking of different IoT eco-systems using Web technologies and RESTful APIs

Interoperability - must be possible to connect a WoT enabled device with a cloud service from different manufacturers out of the box

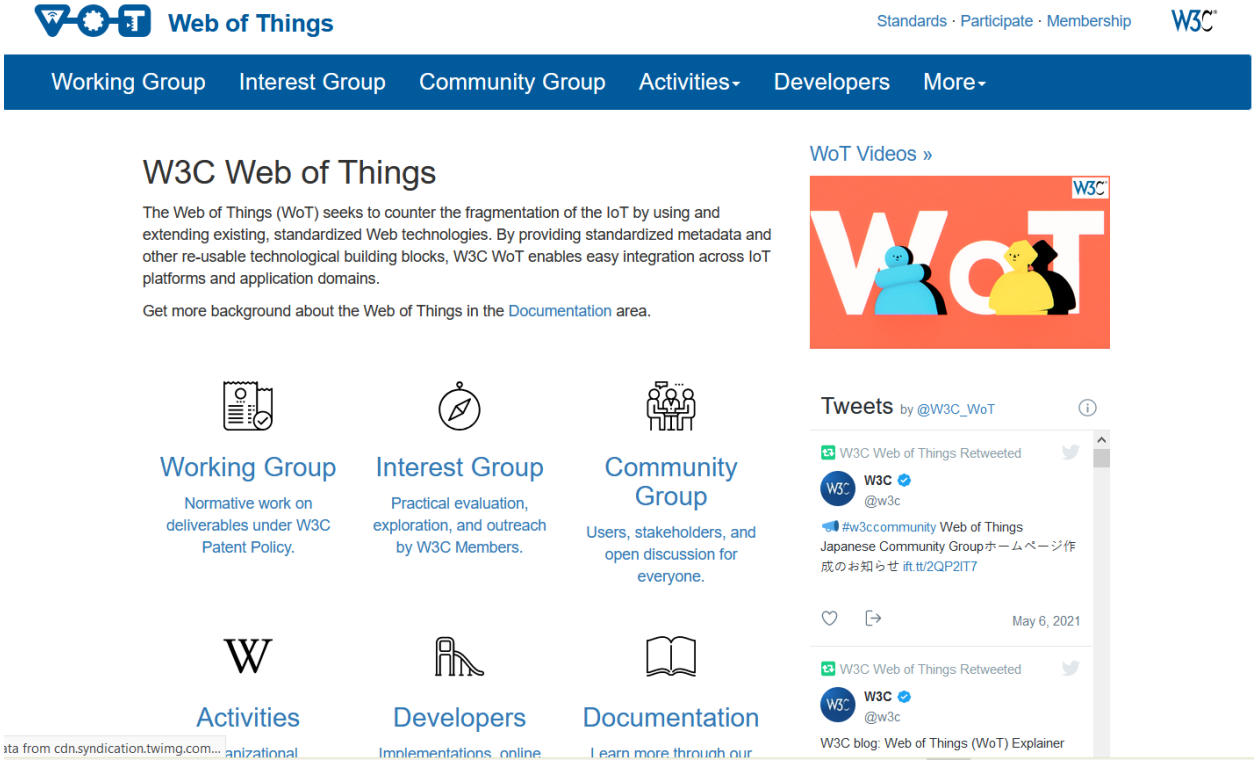
Compatibility – must bridge between existing and developing IoT solutions including upwards compatibility with current standards

Flexibility – shall cover a wide variety of device configurations and IoT implementations

Scalability - must scale for IoT solutions that incorporate thousands to millions of devices

W3C Initiative

https://www.w3.org/WoT/



HAW HAMBURG

INFORMATION PROCESSING

WoT Information

Information is addressable by URIs in the WoT

Unlock IoT fragmentation by describing

- Properties: Values, configurations, results
- Actions: Operations to perform
- Events: Triggered state changes

