IRTF T2TRG, W3C WoT IG, OCF joint meeting

Thing-to-Thing RG (T2TRG) / OCF meeting

San Jose, CA, US, 2016-03-16

Chairs: Carsten Bormann, Ari Keränen

t2trg@irtf.org

Agenda

- IRTF T2TRG Intro
- OCF (OIC) Spec Overview
- OCF from OIC
- W3C WoT Overview
- W3C WoT Current Practices & Slugfests
- OCF/IETF Alignment
- **Discussion**

Disclaimer (IETF/IRTF)

- Nobody speaks for the IETF
  - The IETF is a collection of consensus processes
- Formal Liaisons are managed by the IAB
- This is a meeting of people interested in progressing the Internet of Things

Eleven years of standardizing the "Internet of Things"

Thing-to-Thing RG (T2TRG) / OCF meeting

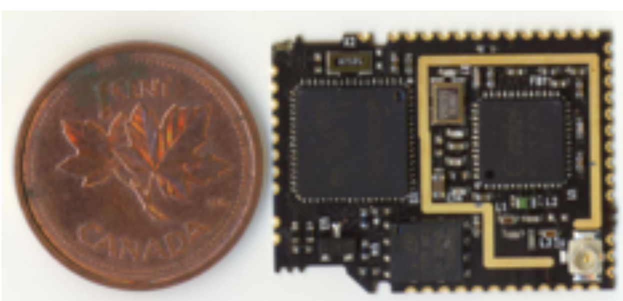
San Jose, CA, US, 2016-03-16

Chairs: Carsten Bormann, Ari Keränen
t2trg@irtf.org

Internet of Things?

- Passive Nodes
  ("RFID")
  Logistics/Supply Chains,
  Payment Cards

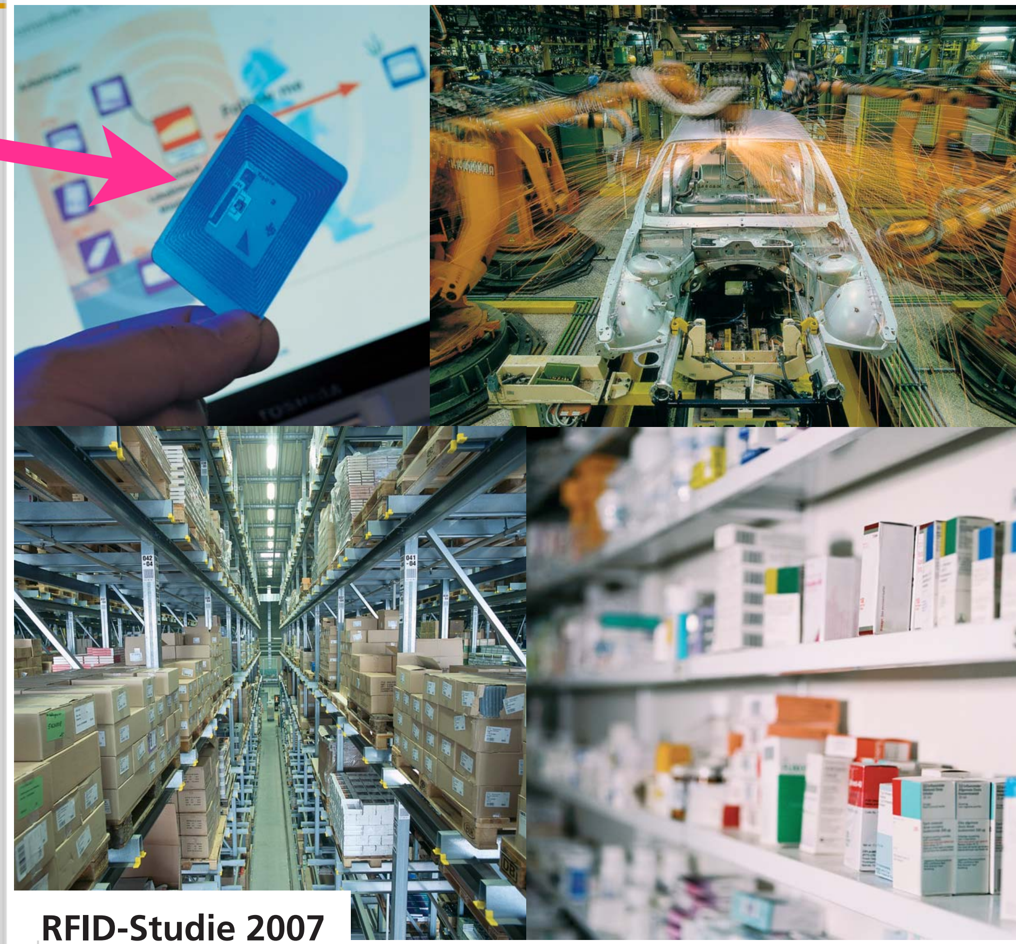
- Active Nodes
  ("Smart Objects")



Technologie-Zentrum Informatik

MES

Fraunhofer Institut Sichere Informations-Technologie



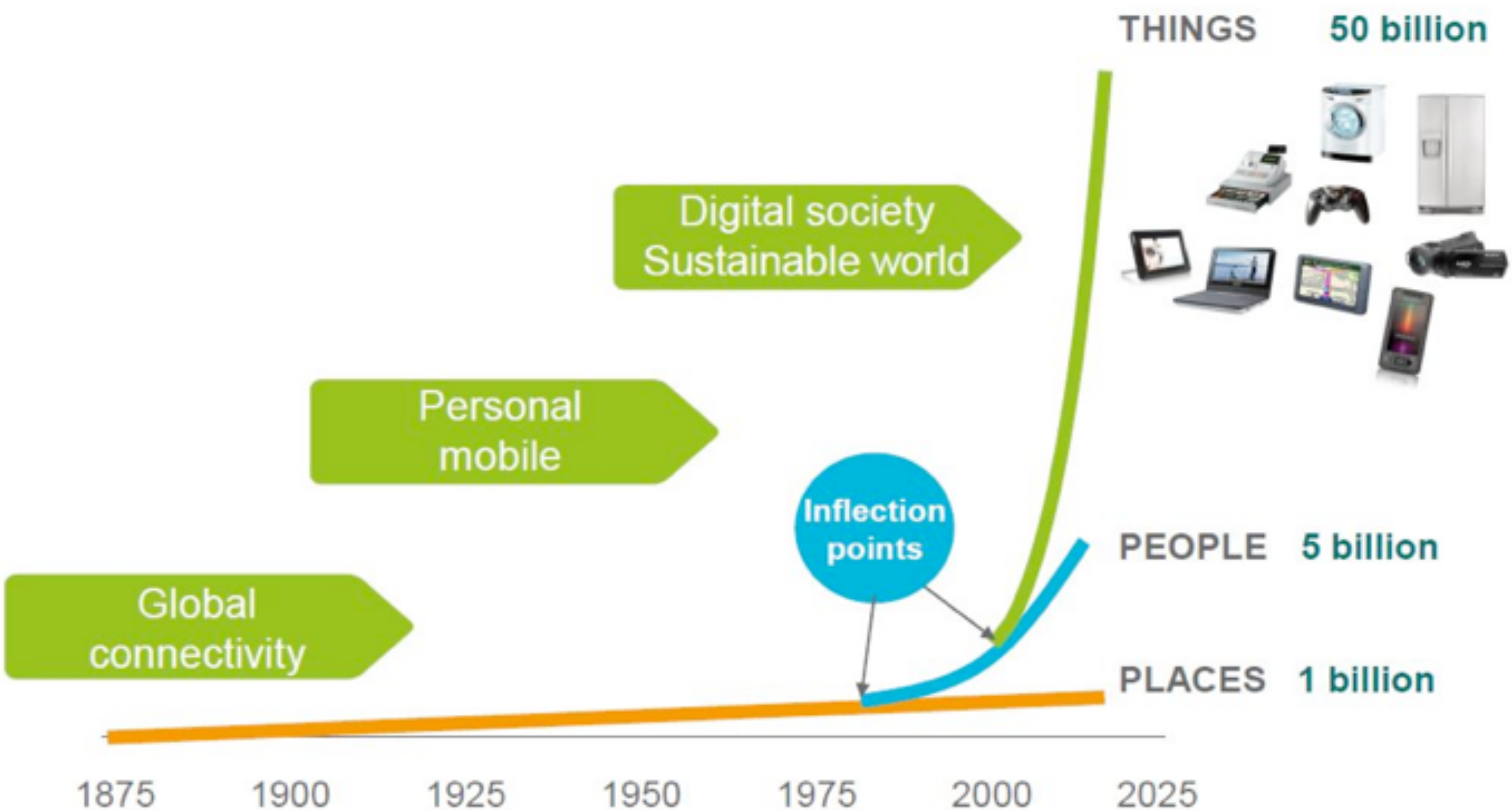
RFID-Studie 2007

Technologieintegrierte Datensicherheit bei RFID-Systemen

Bundesministerium für Bildung und Forschung

Diese Studie wurde im Rahmenprogramm Mikrosysteme 2004-2009 durch das Bundesministerium für Bildung und Forschung gefördert.

