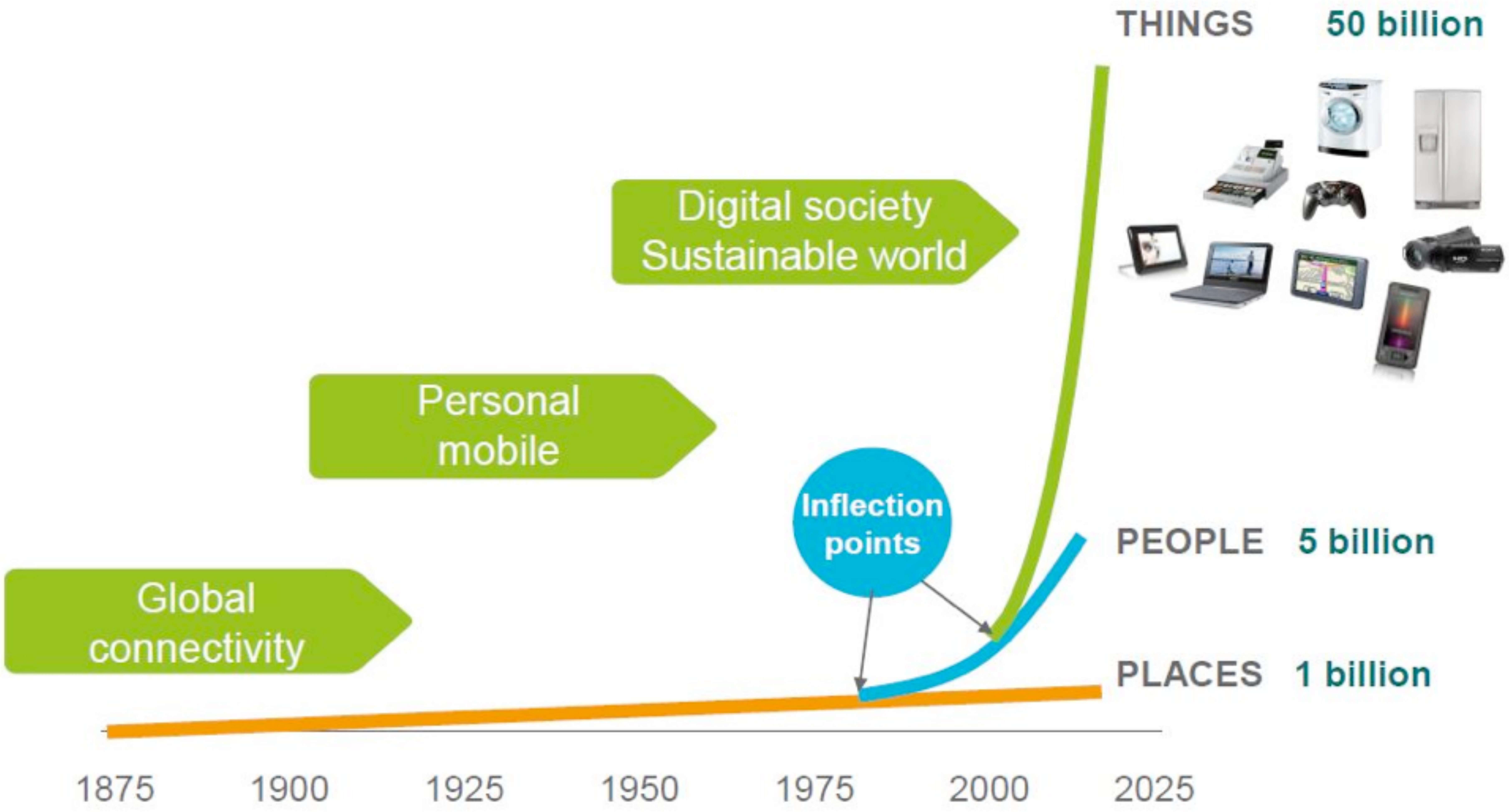
Semantics:

What do you mean? (1)

Carsten Bormann, 2019-11-15
T2TRG/W3C WoT workshop, Singapore, SG

Connecting:
Places → People → Things



Source: Ericsson

Scale up:

Number of nodes

(NNN billion by 2025)

Scale down:

node

What is different in IoT?

- Scale: **pet → cattle**
- Unattended, often unmanaged
- Lifetime of decades; updates may be difficult
- Many more **stakeholders**

“Thing” (as in IoT)

- Device with a **physical presence**, physical object
- Provides a digital **interface** to interact with that physical presence
- **Not**: creatable/destroyable in software only (digital objects)
- (We generally exclude devices the physical aspects of which pertain to digital functions only, e.g. routers, storage devices, CPUs, although these do have *some* **thingness**.)