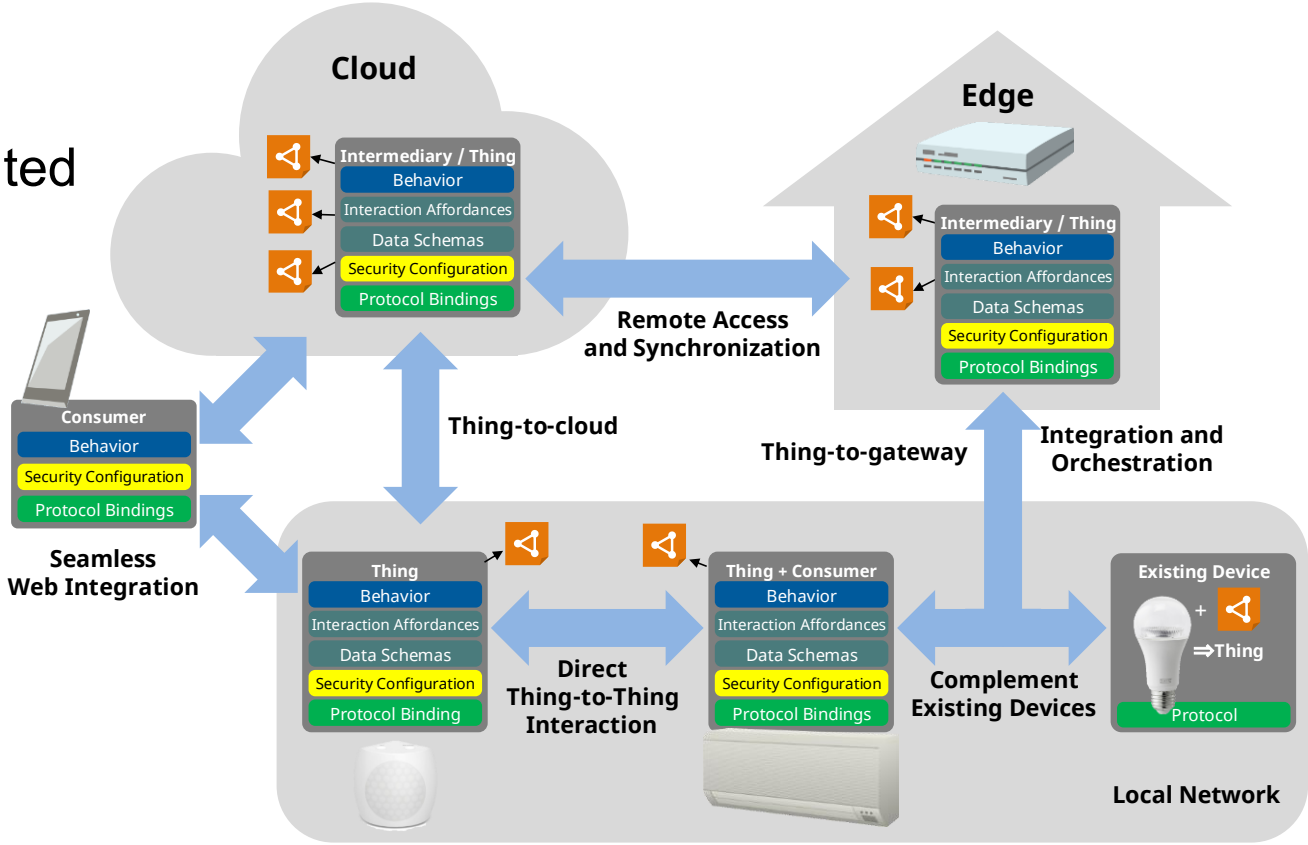
Information is structured in

- Description of Things
- Protocol bindings
- Scripting API
- Security and Privacy

WoT Abstract Architecture

Entities are represented by processable WoT Things Descriptions

This enables various integration patterns s. a. Thing-to-Thing, Thing-to-Gateway, Thing-to-Cloud, etc.

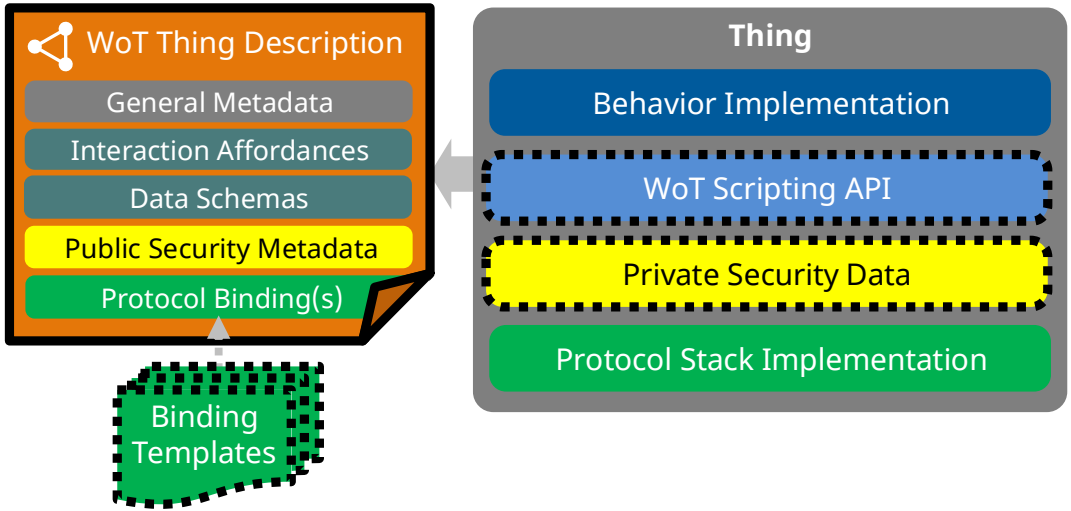


WoT Relationship with a Thing

Things Descriptions (TD)

- Semantic information model: Meta-data for things, human & machine understandable
- Domain-specific vocabulary required (not specified)
- JSON serialization