Connecting: Places → People → Things



Source: Ericsson

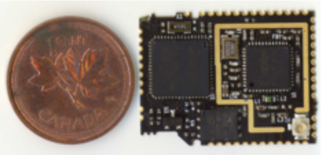
Scale up:

Number of nodes

(50 billion by 2020)

Scale down:

node



Scale down:

cost

complexity

cent

kilobyte

megahertz

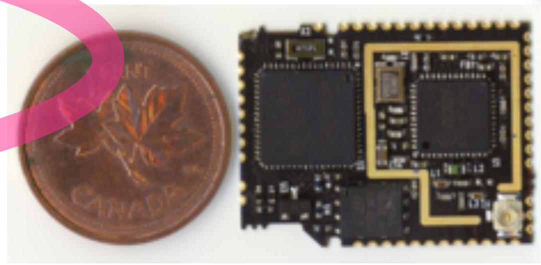
Constrained nodes: orders of magnitude

RFC 7228

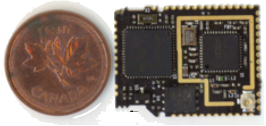
10/100 vs. 50/250

- There is not just a single class of “constrained node”

- Class 0: too small to securely run on the Internet
  - “too constrained”
- Class 1: ~10 KiB data, ~100 KiB code
  - “quite constrained”, “10/100”
- Class 2: ~50 KiB data, ~250 KiB code
  - “not so constrained”, “50/250”

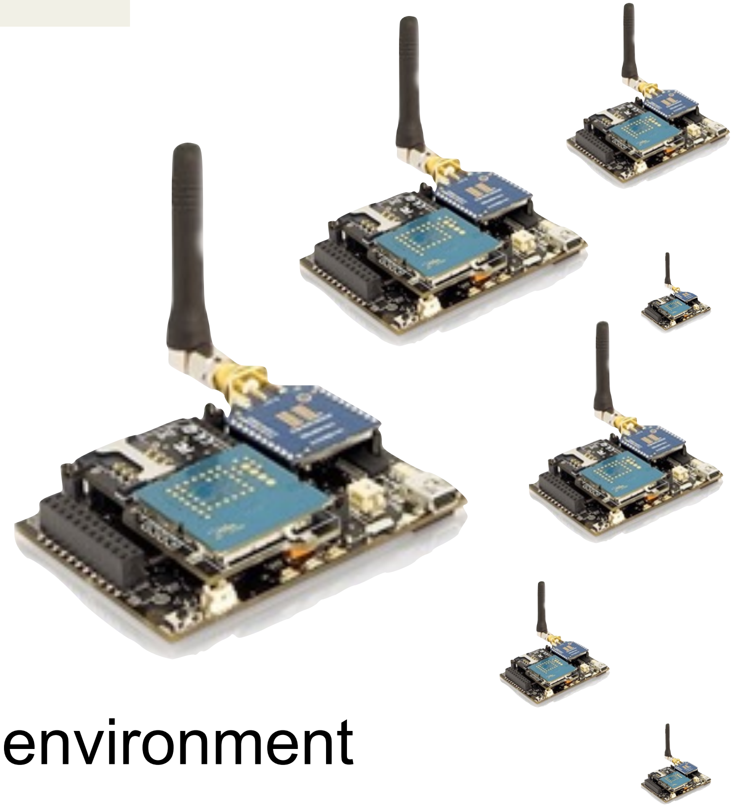


- These classes are not clear-cut, but may structure the discussion and help avoid talking at cross-purposes



http://www.flickr.com/photos/blahflowers/3878202215/sizes/l

Constrained **networks**



- **Node**: ... must sleep a lot (**µW!**)
  - vs. “always on”
- **Network**: **~100 kbit/s**, high loss, high link variability
- May be used in an unstable radio environment
- Physical layer packet size may be limited (**~100 bytes**)
- “LLN low power, lossy network”

802.15.4 „ZigBee“
Bluetooth Smart
Z-Wave
DECT ULE

Constrained Node Networks

Networks built from
Constrained Nodes,
where much of the
Network Constraints come from
the constrainedness of the Nodes

Constrained Node Networks

Internet of Things	IoT
Wireless Embedded Internet	WEI
Low-Power/Lossy Networks	LLN
IP Smart Objects	IPSO

Internet of Things?

IP = *Internet* Protocol

TZI

“IP is important”

IP = *Integration* Protocol

IP: drastically reducing barriers

- **IP telephony** (1990s to now): replace much of the special telephony hardware by routers and servers
  - several orders of magnitude in cost reduction
  - available programmer pool increases massively
  - → What started as convergence, turned into **conversion**
- Before: “**Btx** externer Rechner” vs. Web Server
- Now: **Internet of Things**

But do we **need** all of the baggage?

Or, just because we *can* move it, do we still **want** it?

Can you put a sofa on a motorcycle?

Yes, you can.

But do you want to?

Is sofa transport even a good criteria for vehicle selection?

http://www.bbc.com/news/world-africa-21769053

Two camps

- IP is too expensive for my microcontroller application (my hand-knitted protocol is better)

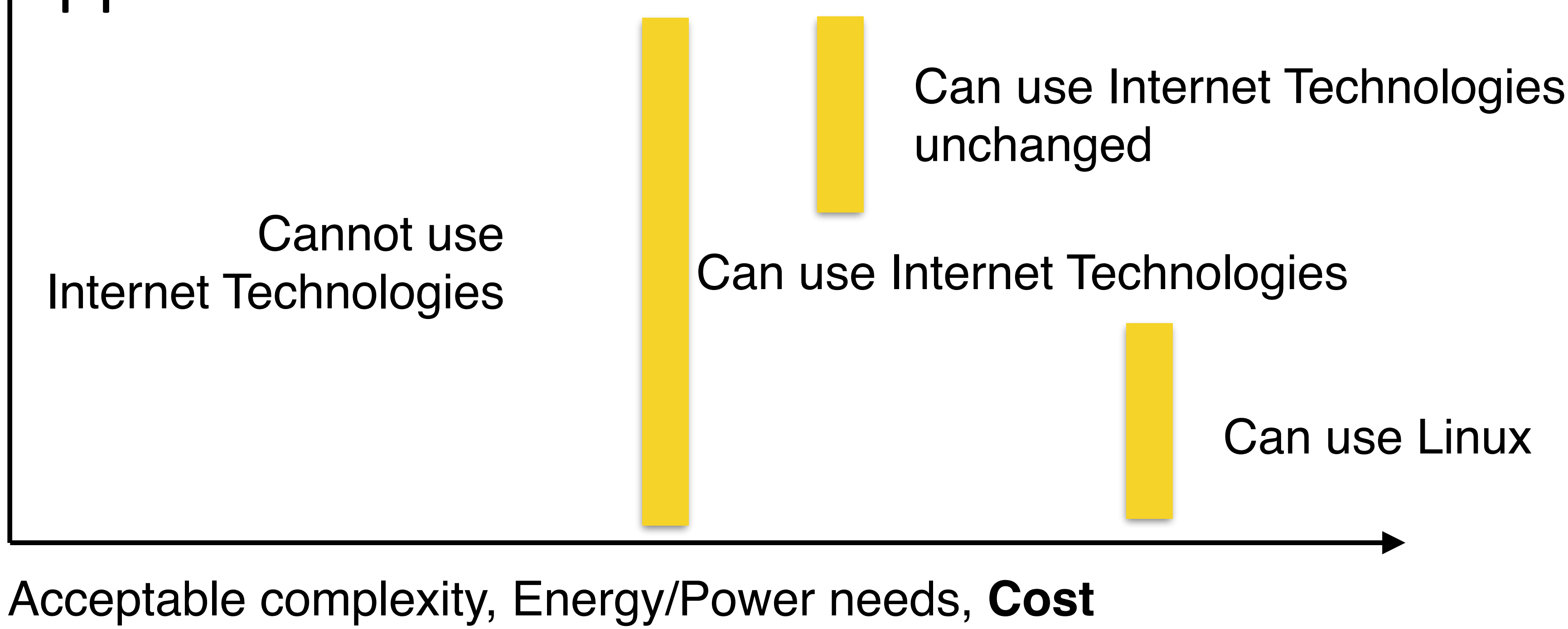
vs.

- IP already works well as it is, just go ahead and use it

- Both can be true!

