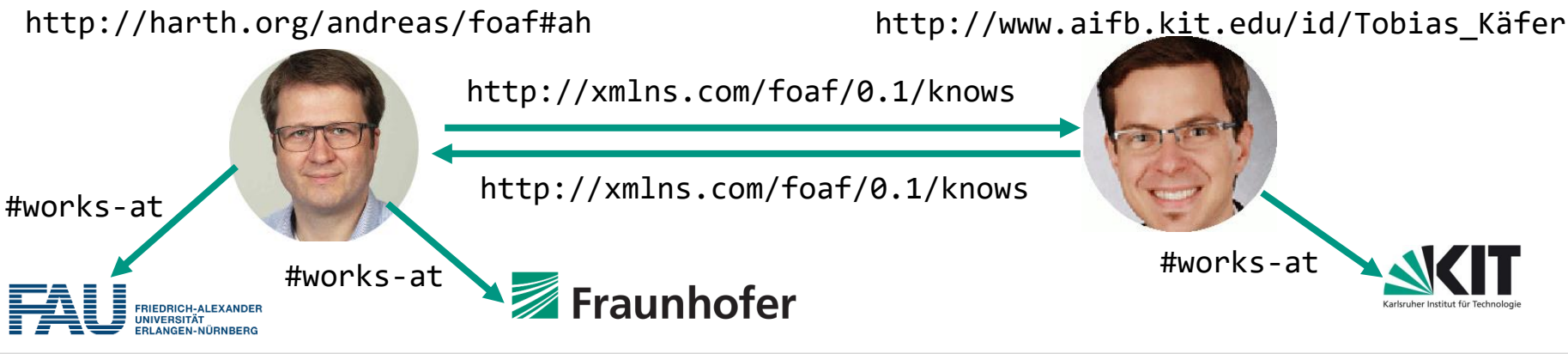
Linked Data Techniques for the Web of Things

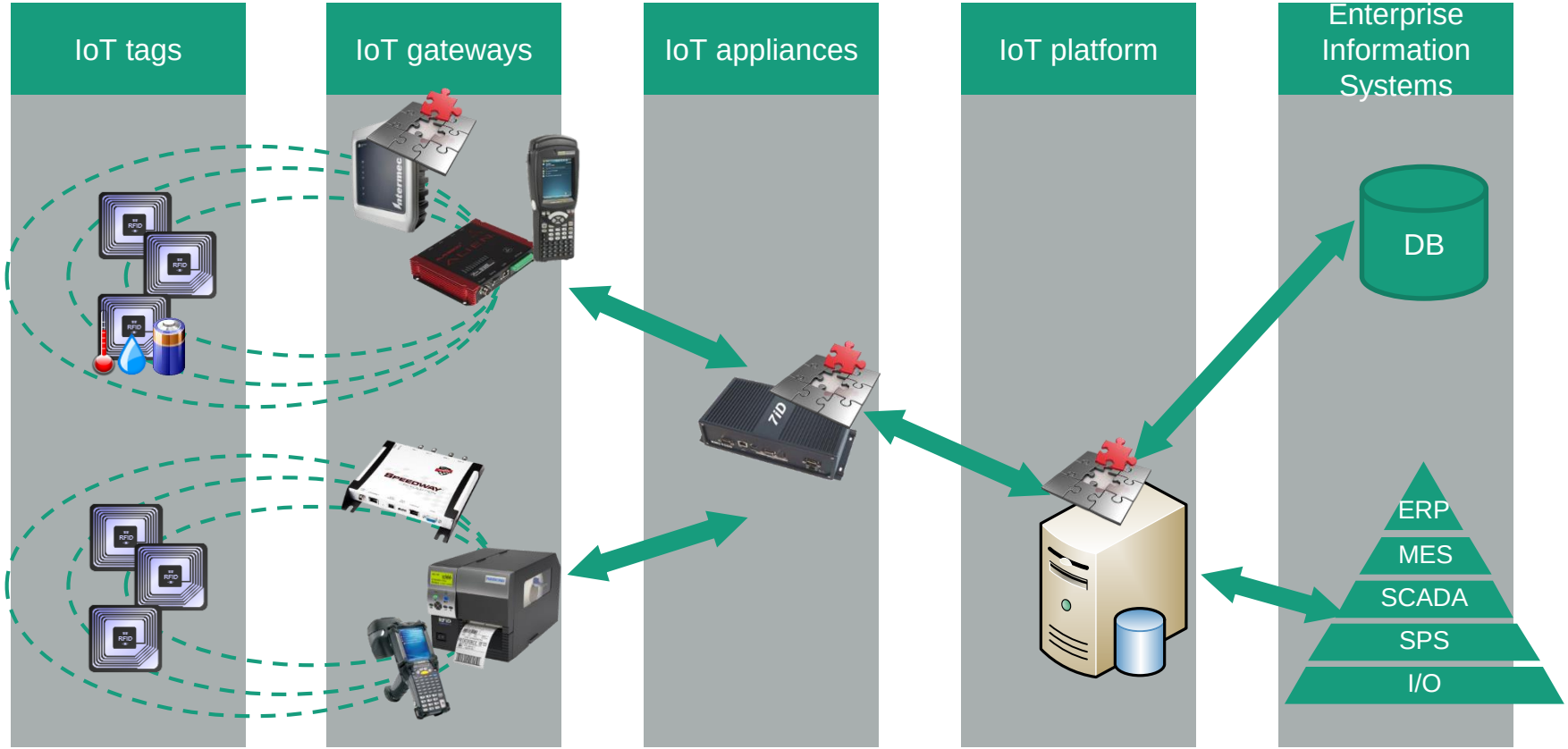
Tutorial at the 8th International Conference on the Internet of Things (IoT 2018)