Thing: Self-Description

- Self-Description: Assets (Devices), Resources make available enough information to use them without a manual and without “intelligent guessing”

- (This may employ **links** to additional digital objects/resources.)

Intrinsic vs. Extrinsic Information

- (Looking for a better pair of terms)
- **Intrinsic**: Information about a device itself, its nature
  - Has sensors for temperature, humidity, CO2 concentration
- **Extrinsic**: Information about its role and context
  - Sits on the front porch to measure outside air
- Somewhat floating between these: **configuration** information that is known to a device but really extrinsically motivated (e.g., IP address)

Instance vs. class

- Devices are rarely one-of-a-kind; usually mass produced
- All copies (**instances**) of these devices share some intrinsic information
- Efficient to factor out into a **class**

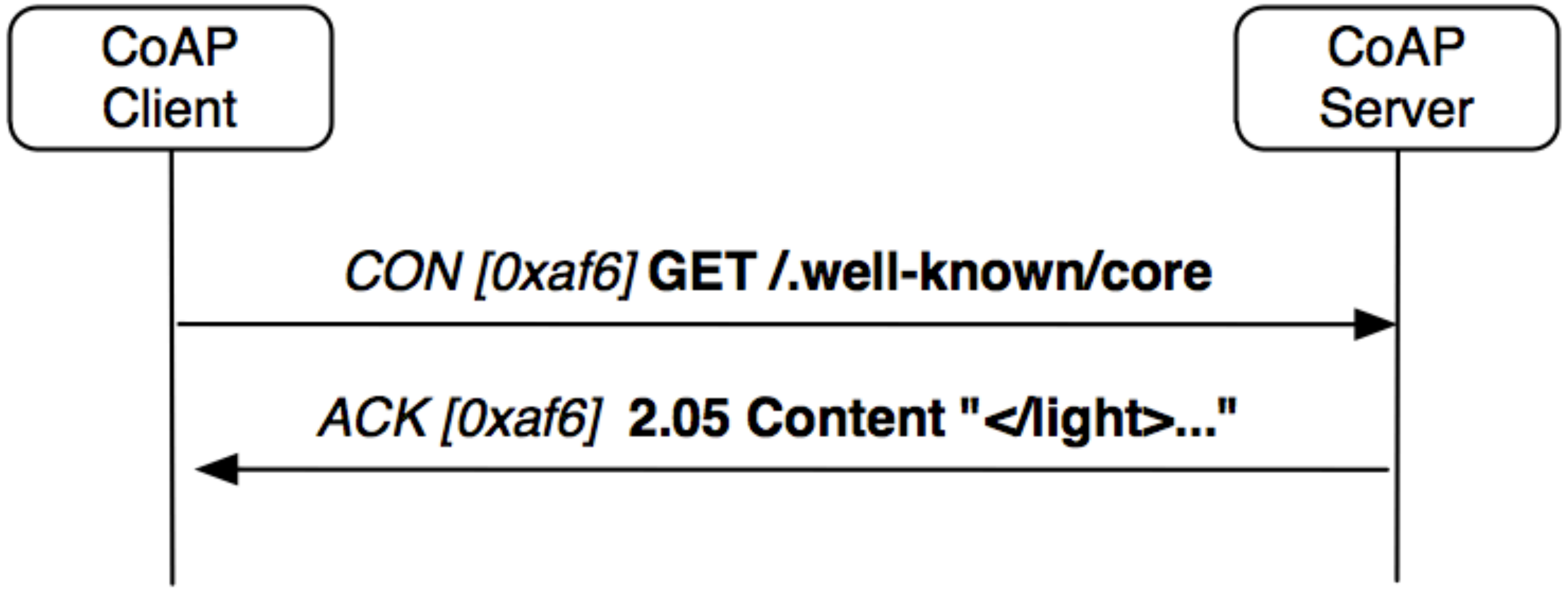
Hypermedia

- **Resources** (“media”) offered by **servers**
- Can contain **links**
    - Special kind of link: **form** (construct parameters)
- **Client** decides how to navigate this offering (“non-linear”): what media to obtain or effects to cause — cf. REST “HATEOAS” (Hypermedia as the engine of application state)

Resource Discovery: Link-Format

- Idea: Describe a Thing as a set of **Resources**
  - **REST:** Resources are remotely accessible sub-objects of a device (acting as a **server**)
- Resource Discovery with **CoRE Link Format**
  - Web linking as per RFC 5988 (8288)
  - Discovering the links hosted by CoAP servers
  - `GET /.well-known/core`
  - Returns a link-header style format
    - URL, type, interface, content-type etc.
- Standardized in ***RFC 6690***

Resource Discovery


CoAP Client
CoAP Server
CON [0xaf6] **GET /.well-known/core**
ACK [0xaf6] **2.05 Content "</light>..."**

```
</s/temp>;rt=“Temperature”;if=“Sensor”;ct=0,
</s/light>;rt=“Illuminance”;if=“Sensor”;ct=60
```

Semantic Soup?