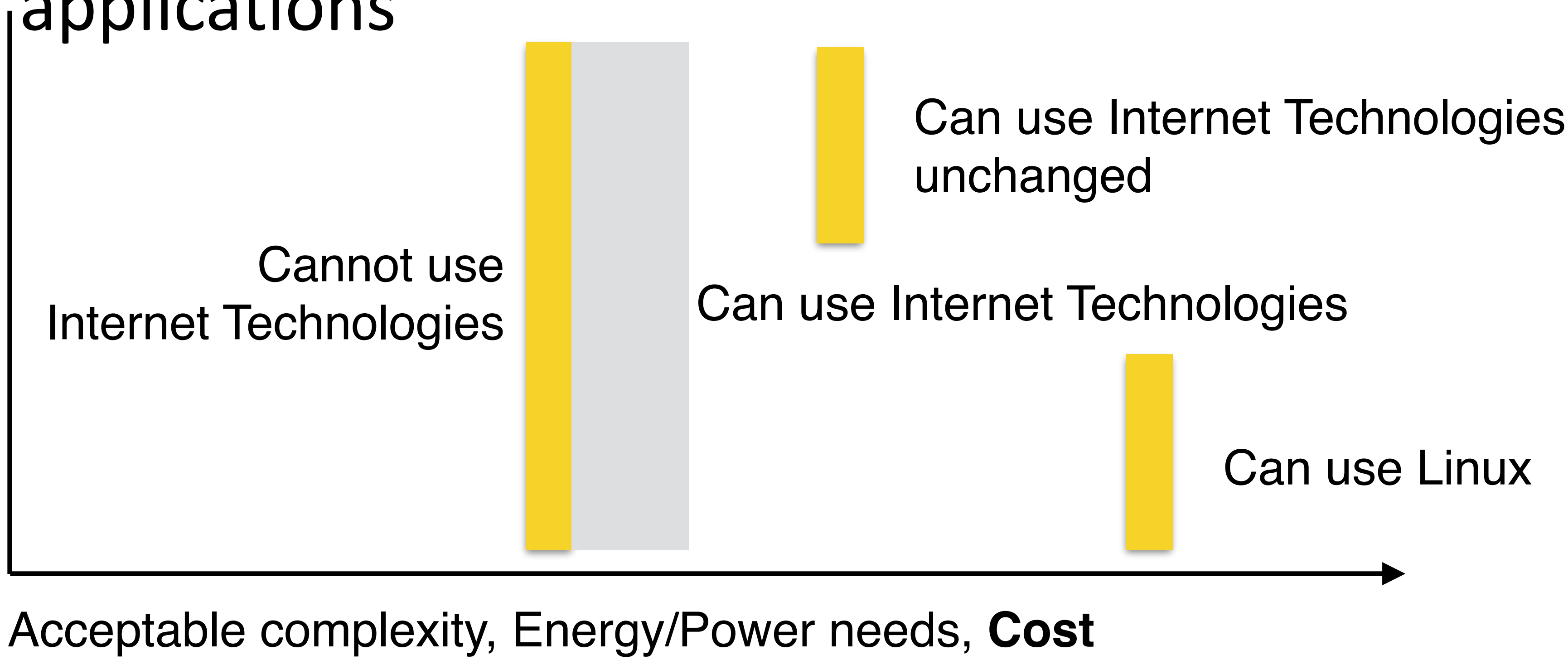
Moving the boundaries

- Enable Internet Technologies for mass-market applications



Moving the boundaries

- Enable Internet Technologies for mass-market applications



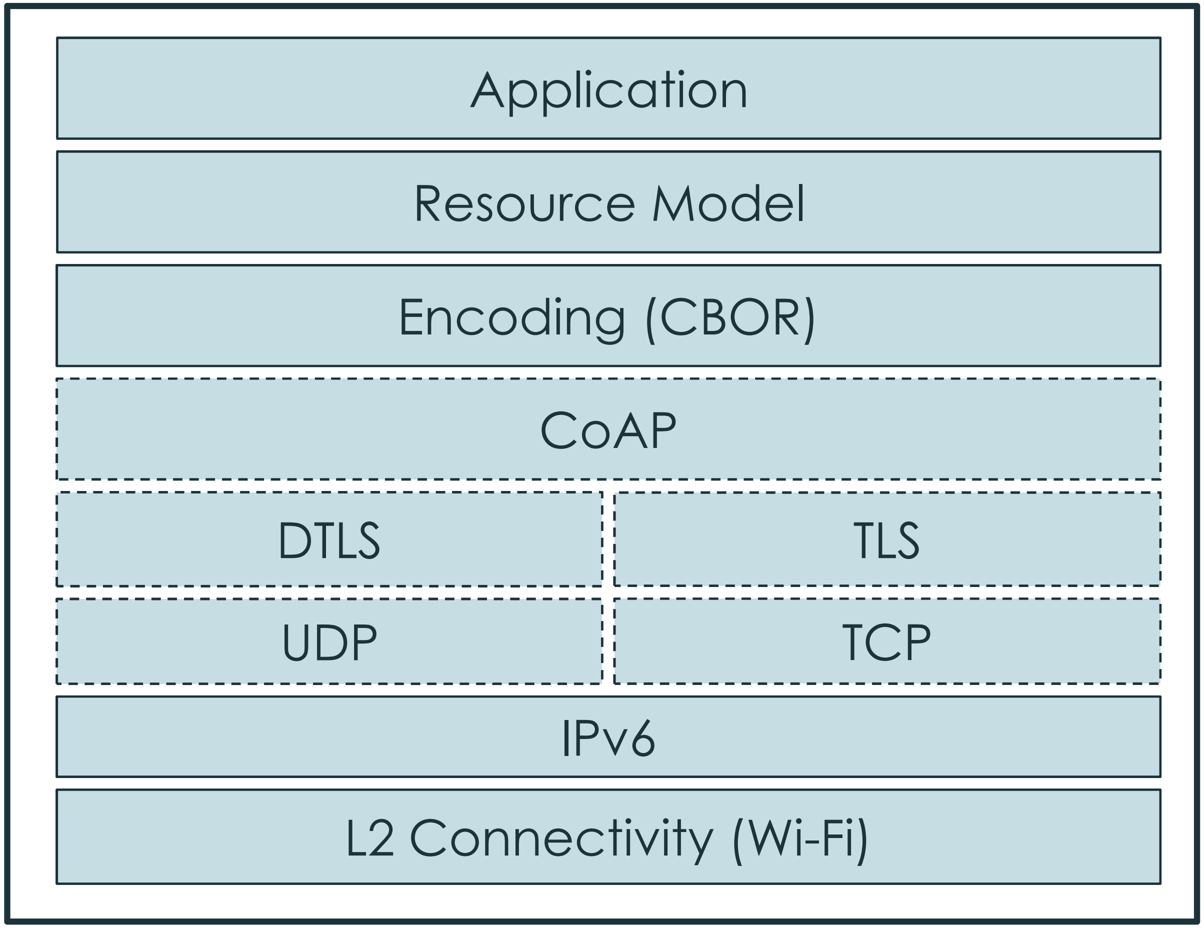
IETF®

We make the net work

IETF: Constrained Node Network WG Cluster

INT	LWIG	Guidance
INT	6LoWPAN ✔	IP over 802.15.4
INT	6Lo	IP-over-foo
INT	6TiSCH	IP over TSCH
RTG	ROLL	Routing (RPL)
APP	CoRE	REST (CoAP) + Ops
SEC	DICE ✔	Improving DTLS
SEC	ACE	Constrained AA
SEC	COSE	Object Security

Protocol Stack



Project B OIC Stack

[Source: OCF]

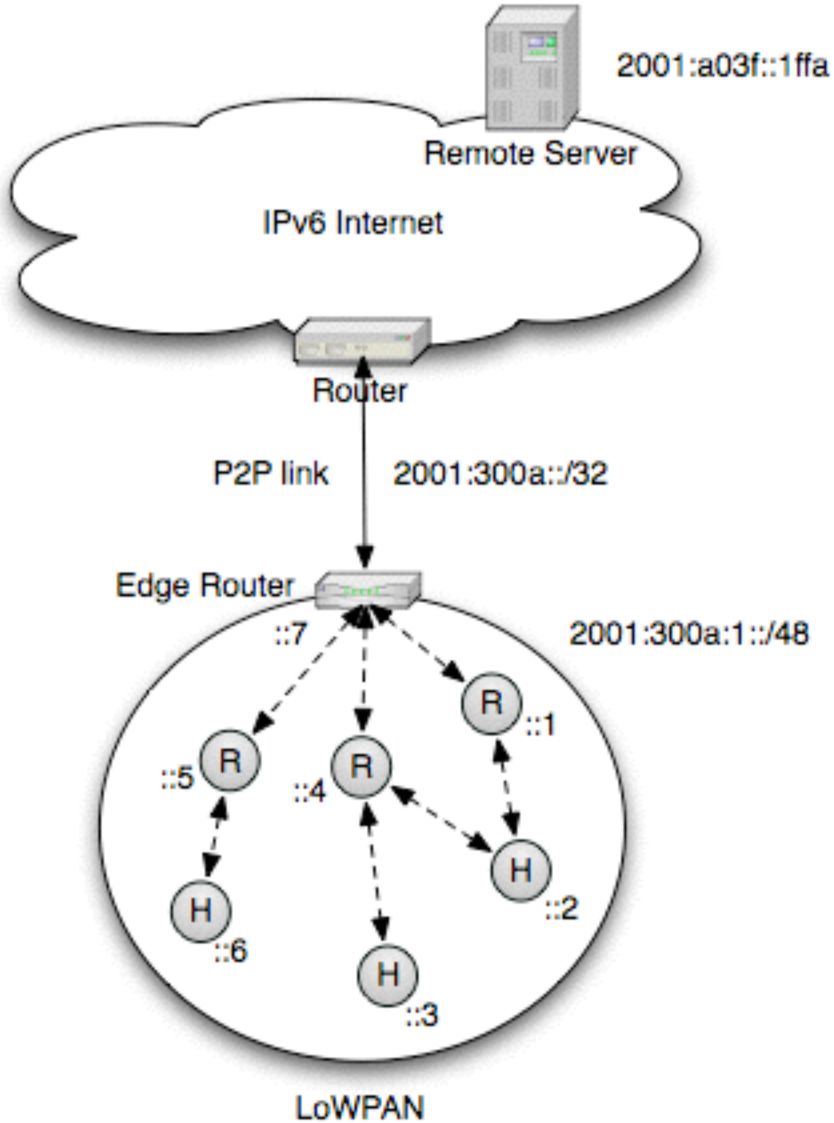
2005-03-03: 6LoWPAN

- “IPv6 over Low-Power WPANs”: IP over X for 802.15.4
  - Encapsulation ➔ RFC 4944 (2007)
  - Header Compression redone ➔ RFC 6282 (2011)
  - Network Architecture and ND ➔ RFC 6775 (2012)
  - (Informationals: RFC 4919, RFC 6568, RFC 6606)

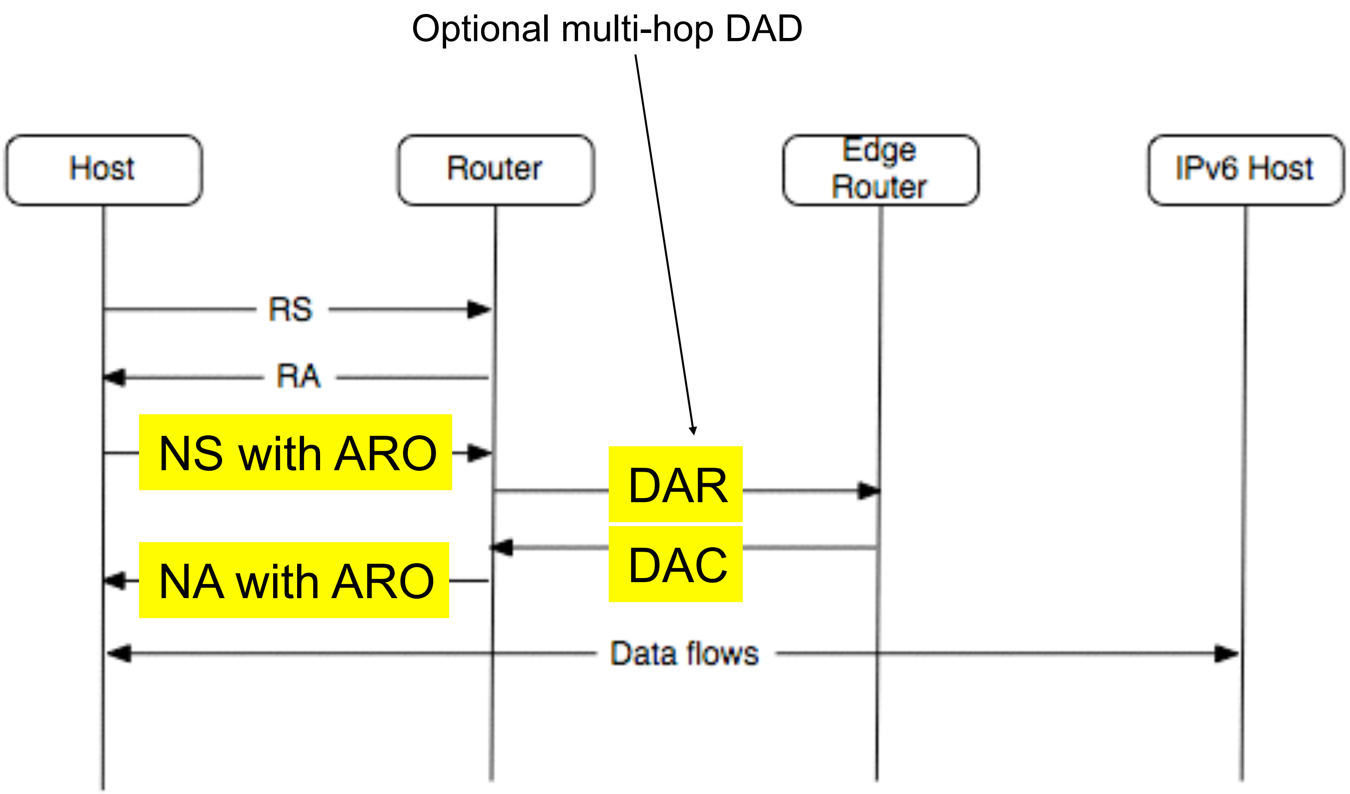
6LoWPAN breakthroughs

- RFC 4944: make IPv6 possible (fragmentation)
- RFC 6282: **area text state** for header compression
- RFC 6775: rethink IPv6
  - addressing: embrace **multi-link subnet** (RFC 5889)
  - get rid of subnet multicast (**link multicast only**)
  - adapt IPv6 ND to this (➔ "**efficient ND**")