Andreas Harth (FAU, Fraunhofer IIS-SCS) and Tobias Käfer (KIT)



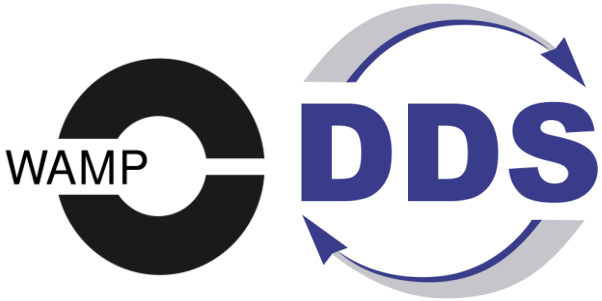
INTRODUCTION

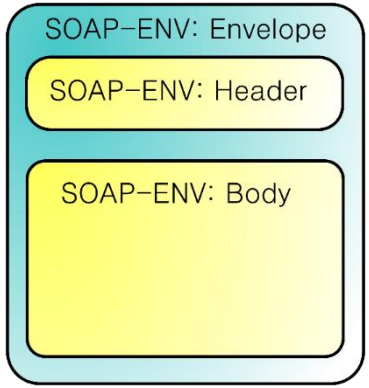
Classic IoT System Architecture



Lempert, S., 2015. *RFID-Middleware. Metastudie und Marktübersicht: Grundlagen, Anbieter und Produkte aus dem proprietären und Open Source-Umfeld*. Stuttgart: Fraunhofer Verlag. ISBN 978-3-8396-0915-6

Which Network Protocol?





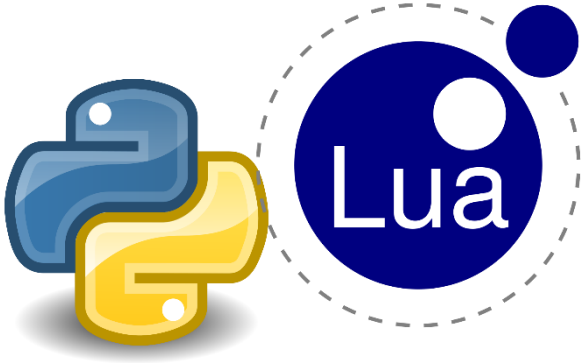
Which Data Model? Which Data Format?