- Where do the **terms** “Temperature” and “Illuminance” come from? (*Semantics*)
  - How is this “**machine readable**” (beyond the trivial sense)?
- A representation needs a **format** (JSON/CBOR etc.), and some **meaning**
- Today: often specific to deployment or industry
  - oBIX, SensorML, EEML etc.

→ What can we make universal/**reusable**?

→ And what needs to be market specific?

→ How do we enable **innovation**?

This is complicated.

Where is my **layer** structure?

Interoperability

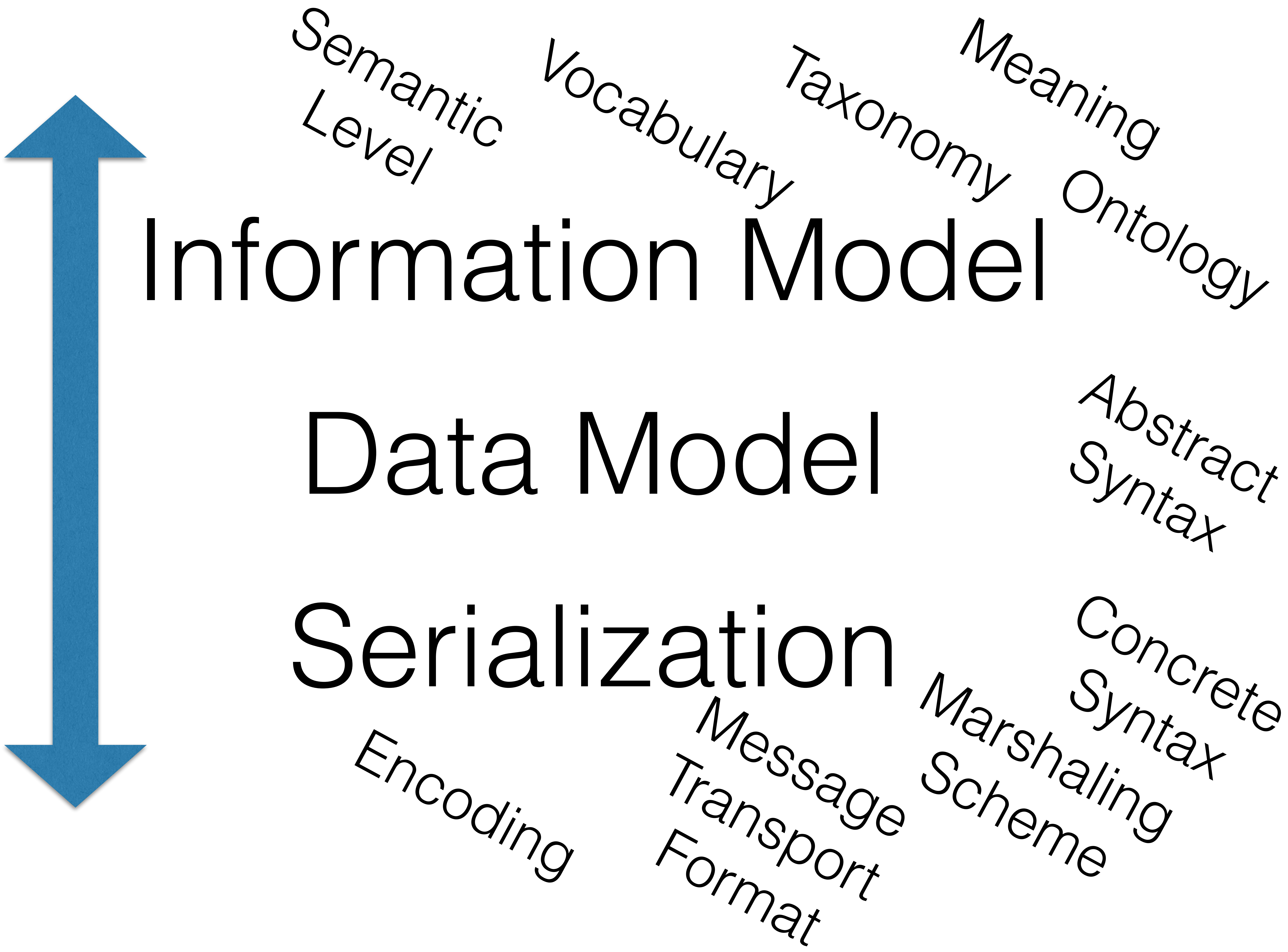
- **Semantic** Interoperability
— I understand what the data/actions **mean**