Addressing Example



Typical 6LoWPAN-ND Exchange



Make good use of less-constrained nodes

- LBR/Edge Router: Runs DAD (and thus 16-bit address allocation)
- LBR keeps list of nodes ("whiteboard")
- LBR is **only** node with a need to **scale** with network
- (LBR already needs more power to talk to non-6LoWPAN side)

6LoWPAN part 2:

- Fix addressing model to be more realistic of a volatile (not really: mobile) wireless network
- Thoroughly get rid of some fluff (IP multicast):
  - Multicast use by ND-classic
  - The resulting need to do multicast forwarding at the subnet level
  - The resulting need to run MLD for solicited-node multicast addresses

TZi

6LoWPAN =

RFC4944
– HC1/HC2
+ **RFC6282** (6LoWPAN-HC)
+ **RFC6775** (6LoWPAN-ND)

6LoWPAN =

IPv6 over IEEE 802.15.4

6Lo =

6LoWPAN Technologies for other radios

Technology	Traits	
IEEE 802.15.4 ("ZigBee")	Many SoCs, 0.9 or 2.4 GHz, 6TiSCH upcoming	2.4 GHz
BlueTooth Smart	On **every Phone**	
DECT ULE	**Dedicated Spectrum**, In every home gateway	1.8 GHz
ITU-T G.9959 ("Z-Wave")	Popular @home	0.9 GHz
802.11ah ("HaLow")	Low power "WiFi"	
NFC	**Proximity**	13.56 MHz
6lobac	**Wired** (RS485)	
IEEE 1901.2 (LF PLC)	Reuses mains **power** lines	
Ethernet + PoE	**Wired**, supplies 12–60 W	
WiFi, LTE, …	**Power?**	

6Lo

2008-02-11: ROLL

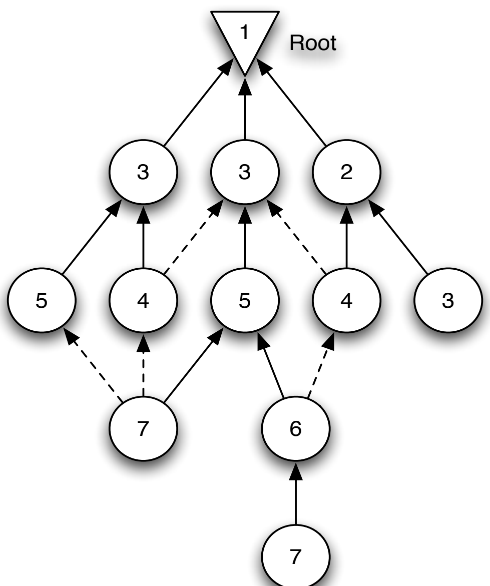
- “Routing Over Low power and Lossy networks”
  - Tree-based routing “RPL” ➔ RFC 6550–2 (2012)
    - with Trickle ➔ RFC 6206 (2011)
    - with MRHOF ➔ RFC 6719
  - Experimentals: P2P-RPL (RFC 6997), Meas. (RFC 6998)
  - In processing: MPL (Semi-Reliable Multicast Flooding)
  - (Lots of Informationals: RFC 5548 5673 5826 5867 7102 7416)

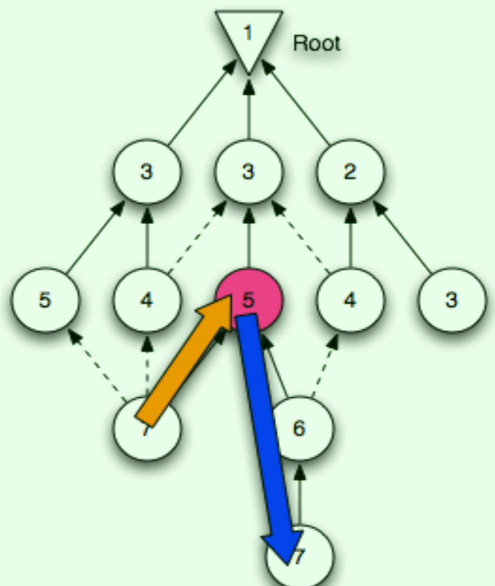
RPL: Routing for CN/N

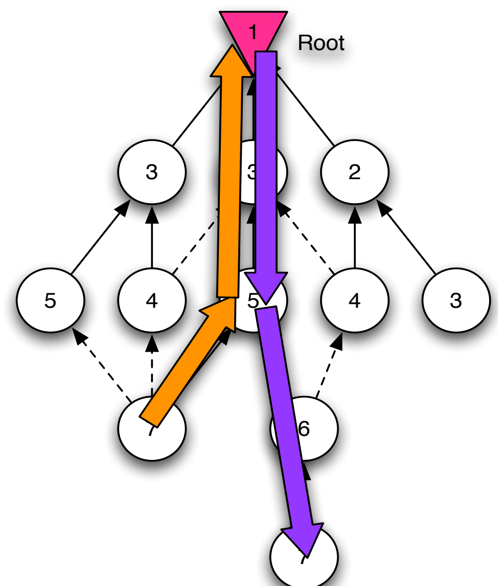
2012

Metrics: e.g., **ETX**

- **RFC 6550**: Specialized routing protocol **RPL** – Rooted DAGs (directed acyclic graphs)
  - **redundancies** in the tree help cope with churn
  - "**rank**": loop avoidance
  - **Storing** Mode: Every router has map of **subtree**
  - **Non-Storing** Mode: Only **root** has map of tree







ROLL breakthroughs

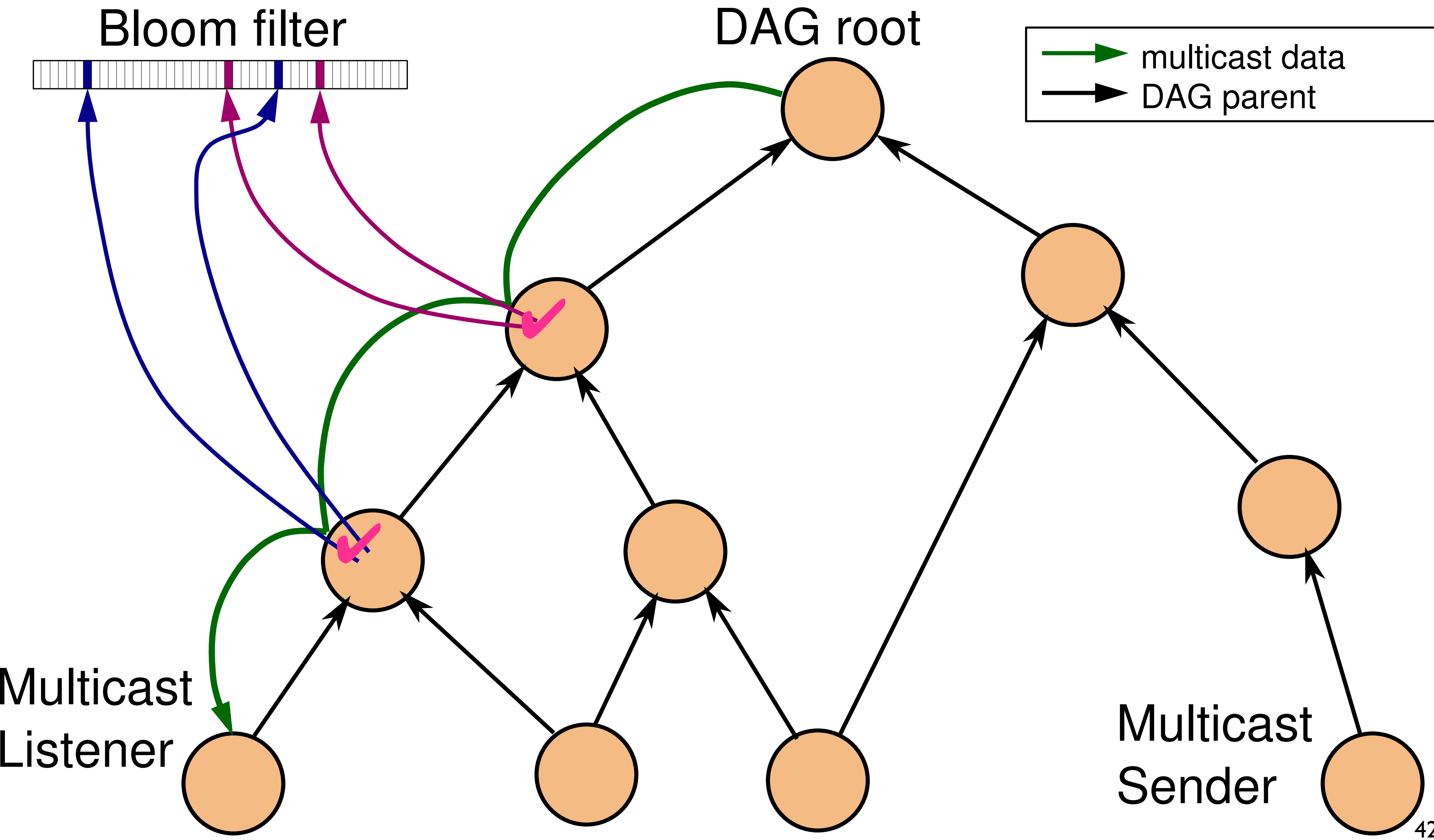
- RFC 6206: **trickle** (benefit from network stability)
- RFC 6550: **DODAG** (multi-parent tree)
    - separate local and global repairs
    - embrace the tree
    - non-storing mode: embrace the root

Make good use of less-constrained nodes

- LBR: "LLN Border Router" (root of DAG)
- Non-Storing mode: LBR keeps map of network
  - LBR is **only** node with a need to **scale** with network
  - (in storing mode, every router needs to scale with its subnetwork — the size of which cannot be controlled)

Multicast?