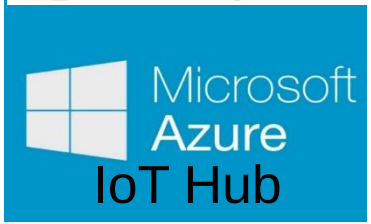
RMI

And How to Specify and Run Applications?

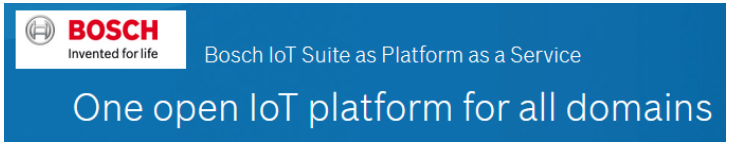
BPEL

BPMN



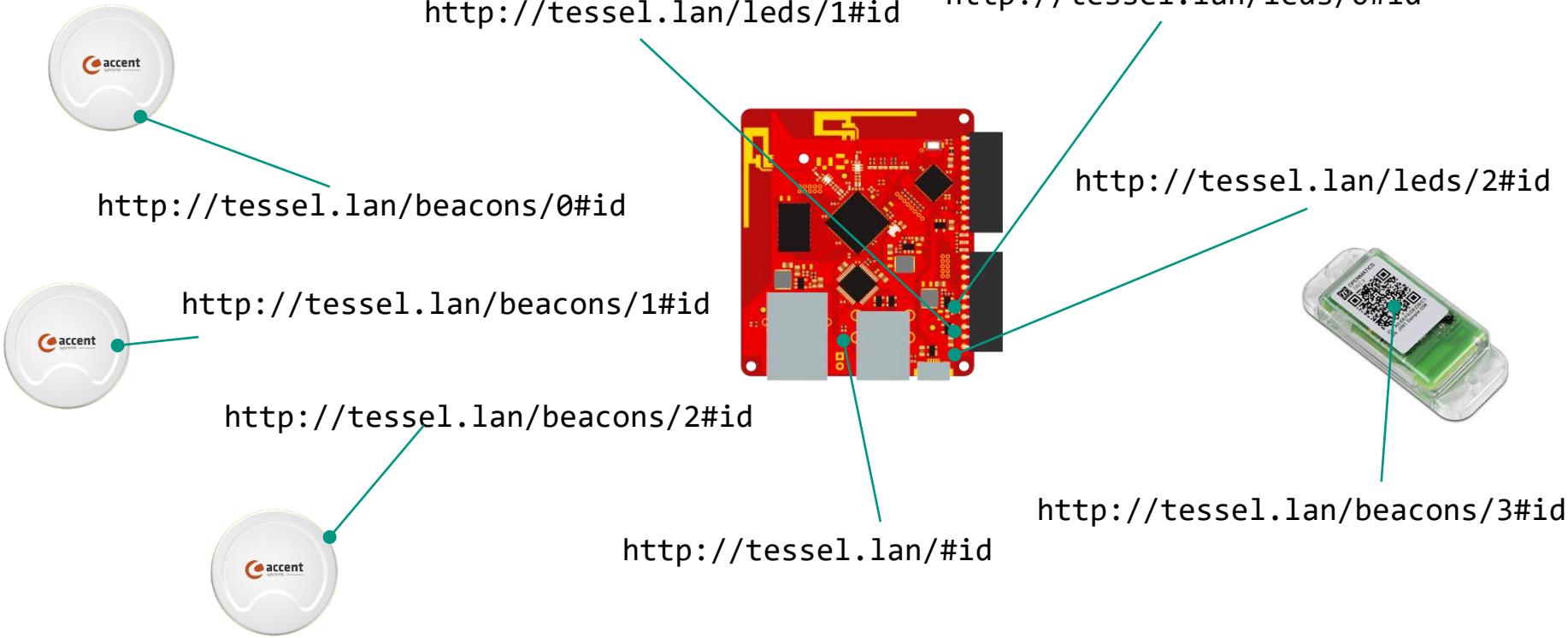


ASM



ECA

Basic Idea 1: HTTP URIs for Sensors and Actuators



Basic Idea 2: Use Linked Data for Data Access

- HTTP for data access and manipulation
- Data providers are HTTP servers, data consumers are HTTP user agents
- Semantic Web languages (RDF, RDFS, a bit of OWL) for data representation and integration
- SPARQL for querying

Basic Idea 3: Specify Application Behaviour Declaratively

- Using rules
  - IF condition THEN assertion (derivation rules)
  - IF condition THEN request (request rules, a variant of production rule)
- Using workflows (layered on top of the rules layer)
  - Using Sequence, Parallel, Conditional
- Run in decentralized settings

Overall Goal of the Tutorial

“The tutorial covers web technologies (URIs, HTTP, RDF)