TD is the “index.html” of things in the WoT



Source: https://www.w3.org/TR/wot-architecture/

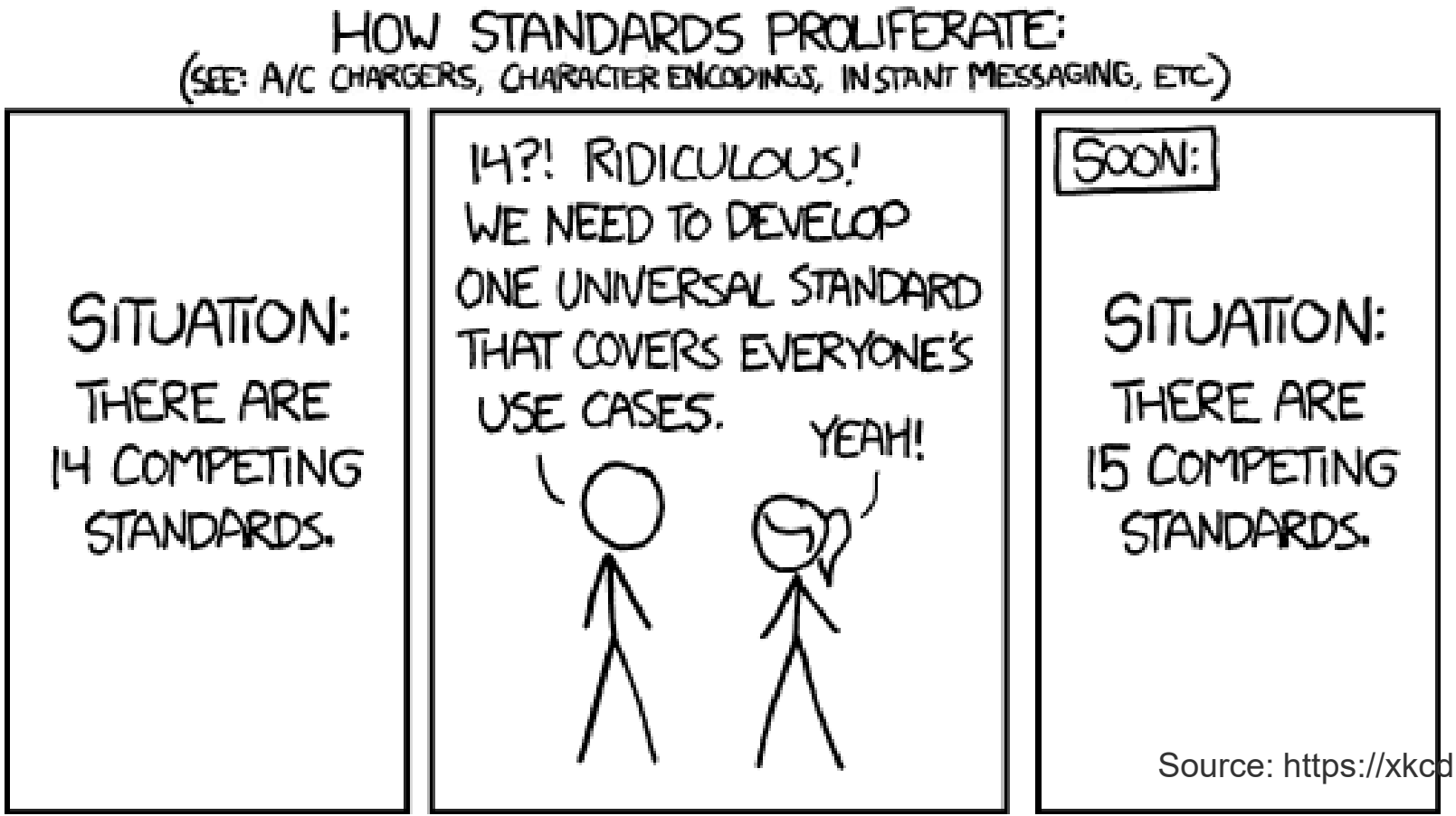
Data Models and Encodings

Description models of specific domains

Various models from diverse bodies:

- SenML (RFC 8428)
- OCF (various appliances)
- OMA (sensor devices)
- IPSO (smart sensors)
- Bluetooth (smart sensors & lighting)
- Zigbee (energy systems & sensors)
- ...

Generic encodings: JSON, CBOR, XML



Source: https://xkcd.com/

One Data Model



Problem: lack of common IoT data models

Goal: arrive at a common set of data and interaction models that describe IoT devices.

Liaison initiative between IoT organizations

- Grounds on semantic exchange techniques
- Evaluate candidate data models
- Select a single model per function
- Defines a single application data model

Semantic Definition Format for Data and Interactions of Things



Working Group ASDF liaises with OneDM

- Creates Semantic Definition Format (SDF)
- Objectives:
  - Common language to write DMs
  - Focus on modeling structure and semantics of data for exchange
  - Serialization left to existing standards
  - Practical focus on tooling: converters and serializers

Semantic Definition Format (SDF)

Format for creating and maintaining domain specific data and interaction models

Defines Objects, their associated Interactions (e.g., Events, Actions, Properties) and data types

Language definition in JSON

- Binding to CBOR/CDDL (RFC8610)
- Further bindings optional

IETF WG ASDF: RFC9880

SDF Example: Switch

```
"sdfObject": {
   "Switch": {
     "sdfProperty": {
          "value": {
            "description": "The state of the switch; false for off and true for on."
            "type": "boolean“          } ..   },
      "sdfAction": {
           "on": {
              "description": "Turn the switch on; equivalent to setting value to true."
           },
           "off": {
              "description": "Turn the switch off; equivalent to setting value to false."
           },
           "toggle": {
              "description": "Toggle the switch; equivalent to setting value to its complement.“       } ... }
```

Current Roadmap of the Community

ASDF Working Group

- Model links and relationships between objects
- Modeling digital twins
- Protocol mappings for gateways to non-IP world
- Compact notation of SDF

OneDM Initiative

- Identify relevant models and description languages of IoT standards groups
- Create converter tools to bring models to SDF
- Create serializer tools to support relevant encodings

HAW HAMBURG

DATA-CENTRIC WOT

How to Best Access Content in the WoT?

Problems with End-to-End data delivery

- Constrained devices shielded by gateways
- Transcoding gateways break E2E security
- Multi-hop forwarding in lossy regimes
- Changing paths by link flux and mobility

Alternative transport concepts

- Information-centric data replication
- WoT relies on REST access by CoAP

Lessons Learned from Information Centric Networking

Performance Boosts from 10 Years of Research

Adaptive Forwarding

In-network Caching

Content Object Security

Adaptive forwarding and **caching** shorten request paths and reduce link traversals on retransmissions

Content object security enables end-to-end security and reduces session management complexity

Lessons Learned from Information Centric Networking

Performance Boosts from 10 Years of Research

Adaptive Forwarding	In-network Caching	Content Object Security
	CoAP Proxy	OSCORE

Adaptive forwarding and **caching** shorten request paths and reduce link traversals on retransmissions

Content object security enables end-to-end security and reduces session management complexity

Smart & Resilient Network Layer

- Hop-wise Data Replication
- Content Object Security
- Adaptive Forwarding
- In-network Caching
- Asynchronous Multi-Fanout
- RESTful Access with CoAP

Smart & Resilient Network Layer

Data-Centric Web of Things

- Hop-wise Data Replication
- Content Object Security
- Adaptive Forwarding
- In-network Caching
- Asynchronous Multi-Fanout
- RESTful Access with CoAP