Constrained-Cast:
Send **Bloom Filter** with packet, match OIF



2010-03-09: CoRE

- “Constrained Restful Environments”
  - CoAP ➔ RFC 7252 (~~2013~~2014)
    - Observe: RFC 7641, Block
  - Experimentals: RFC 7390 group communications
  - Discovery (»Link-Format«) ➔ RFC 6690

The elements of success of the Web

- HTML
  - uniform **representation** of documents
  - (now moving forward to HTML5 with CSS, JavaScript)
- URIs
  - uniform **referents** to data and services on the Web
- HTTP
  - universal **transfer protocol**
  - enables a distribution system of proxies and reverse proxies

Translating this to M2M

- HTML
  - uniform **representation** of documents
  - (now moving forward to HTML5 with CSS, JavaScript)
- URIs
  - uniform **referents** to data and services on the Web
- HTTP
  - universal **transfer protocol**
  - enables a distribution system of proxies and reverse proxies

New data formats:
M2M semantics instead of
presentation semantics

"Make things
as simple as possible,
but not simpler.

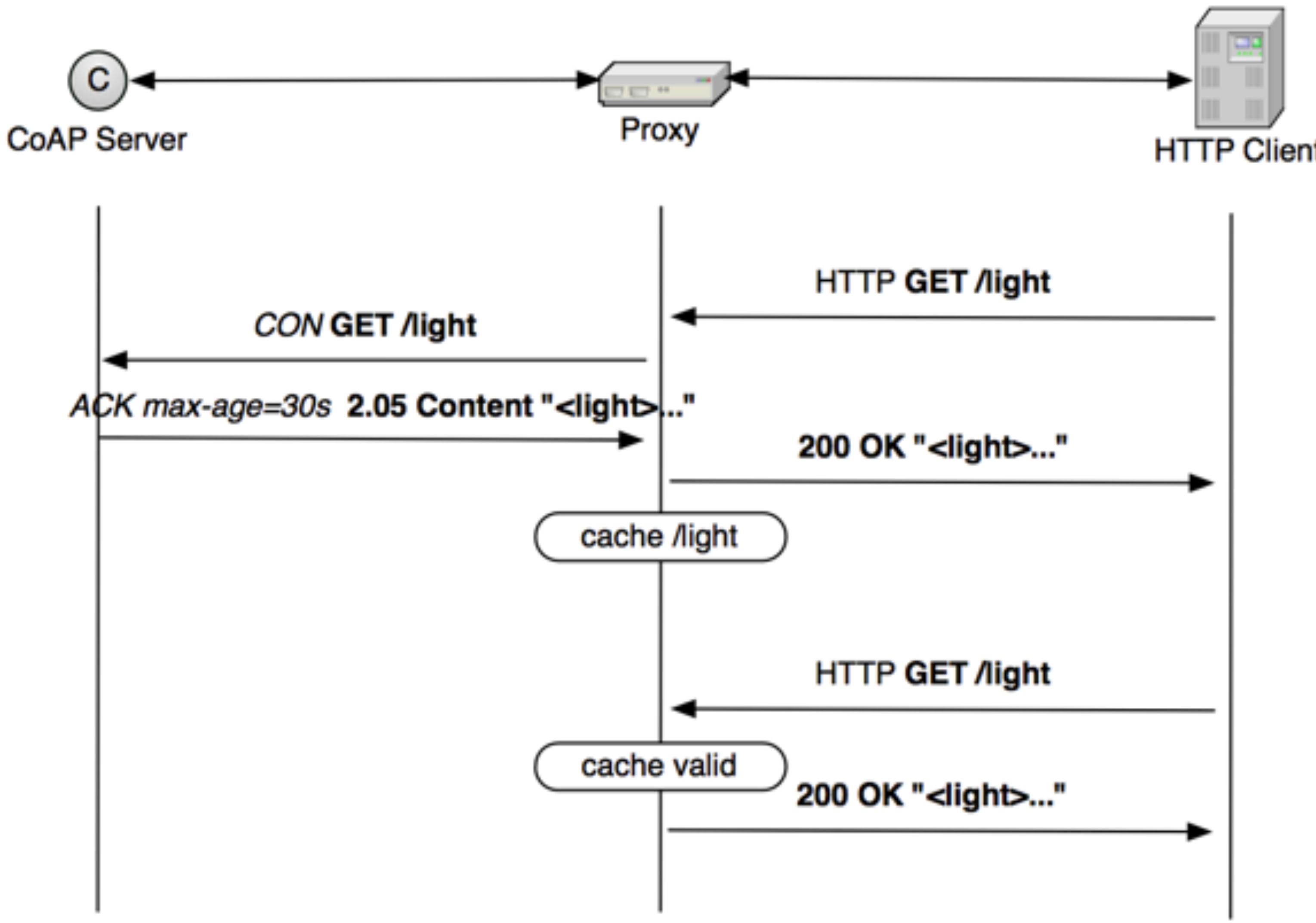
Attributed to Albert Einstein

The **Co**nstrained **A**pplication **P**rotocol

CoAP

- implements HTTP's **REST** model
  - GET, PUT, DELETE, POST; media type model
- while avoiding most of the complexities of HTTP
- **Simple** protocol, datagram only (UDP, DTLS)
- 4-byte header, compact yet simple options encoding
- adds "observe", a lean notification architecture

Proxying and caching



Source: 6lowpan.net

CoRE breakthroughs

- RFC 7252: embrace **REST**
  - but get rid of HTTP **baggage**
  - and extend REST with **Observe**
- RFC 6690: **Web Linking** for discovery: `/.well-known/core`
  - building **resource-directory** on top of that

Overview | Specification | Implementations | Tools

http://coap.technology

CoAP

RFC 7252 Constrained Application Protocol

"The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the **Internet of Things.**
The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation."

REST model for small devices

Like HTTP, CoAP is based on the wildly successful REST model: Servers make resources available under a URL, and clients access these resources using methods such as GET, PUT, POST, and DELETE

Made for billions of nodes

The Internet of Things will need billions of nodes, many of which will need to be inexpensive. CoAP has been designed to work on microcontrollers with as low as 10 KiB of RAM and 100 KiB of code space (RFC 7228)

Well-designed protocol

CoAP was developed as an Internet Standards Document, RFC 7252. The protocol has been designed to last for decades. Difficult issues such as congestion control have not been swept under the rug, but have been addressed using the state

Security is not optional!

- HTTP can use TLS (“SSL”)
- CoAP: Use **DTLS** 1.2
  - Add 6LoWPAN-**GHC** for efficiency
- Crypto: Move to **ECC**
  - **P-256** curve
  - **SHA-256**
  - **AES-128**
- To do:
  - Commissioning models (Mother/Duckling, Mothership, …)
  - Authorization format and workflow
  - Performance fixes (DICE)

128-bit security (~ RSA 3072-bit)

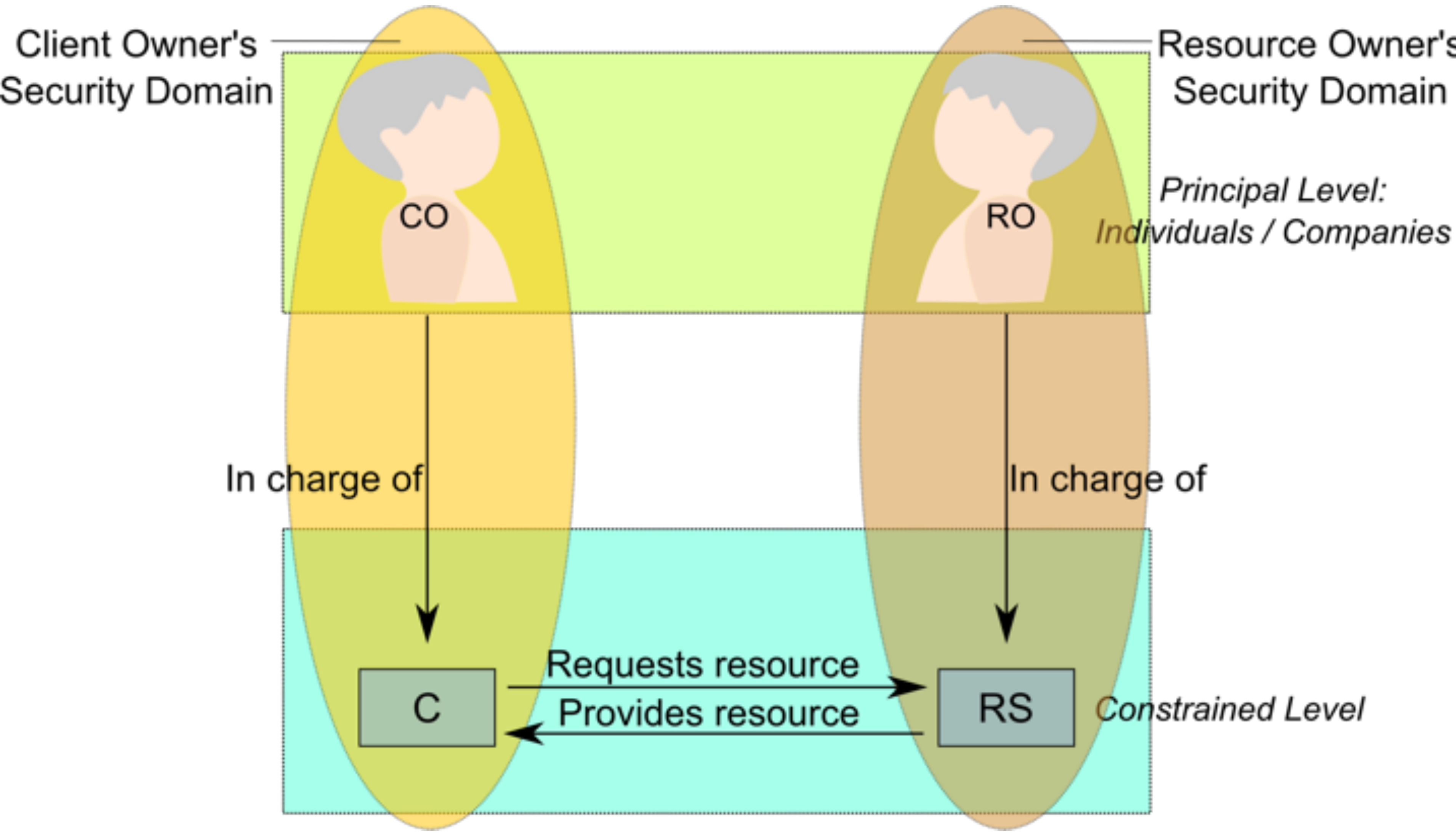
IoT “Security” today

- Thin perimeter protection
- WiFi password = keys to the kingdom
  - Once you are “in”, you can do everything
  - No **authorization**
- Doesn’t even work for a three-member family…