for specifying and executing applications (Recipes, Applets)
involving networked sensors and actuators (Internet of Things)
based on a logical representation of world state (knowledge representation)
and application behaviour.” (dynamical systems)

Goals of the Tutorial

- We build on the basic notions of web architecture
  - HTTP URIs, Request/Response
  - Manipulation of resource state
  - Hyperlinks
- Our user agents operate on resources
  - Current resource state („now“), no distinction between „static“ and „dynamic“ data
- We show how to specify user agent behaviour
  - Declarative queries and rules, no imperative programming
  - Simple language can be basis for visual programming language
- Overall: we value simplicity to achieve web-scale
  - Focus on execution of application behaviour on standardised interfaces
  - No protocol mappings, not syntax mappings, no artificial intelligence planning

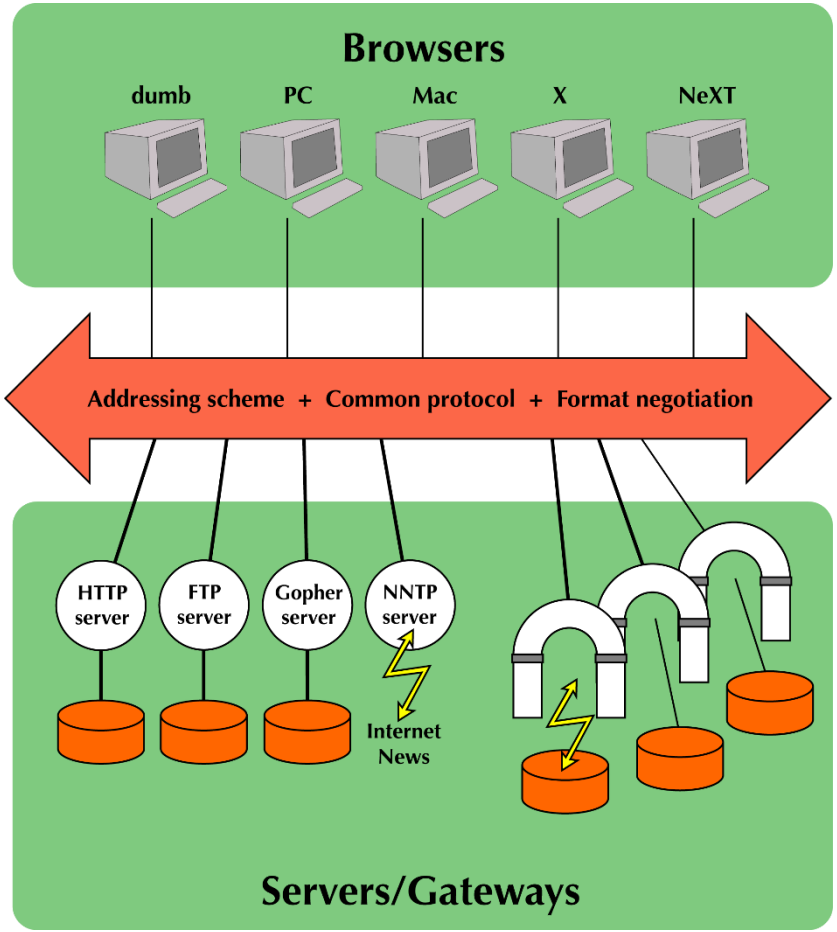
Agenda – Part I (Andreas)