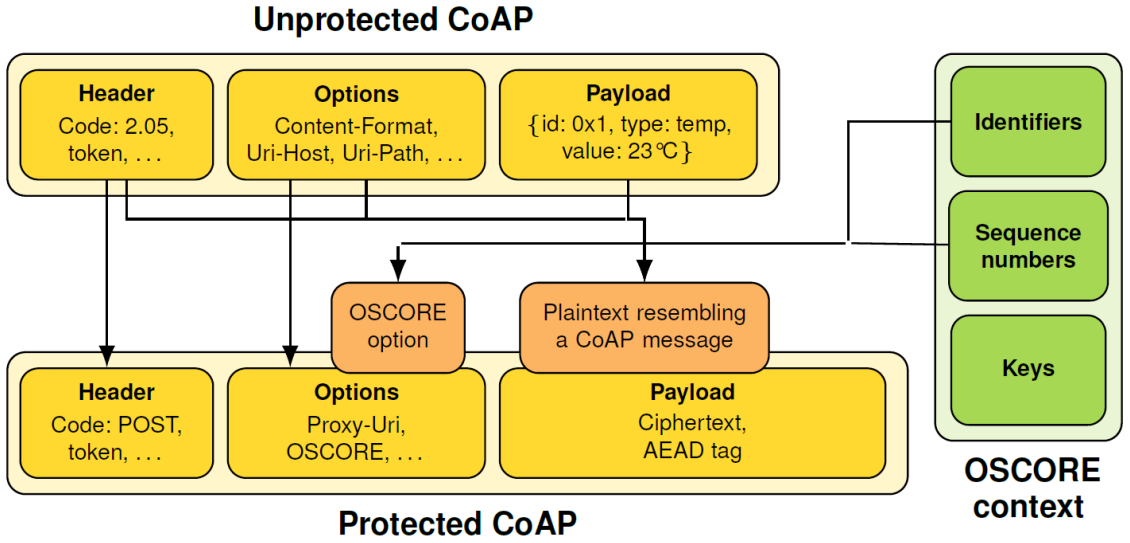
Object Security for Constrained RESTful Environments (OSCORE – RFC 8613)

Making IoT Content Cacheable

OSCORE protects CoAP messages providing integrity, authenticity, and confidentiality on an object level

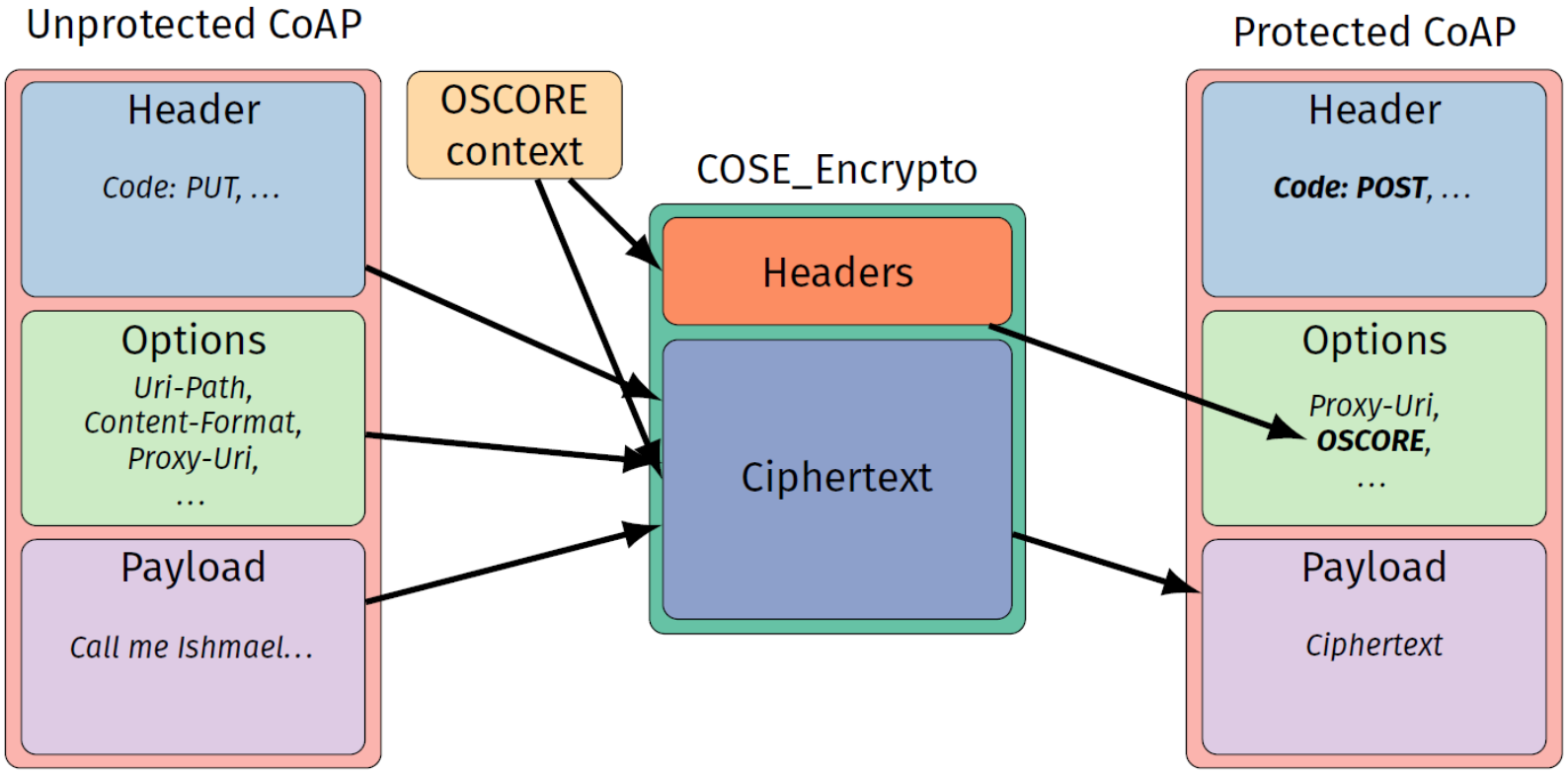
CoAP messages are encapsulated as an authenticated and encrypted COSE object