If it is not **usably secure**,
it's not
the **Internet of Things**

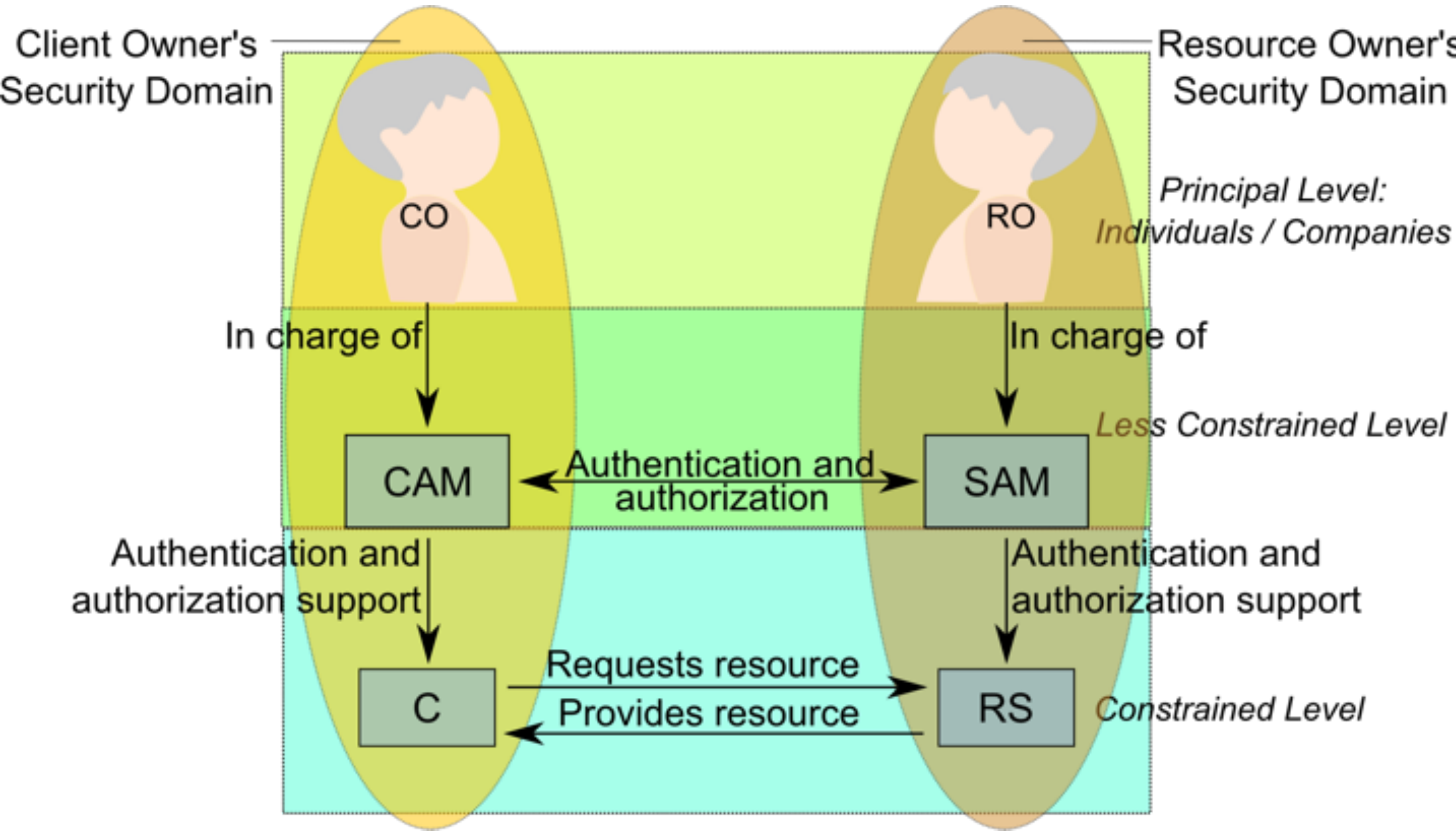
2014-05-05: ACE

- “Authentication and Authorization for Constrained Environments”
  - currently applying OAuth framework to IoT


Client Owner's
Security Domain
Resource Owner's
Security Domain
Principal Level:
Individuals / Companies
CO
RO
In charge of
In charge of
C
Requests resource
Provides resource
RS
Constrained Level

Make good use of less-constrained nodes

- C and RS may be too simple to run detailed business logic
  - Much more straight-forward to employ existing web-based systems for that
- Pair C and RS with a less-constrained node for running the business logic: C ➔ CAM, RS ➔ SAM


Client Owner's Security Domain
Resource Owner's Security Domain
Principal Level: Individuals / Companies
CO
RO
In charge of
In charge of
Less Constrained Level
CAM
Authentication and authorization
SAM
Authentication and authorization support
Authentication and authorization support
C
Requests resource
Provides resource
RS
Constrained Level

Make good use of less-constrained nodes

- C and RS then only need to run a simple, business-logic independent authentication and authorization protocol
- Security of C and RS can be based on inexpensive symmetric encryption

2013-09-13: CBOR

- “Concise Binary Object Representation”:
  JSON equivalent for constrained nodes
  - start from JSON data model (no schema needed)
  - add binary data, extensibility (“tags”)
  - concise binary encoding (byte-oriented, counting objects)
  - add diagnostic notation
- Done without a WG (with APPSAWG support)

Data Formats

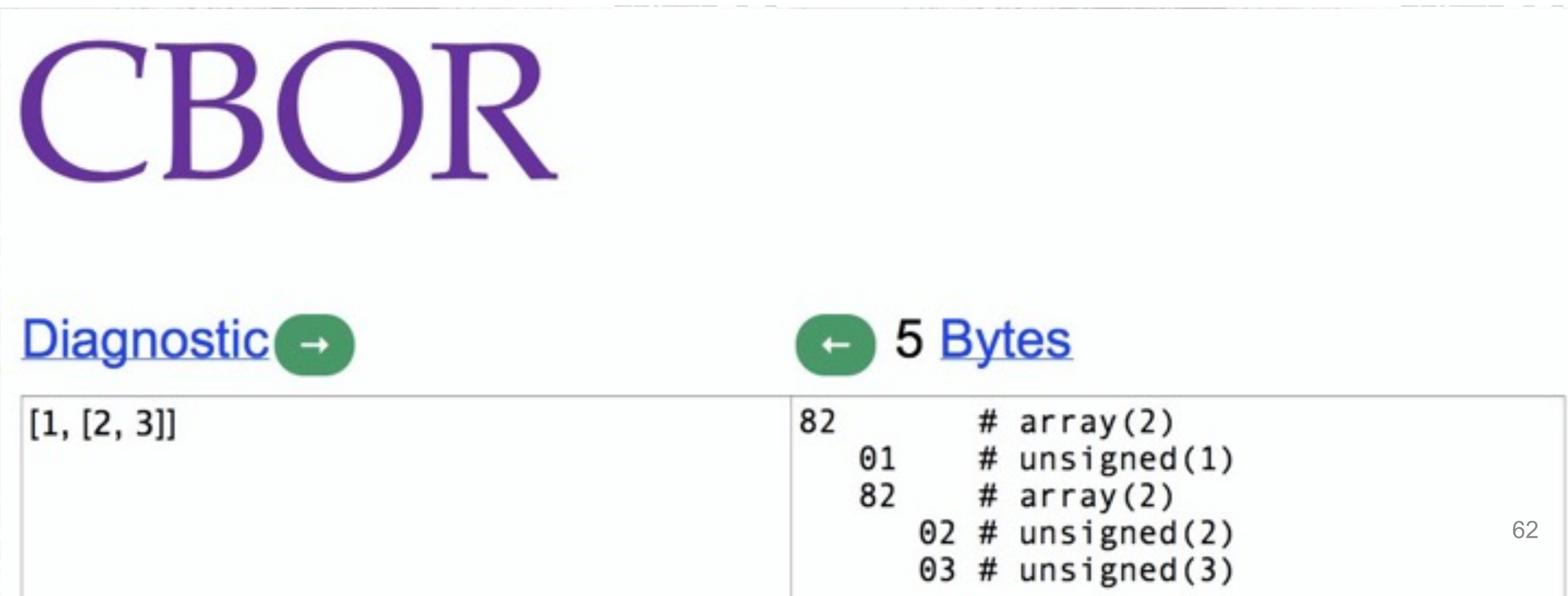
	Character-based	Concise Binary
Document-Oriented	XML	EXI
Data-Oriented	JSON	???

	Concise (Counted)	Streaming (Indefinite)
Format	[1, [2, 3]]	[_ 1, [2, 3]]
RFC 713*	c2 05 81 c2 02 82 83	
ASN.1 BER*	30 0b 02 01 01 30 06 02 01 02 02 01 03	30 80 02 01 01 30 06 02 01 02 02 01 03 00 00
MessagePack	92 01 92 02 03	
BSON	22 00 00 00 10 30 00 01 00 00 00 04 31 00 13 00 00 00 10 30 00 02 00 00 00 10 31 00 03 00 00 00 00 00	
UBJSON	61 02 42 01 61 02 42 02 42 03	61 ff 42 01 61 02 42 02 42 03 45*
CBOR	82 01 82 02 03	9f 01 82 02 03 ff

Table 5: Examples for different levels of conciseness

http://cbor.me: CBOR playground

- Convert back and forth between **diagnostic notation** (~JSON) and binary encoding



Overview | Specification | Implementations | Tools

http://cbor.io

CBOR

RFC 7049 Concise Binary Object Representation

"The Concise Binary Object Representation (CBOR) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation."