- Introduction
- **The Four Linked Data Principles**
- Vocabularies and Ontologies
- State vs. Event
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

Web Architecture

- URI: RFC 1630 (1994), now RFC 3986
- HTTP: RFC 1945 (1996), now RFC 7230, 7231, 7232, 7234, 7235



https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png
Redrawn from an image from 1990

Servers and User Agents

- Client

“A program that establishes connections for the purpose of sending requests.”

- User Agent

“The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.”

- Server

“An application program that accepts connections in order to service requests by sending back responses.”

“Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. […]”

All definitions from RFC2616 (obsolete)

Uniform Resource Identifiers (RFC 3986)

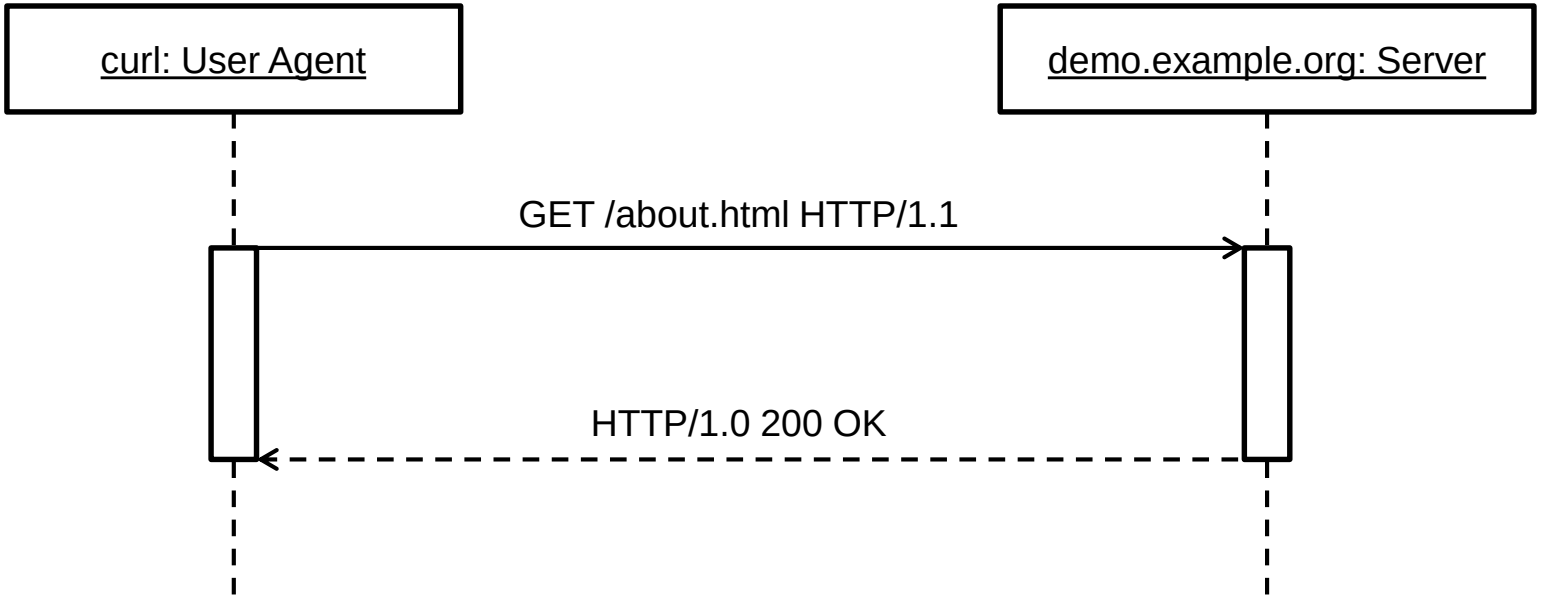
- Uniform
- Resource

“This specification does not limit the scope of what might be a resource; rather, the term ‘resource’ is used in a general sense for whatever might be identified by a URI.”