OSCORE makes its objects transport-agnostics and is able to secure multicast messages



Caching: [draft-ietf-core-cacheable-oscore]

OSCORE Integration in CoAP



Forwarding OSCORE Content Objects w/ Proxies

Cacheability

- Strong response binding prevents cache hits for subsequent requests
- Use retransmission caches to recover messages of same transaction

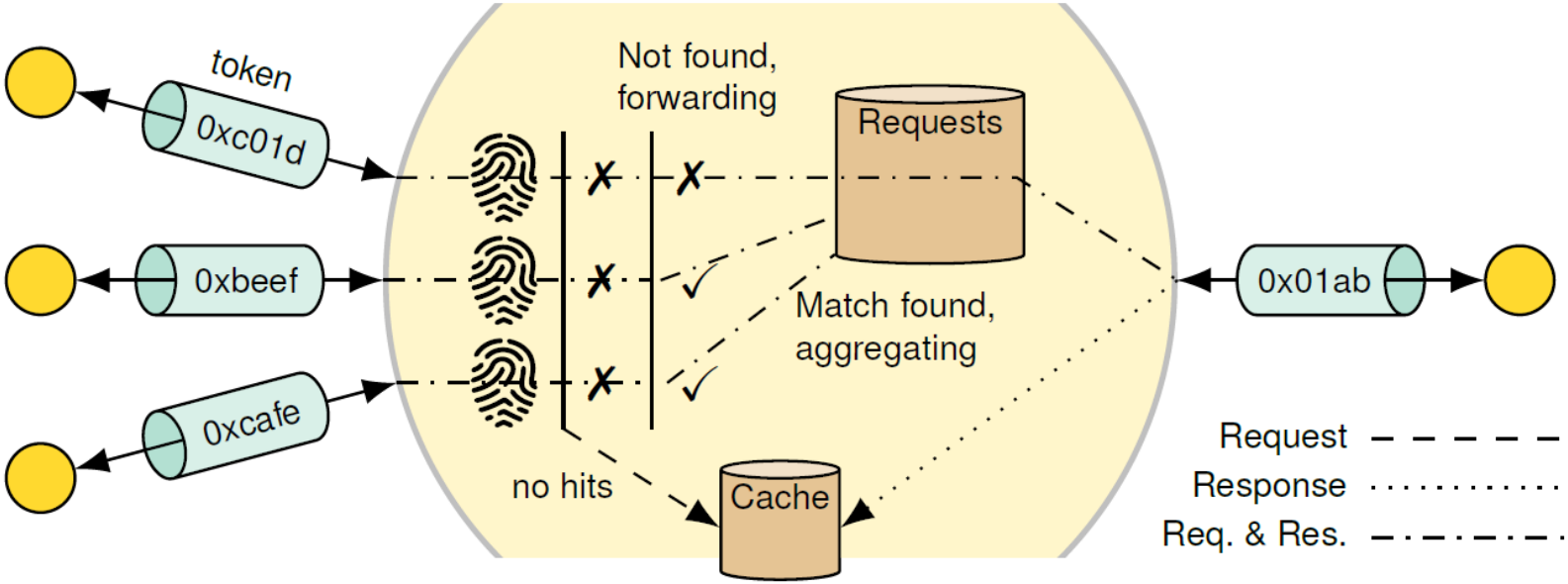
Proxy on each forwarding node

- OSCORE Objects cached
- Hop-wise message timeout
- Retransmissions on each forwarder

Decoupling of data from location

- Link-local IP addressing
- Forwarding via resource name

Forwarding and Caching with CoAP Proxy



Constructing a Data-Centric Web of Things

Communication Model & Flow Control

- CoAP GET method provides request-response paradigm
- Acknowledgments for requests and optionally for responses

Adaptive Forwarding & Caching

- CoAP proxies forward requests and build reverse path
- Proxies cache incoming responses

Content Object Security

- OSCORE provides authenticated encryption
- End-to-end security persists across gateways