JSON data model

CBOR is based on the wildly successful JSON data model: numbers, strings, arrays, maps (called objects in JSON), and a few values such as false, true, and null.

No Schema needed

Embracing binary

Some applications that would like to use JSON need to transport binary data, such as encryption keys, graphic data, or sensor values. In JSON, these data need to be encoded (usually in base64 format), adding complexity and bulk.

Concise encoding

Stable format

CBOR is defined in an Internet Standards Document, RFC 7049. The format has been designed to be stable for decades.

Extensible

To be able grow with its applications and to

Data Formats

	Character-based	Concise Binary
Document-Oriented	XML	EXI
Data-Oriented	JSON	CBOR

Data Definition Language?

- Various “JSON Schema” proposals
  - e.g., “JSON Content Rules” (JCR)
  - geared to specific specification styles
- CBOR Data Definition Language: **CDDL**
  - simple, production-based language (similar to ABNF)

2015-06-03: COSE

- CBOR Object Signing and Encryption:
  Object Security for the IoT
- Based on **JOSE**: JSON Web Token, JWS, JWE, …
  - Data structures for signatures, integrity, encryption…
  - Derived from on OAuth JWT
  - Encoded in JSON, can encrypt/sign other data
- **COSE: use CBOR instead of JSON**
  - Can directly use binary encoding (no base64)
  - Optimized for constrained devices

Constrained Environment Requirements

- Message payloads are often **small** (nature of data)
  - transmission system optimized for that
  - fixed-size overheads hurt much more!
- Transmission/reception requires **power** (~100 µW → 50 mW)
  - keep message sizes reasonably small
  - don't rely on compression for that
    - compression requires CPU power, RAM, code space
- Handling messages requires **RAM** (~10 KiB)
  - minimize copying around things
    - or, worse, re-encoding, escape processing, ...
- all this requires code space in **Flash** (~100 KiB)
  - minimize code complexity
  - avoid multiple different ways to do the same thing

What to avoid

- **avoid**: base64 coding of binary
  - (message expansion, requirement for creating copies)
  - Easy to avoid for outer shell (cf. Richard Barnes' msgpack experiment)
  - Incompatible change: signing input
- **avoid**: JSON-encoding of data
  - (message expansion, creating copies for escape processing, code size)
  - ➔ Incompatible change: signing input
- secondary, but useful: minimize strings by enumerating frequent member names
  - (reduces message size, code space)

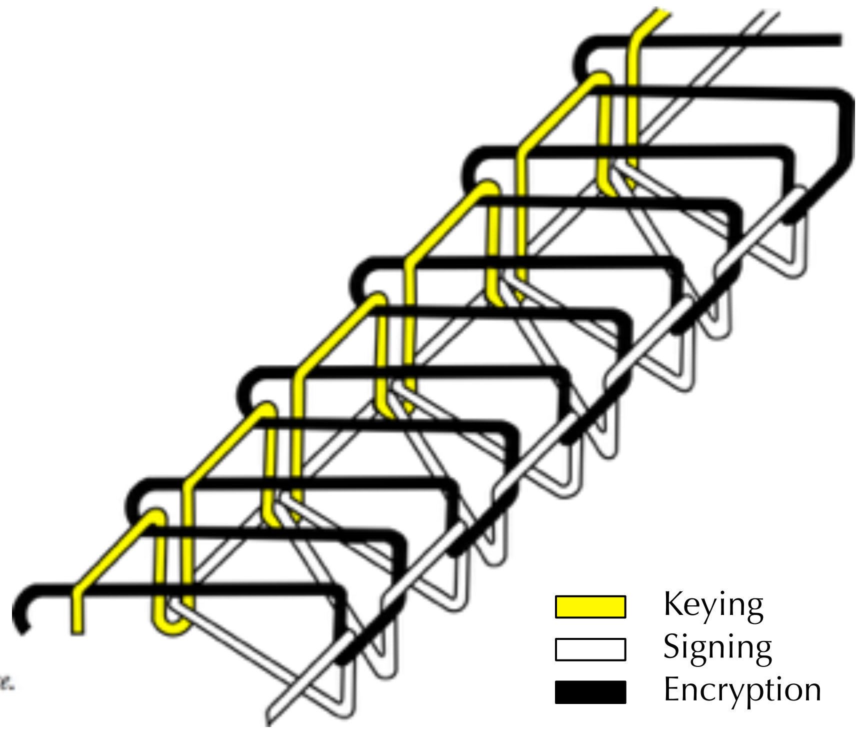
COSE?

- COSE is like JOSE, except
  - each use of JSON is replaced by an equivalent use of CBOR
  - base64-encoding is never done
  - frequent member names (“alg”…) are enumerated

coser

verbo transitivo/verbo intransitivo

1 Unir con hilo enhebrado en una aguja pedazos o partes de una tela, de cuero o de otro material semejante: *máquina de coser; coser el dobladillo de unos pantalones; coser una camisa; escucha la radio mientras cose.*



Application Layer Technologies

- The Web of Things: **CoAP** and HTTP
  - Using CoAP for management: OMA LWM2M, COMI
  - Time Series Data: CoAP-Pubsub (and XMPP, MQTT)
- Data Formats: **CBOR** and JSON
  - Data objects: OMA LWM2M, IPSO Smart Objects
  - Sensor data: SenML (in use in OMA LWM2M)
- Real Security
  - Communications: **DTLS** and TLS
  - Object Security: **COSE** and JOSE
  - Authenticated Authorization: ACE

IETF: Constrained Node Network WG Cluster

INT	LWIG	Guidance
INT	6LoWPAN ✔	IP over 802.15.4
INT	6Lo	IP-over-foo
INT	6TiSCH	IP over TSCH
RTG	ROLL	Routing (RPL)
APP	CoRE	REST (CoAP) + Ops
SEC	DICE ✔	Improving DTLS
SEC	ACE	Constrained AA
SEC	COSE	Object Security

Machine to Machine Application Protocols

- CoAP and Related IETF Standards
  - Machine to Machine (M2M) protocol modeled after HTTP
  - Compressed Binary mapping of REST API protocol
  - Asynchronous Notifications to support M2M use cases
  - Format for Machine Hyperlinks, CoRE Link-Format
- HTTP
  - Useful for less resource constrained environments
  - Works with existing libraries and servers
  - Well known extensions for asynchronous notification

Object Models and Data Models

- IPSO Smart Objects
  - Object/Resource URI template for M2M REST API
  - Defines Structure and Data Types for functionally specialized objects
  - E.g. Temperature Sensor, Light Controller, Load Controller
  - Compatible with CoAP, HTTP, and other underlying protocols
- Others being considered by various IoT Interest Groups (IOTWF, IIC, OIC)
- W3C Community group on Web of Things considering work on data models

ARM

IRTF: Internet Research Task Force (sister of IETF)

- IRTF complements IETF with longer-term **Research Groups**
- New: Thing-to-Thing Research Group (T2TRG)
- Investigate open research issues in:
  - turning a true "Internet of Things" into reality,
  - an Internet where low-resource nodes ("Things", "Constrained Nodes") can communicate among themselves and with the wider Internet, in order to partake in permissionless innovation.

How to use REST in IoT?

- Ignore it, build a SOAP on top
- Use it half-heartedly and reap some of the benefits
- Use it right
    - But what are the **best practices** that work well in the IoT?

RESEARCH

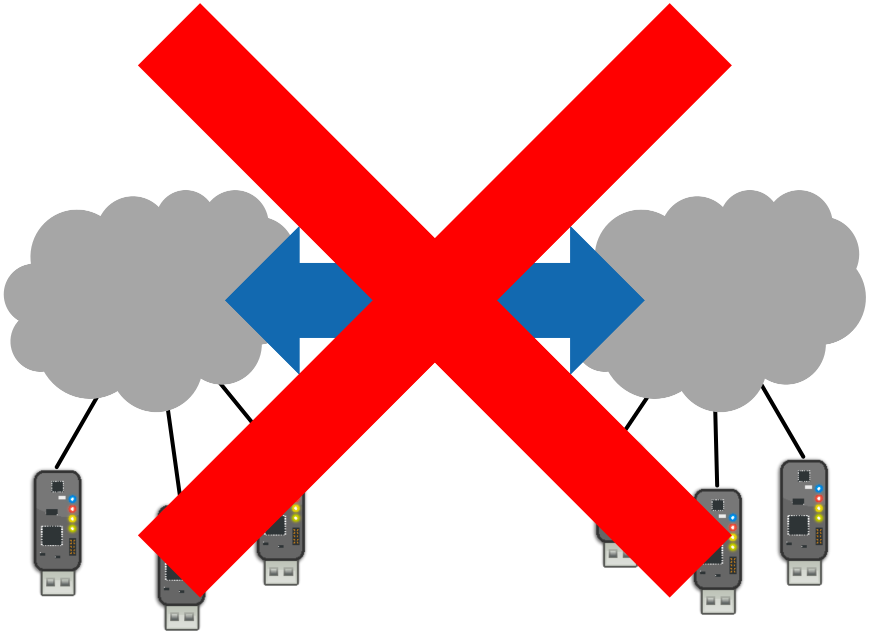
Near-term milestones

- Collect a small number of non-trivial, realistic **scenarios**
- Map technology to these scenarios;
  evaluate, benchmark, find gaps
- Document findings, best practices in **cookbooks**
- Run **plugReSTs** so researchers can test their approaches in the context of the scenarios

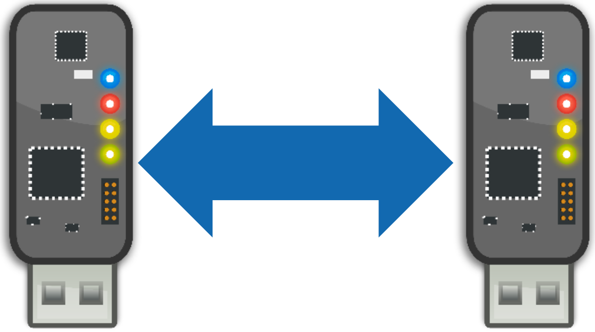
Evaluation
Framework

REST for Thing-to-Thing Communication

Cloud-to-Cloud (with Things)