- an electronic document
- an image
- a source of information with a consistent purpose (e.g., ‘today's weather report for Los Angeles’)
- a service (e.g., an HTTP-to-SMS gateway)
- a collection of other resources
- the operators and operands of a mathematical equation
- the types of a relationship (e.g., ‘parent’ or ‘employee’)
- numeric values (e.g., zero, one, and infinity)

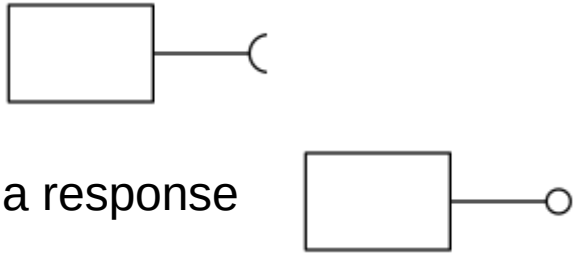
- Identifier

Request/Response in HTTP



Request/Response and User Agent/Server

- REST assumes request/response communication pattern between components with client connector and server connector
  - Clients emit requests, receive response
  - Servers answer to incoming requests with a response



Linked Data

- Postulated by Tim Berners-Lee in 2006.

> *"The Semantic Web isn't just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With linked data, when you have some of it, you can find other, related, data."* [1]

- Collection of principles governing the publication and consumption of data on the web
- Aim: unified method for describing resources and accessing the state of resources
- Later we shall see how to manipulate (change) the state of resources

[1] http://www.w3.org/DesignIssues/LinkedData.html

Linked Data Principles

1. Use URIs as names for things

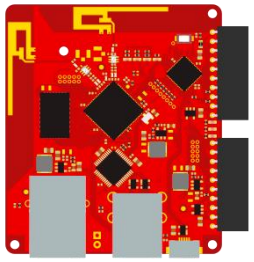
2. Use HTTP URIs so that people can look up those names.

3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

4. Include links to other URIs. so that they can discover more things.

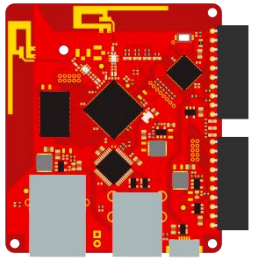
http://www.w3.org/DesignIssues/LinkedData.html

1 and # 2 Resources and URIs
Tessel and Beacons



Resource	URI
The Tessel device	`http://tessel.lan/#id`
The first beacon	`http://tessel.lan/beacons/0#id`
The second beacon	`http://tessel.lan/beacons/1#id`
The third beacon	`http://tessel.lan/beacons/2#id`
The fourth beacon	`http://tessel.lan/beacons/3#id`
The first LED	`http://tessel.lan/leds/0#id`
The second LED	`http://tessel.lan/leds/1#id`
The third LED	`http://tessel.lan/leds/2#id`

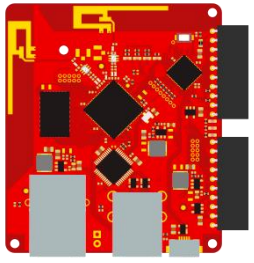
3: Provide Useful Information in RDF



- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/0#id` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/0#id> rdf:type :Sensor ;
                :hasProperty <#prop1> .
<#prop1> rdf:type :ObservableProperty ;
       :measure :RSSI ; :value "-59" ; :unit "dbm" .
```

3: Provide Useful Information in RDF



- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/3#id` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/0#id> rdf:type :Sensor ;
                :hasProperty <#prop1> , <#prop2> .
<#prop1> rdf:type :ObservableProperty ;
       :measure :RSSI ; :value "-52" ; :unit "dbm" .
<#prop2> rdf:type :ObservableProperty ;
       :measure :temperature ; :value "23" ; :unit "Celcius" .
```