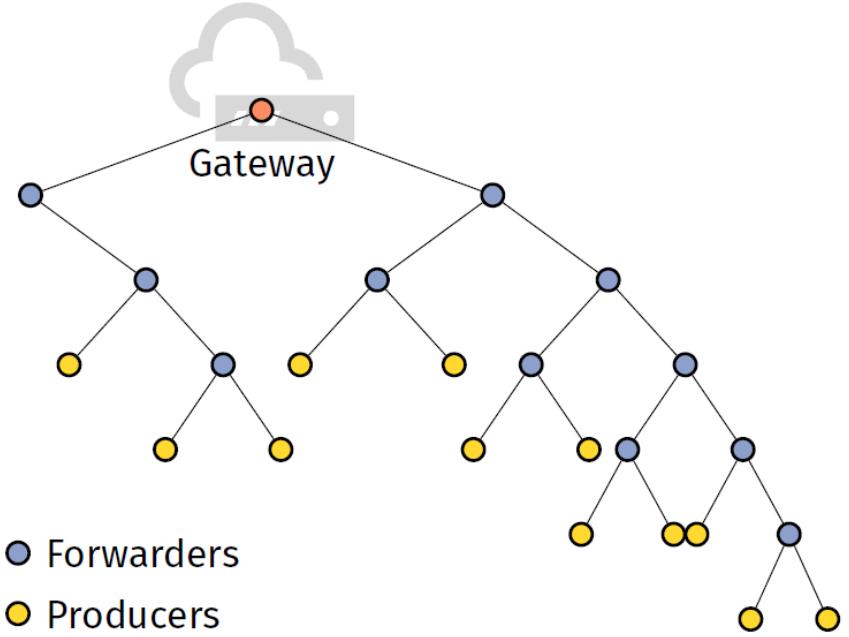
Performance Evaluation in a Testbed

Hardware M3 node in IoT Lab testbed, IEEE 802.15.4

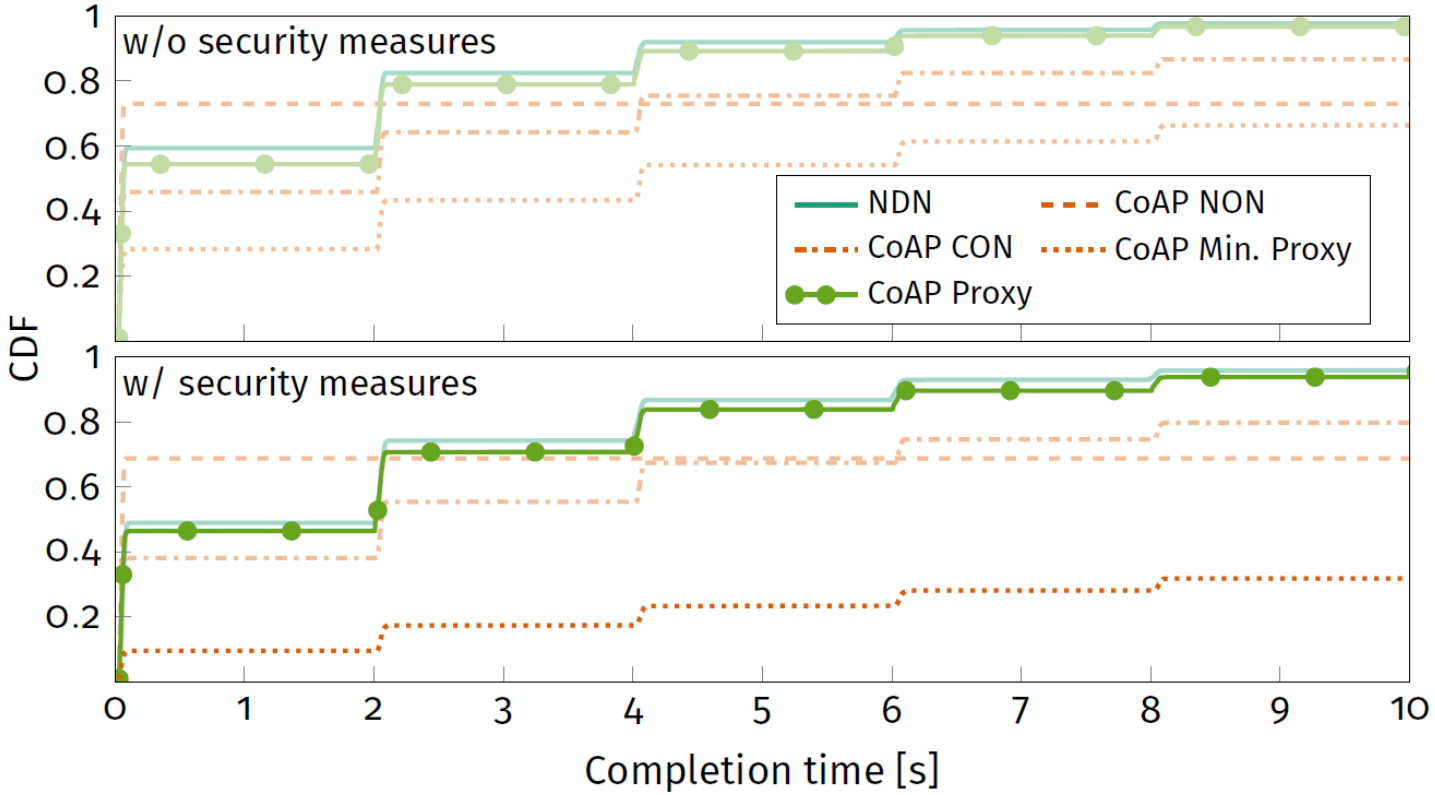
Software RIOT

Topology 12 producers, 11 forwarders

Scenario Gateway requests 2-byte temperature every ≈ 1 s

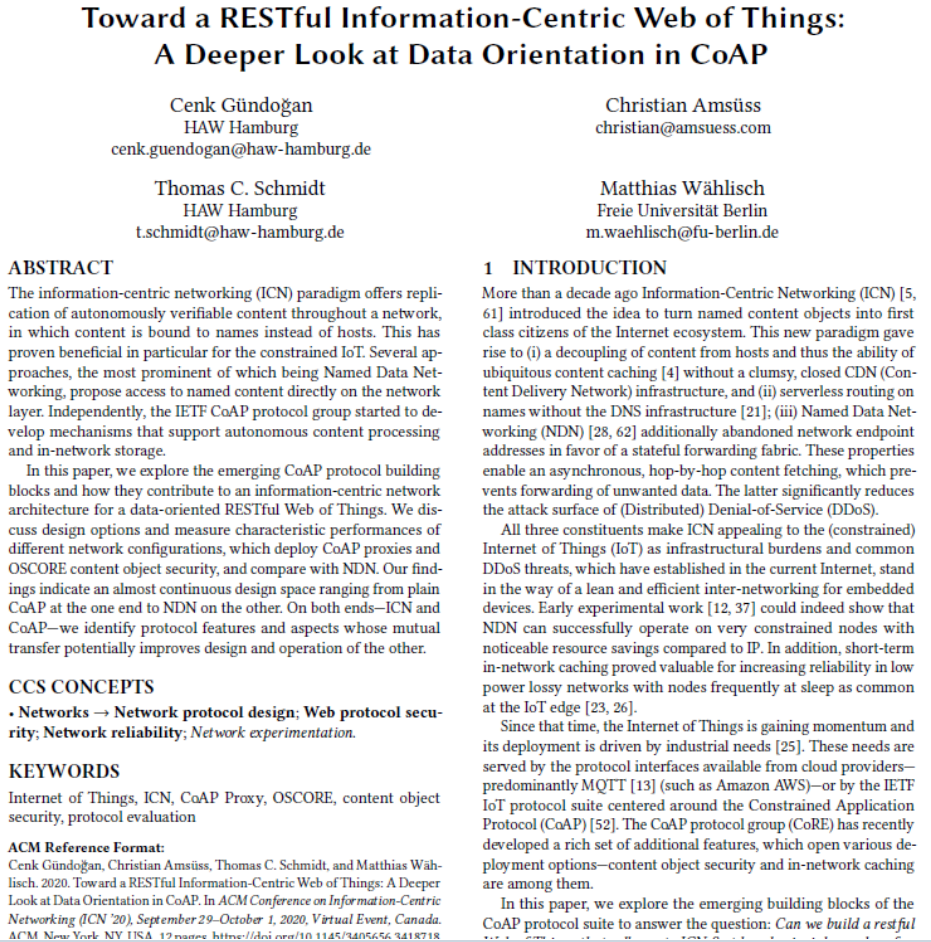


Time to Content Arrival



Literature

Cenk Gündogan, Christian Amsüss, Thomas C. Schmidt, Matthias Wählisch, **Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP**, **In:** *Proc. of 7th ACM Conference on Information-Centric Networking (ICN),* p. 77–88, ACM : New York, September 2020. https://doi.org/10.1145/3405656.3418718



Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP

Cenk Gündoğan
HAW Hamburg
cenk.guendogan@haw-hamburg.de

Christian Amsüss
christian@amsuess.com

Thomas C. Schmidt
HAW Hamburg
t.schmidt@haw-hamburg.de

Matthias Wählisch
Freie Universität Berlin
m.waehlisch@fu-berlin.de

ABSTRACT

The information-centric networking (ICN) paradigm offers replication of autonomously verifiable content throughout a network, in which content is bound to names instead of hosts. This has proven beneficial in particular for the constrained IoT. Several approaches, the most prominent of which being Named Data Networking, propose access to named content directly on the network layer. Independently, the IETF CoAP protocol group started to develop mechanisms that support autonomous content processing and in-network storage.

In this paper, we explore the emerging CoAP protocol building blocks and how they contribute to an information-centric network architecture for a data-oriented RESTful Web of Things. We discuss design options and measure characteristic performances of different network configurations, which deploy CoAP proxies and OSCORE content object security, and compare with NDN. Our findings indicate an almost continuous design space ranging from plain CoAP at the one end to NDN on the other. On both ends—ICN and CoAP—we identify protocol features and aspects whose mutual transfer potentially improves design and operation of the other.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Web protocol security**; **Network reliability**; *Network experimentation.*

KEYWORDS

Internet of Things, ICN, CoAP Proxy, OSCORE, content object security, protocol evaluation