- **Structural/**syntactic Interoperability
— I understand the **structure** of the data/actions

- **Syntactic/**lexical Interoperability
— I can parse/generate the **bytes** of data/actions

Models

- A way to represent a **machine-readable** description of the self-description information
    - (Best case; often really just “read a manual”)
- Words also sometimes used:
    - Schema (originally from databases, often understood to include semantics and not just structure)
- Note that there usually needs to be a Model for the Models (Metamodel) — easy to confuse these



Information Model

Data Model

Serialization

Semantic Level

Vocabulary

Taxonomy

Meaning

Ontology

Abstract Syntax

Concrete Syntax

Marshaling Scheme

Message Transport Format

Encoding

Layering may be recursive

- E.g., within structural interoperability, there may be
    - Information models (more semantic)
    - Data models (more structural)
    - Generic data models/serialization frameworks (more syntactic)

Representation frameworks

- Modeling languages (optional!):
  ASN.1, W3C Schema/Relax-NG, ____, CDDL…
- **Generic Data Models** (what can you represent):
  Base types, Containers, …
- **Serializations** (BER, XML/EXI, JSON, CBOR)

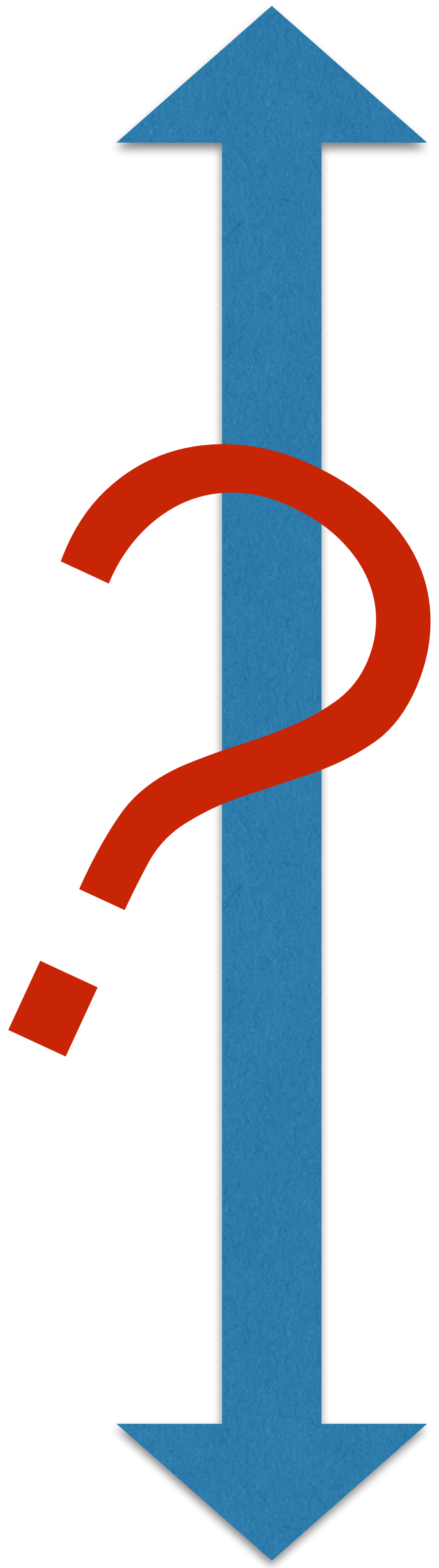
RDF: Resource Description Framework

- Extremely simple data model: set of **triples**, Subject Predicate Object, each a **statement**
  - Items can be literals, IRIs, or “blank nodes”
  - Information model: labeled, directed multi-**graph**
- Half a dozen **serialization** formats (RDF/XML, JSON-LD, Turtle, N3, …), none dominant
- Tools like GRDDL (extracting RDF from XML), SPARQL (SQL-like query language), SHACL (validation/description)
- Can add languages on top, e.g. RDFS, OWL for developing **ontologies** — constraints on sets of individuals ("classes") and the types of relationships permitted between them.

Data/Information Models

vs.

Interaction Models



Semantic Interaction Model

— know what the interactions **mean**

Structural Interaction Model

— know how to **structure** interactions

Protocol Mapping

— can **send/receive** interactions over the wire

Interaction Patterns

- **Property**: Can **retrieve** information/observe it; sometimes also **set** it (somewhat atomically)

- **Action**: Can somehow initiate, control, and abort **effects** (a.k.a. Command in Bluetooth)

- **Event**: ??? Something about time series, or maybe a sequence/collection of discrete happenings? Commands vs. indications? ➔ Telemetry, Alarms, …

- Actually, Interaction Patterns can be much more complex (e.g., how do they *combine*?)