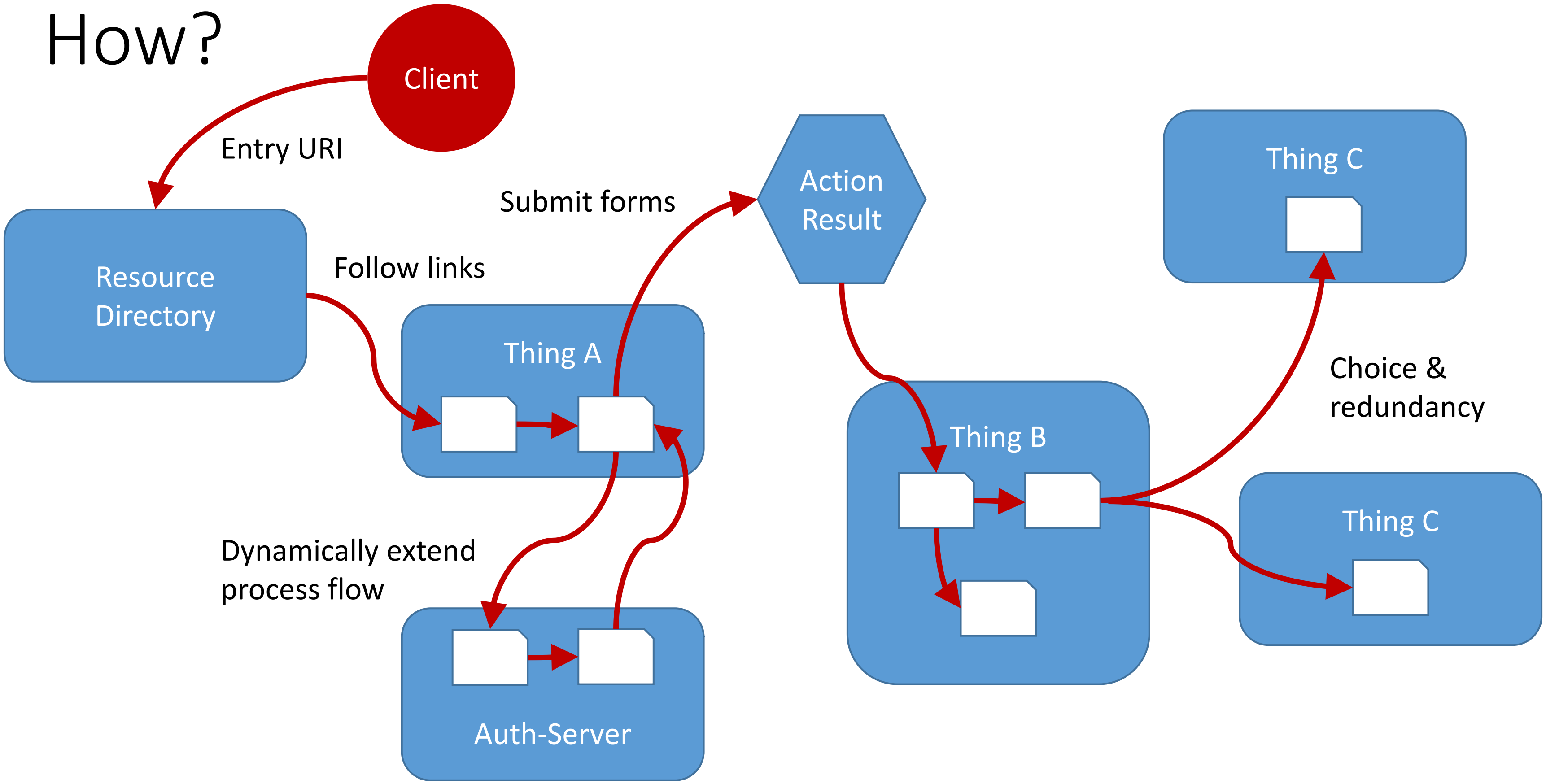


Thing-to-Thing
(may include cloud services)



Matthias Kovatsch
http://people.inf.ethz.ch/mkovatsc

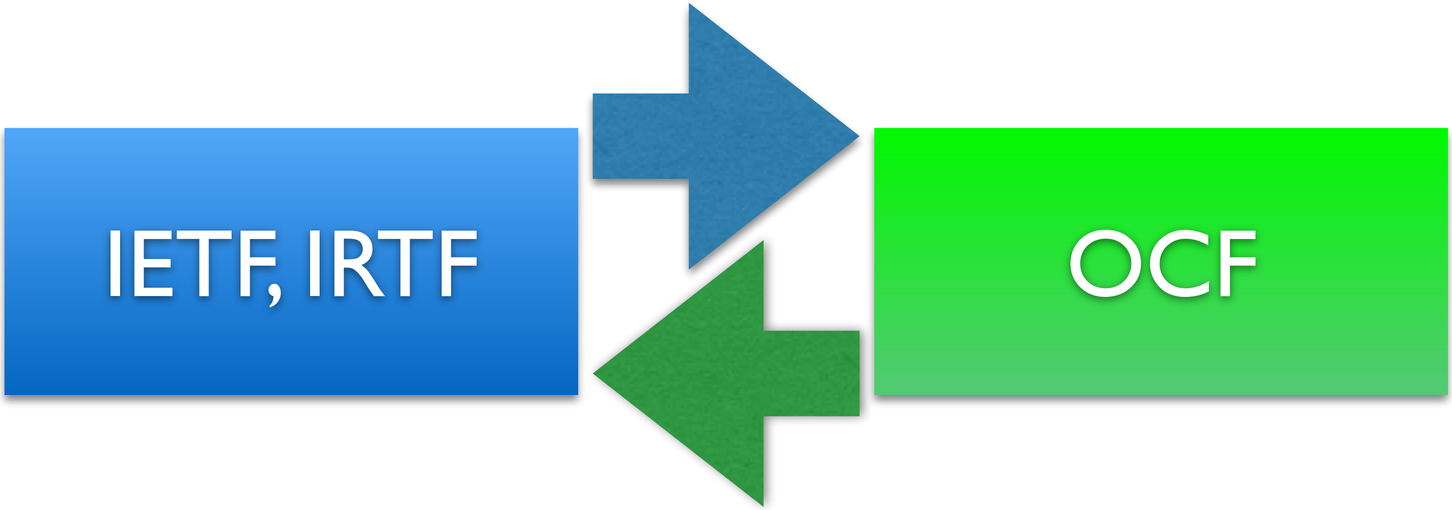
How?

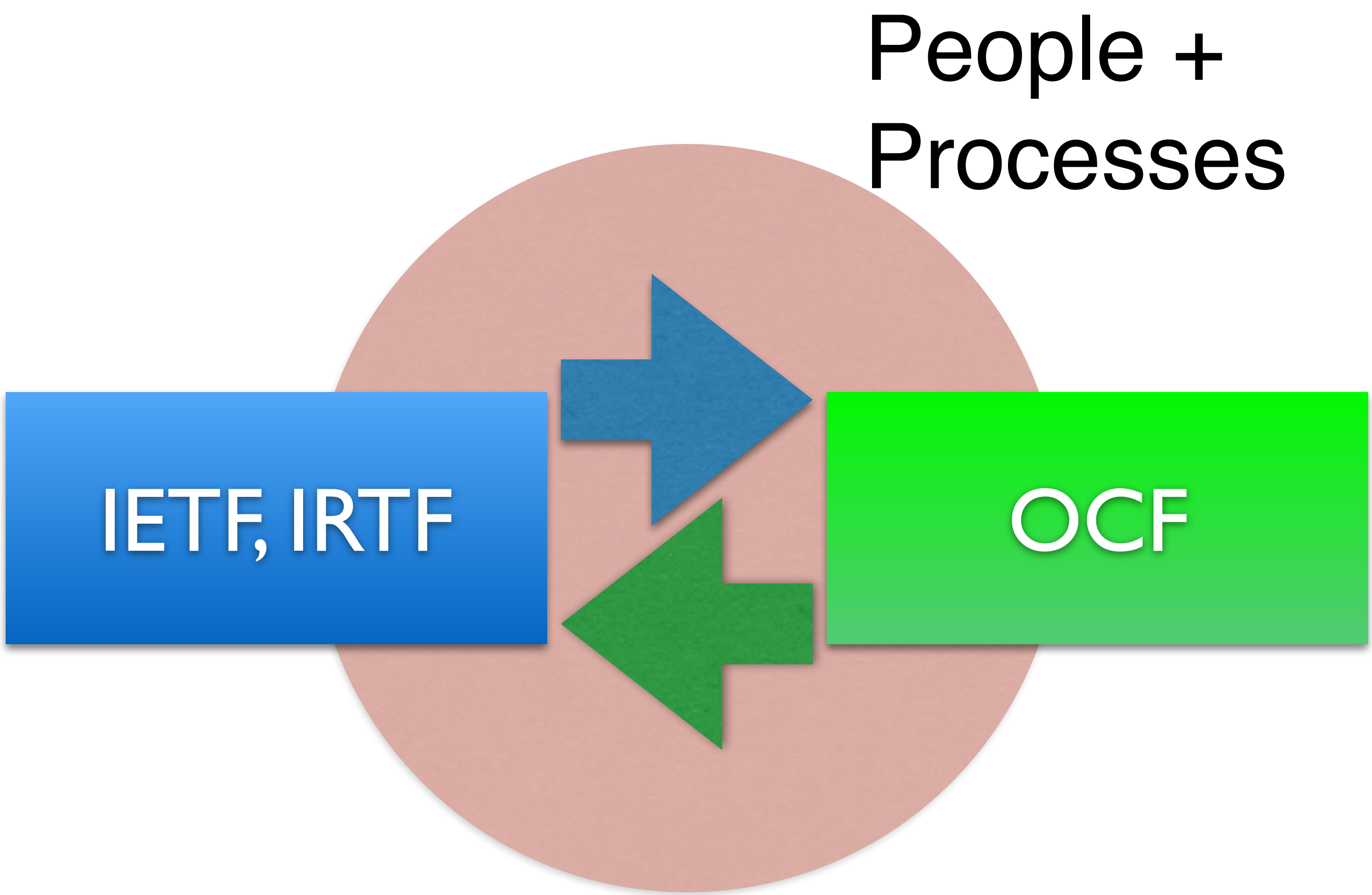


17

[Source: Kovatsch/Hartke]

IETF, IRTF → OCF


IETF, IRTF
OCF


People +
Processes
IETF, IRTF
OCF