ACM Reference Format:
Cenk Gündoğan, Christian Amsüss, Thomas C. Schmidt, and Matthias Wählisch. 2020. Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP. In *ACM Conference on Information-Centric Networking (ICN '20), September 29–October 1, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3405656.3418718

1 INTRODUCTION

More than a decade ago Information-Centric Networking (ICN) [5, 61] introduced the idea to turn named content objects into first class citizens of the Internet ecosystem. This new paradigm gave rise to (i) a decoupling of content from hosts and thus the ability of ubiquitous content caching [4] without a clumsy, closed CDN (Content Delivery Network) infrastructure, and (ii) serverless routing on names without the DNS infrastructure [21]; (iii) Named Data Networking (NDN) [28, 62] additionally abandoned network endpoint addresses in favor of a stateful forwarding fabric. These properties enable an asynchronous, hop-by-hop content fetching, which prevents forwarding of unwanted data. The latter significantly reduces the attack surface of (Distributed) Denial-of-Service (DDoS).

All three constituents make ICN appealing to the (constrained) Internet of Things (IoT) as infrastructural burdens and common DDoS threats, which have established in the current Internet, stand in the way of a lean and efficient inter-networking for embedded devices. Early experimental work [12, 37] could indeed show that NDN can successfully operate on very constrained nodes with noticeable resource savings compared to IP. In addition, short-term in-network caching proved valuable for increasing reliability in low power lossy networks with nodes frequently at sleep as common at the IoT edge [23, 26].

Since that time, the Internet of Things is gaining momentum and its deployment is driven by industrial needs [25]. These needs are served by the protocol interfaces available from cloud providers—predominantly MQTT [13] (such as Amazon AWS)—or by the IETF IoT protocol suite centered around the Constrained Application Protocol (CoAP) [52]. The CoAP protocol group (CoRE) has recently developed a rich set of additional features, which open various deployment options—content object security and in-network caching are among them.

In this paper, we explore the emerging building blocks of the CoAP protocol suite to answer the question: *Can we build a restful*

DEVICE MANAGEMENT

Management Problem

In the IoT, large numbers of devices require the following management tasks:

- Bootstrapping
- Device identification & registration
- Firmware updates
- Fault management
- Configuration & control
- Reporting

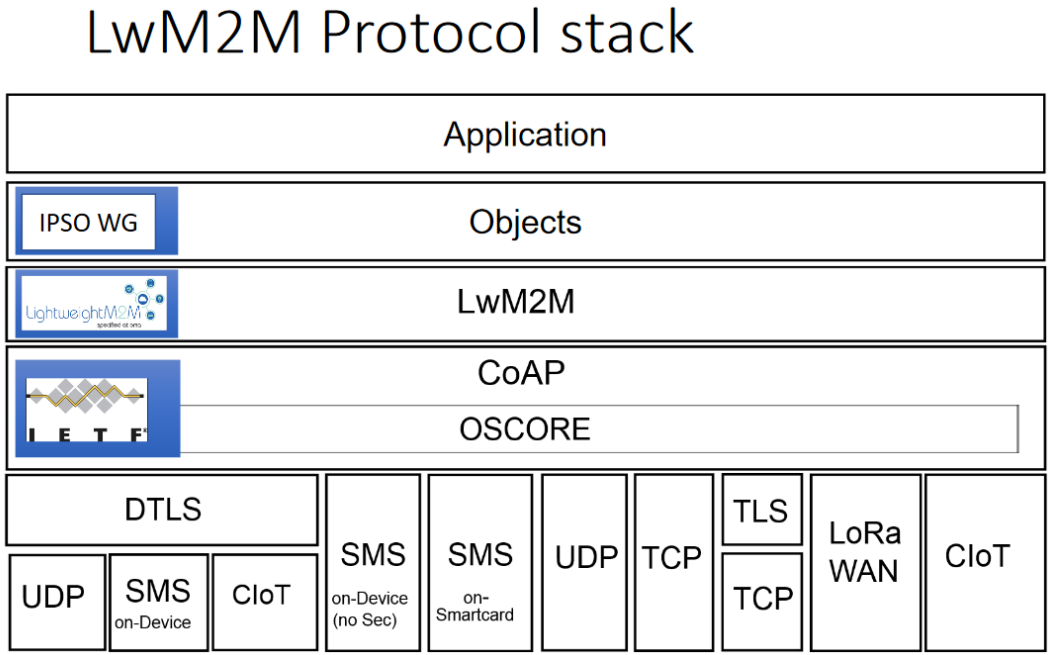
Management Integration Platforms

Several platforms for integrated management exist

- Watson IoT, Thingsware, Fiware, AWS IoT Mgmt, ..

LwM2M optimized for constrained IoT (< 20 kB M)

- OpenMobileAlliance
- Integrated with IETF stack



LwM2M Architecture

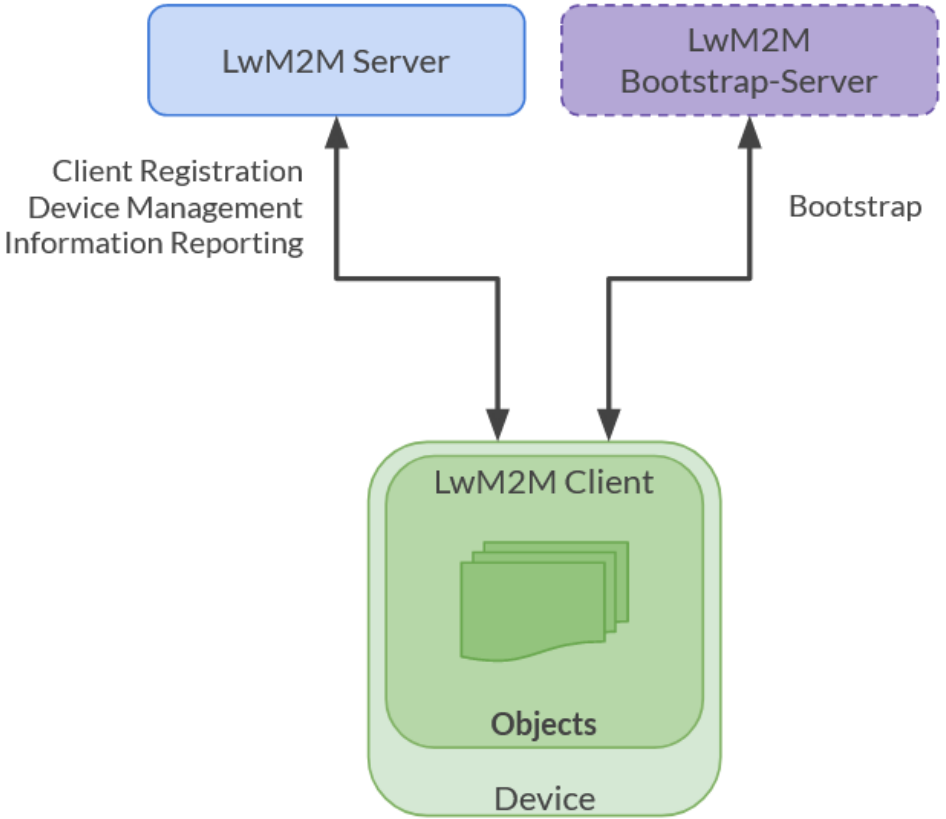
Bootstrap process

- Access Control Lists are installed
- Server credentials and information are installed

Client registers itself to the server.

The server may operate on the client's resources.

Access control is done **locally** by the client, upon installation of access rules.



Operations on LwM2M Objects

All operations are performed on resources of object instances

Objects' and resources' schemas placed in "OMA LwM2M Object and Resource Registry"

Access is controlled via Access Control Object instances associated to servers. Example: read the boolean input of a presence sensor:

```
GET
coaps://[fd00:c0de::1234]/3302/1/5
500
```

- ➔ Method for a Read operation in CoAP binding
- ➔ Authority
- ➔ Object ID
- ➔ Instance 1 (many sensors may be hosted)
- ➔ Digital Input State resource of the instance

Resumé

LwM2M is a lean, popular management approach to constrained IoT devices

Resources are easily accessible via simple CoAP requests from a **preconfigured server**

LwM2M does not define an interface to **request a server access** to a resource

LwM2M does not define an **interface between clients** to operate on resources