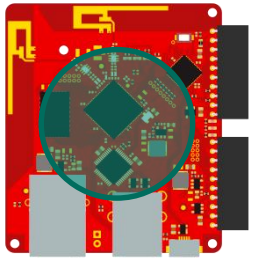
Resource Description Framework (RDF)

- A graph-structure data format (a formal language) for knowledge representation
- Subject-predicate-object triples
- URIs and literals to name resources; blank nodes as “variables”
- Graph structure enables merging of RDF graphs from multiple sources
- Different serialization syntaxes for the triple structure (e.g., XML, JSON)
- We use Turtle syntax to encode RDF triples and graphs
  - @prefix for compact URIs (CURIEs)
  - <> for URIs, e.g., <http://tessel.lan/beacons/0#id>
  - "" for literals, e.g., "23"
  - _: for blank nodes, e.g., _:bn
  - One triple per line, with . at the end
  - , to repeat subject and predicate, ; to repeat subject

RDF Abstract Syntax

Definition (RDF Terms, RDF Triple, RDF Graph) *The set of RDF terms consists of the set of URIs $\mathcal{U}$, the set of blank nodes $\mathcal{B}$ and the set of RDF literals $\mathcal{L}$, all being pairwise disjoint. A tuple $\langle s, p, o \rangle \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple, where s is the subject, p is the predicate and o is the object. A set of triples is called RDF graph.*

4: Links to other URIs



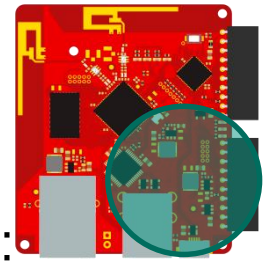
- A User Agent performing a HTTP GET on `http://tessel.lan/` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/woto#> .

<#id> rdf:type :Platform ; rdfs:comment "Andreas' Tessel2" ;
      :hosts <beacons/#id> , <leds/#id> .
```

- A HTTP GET on `http://tessel.lan/beacons/` leads to:

```
@prefix : <http://example.org/woto#> .

<#id> :hosts <0#id> , <1#id> , <2#id> , <3#id> .
```

Read-Write Linked Data



- A User Agent performing a HTTP PUT on `http://tessel.lan/leds/0` with the following:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</leds/0#id> rdf:type :Actuator ;
             :hasProperty <#prop1> .
<#prop1> rdf:type :ActuatableProperty ;
       :onOffStatus :on .
```

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- **Vocabularies and Ontologies**
- State vs. Event
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

Overview

- Vocabularies and ontologies contain a set of terms (URIs) with an agreed meaning within a community
- We consider the following vocabularies for describing devices (sensors, actuators):
  - Semantic Sensor Network Ontology (SSN)
  - Web of Things (WoT) Thing Descriptions (TD)
- We consider the following vocabularies for describing units and measurements:
  - Units of Measure, Quantity Kinds, Dimensions and Types (QUDT)
  - The Unified Code for Units of Measure (UCUM)
- Schema.org IoT is looking to combine existing vocabularies

Semantic Sensor Network Ontology

- Designed for recording historical observation results and actuation results (including scientific measurements)
- Comprehensive, large ontology:
  - Deployment, System, SystemProperty
  - Condition, Feature, Procedure, Result
  - Observation/Actuation/Sampling
- The act of sensing and actuating are modelled as part of the semantic model expressed in RDF
- In REST and Linked Data, sensing and actuation could be seen as “baked in” to the HTTP protocol

Web of Things (WoT) Thing Descriptions (TD)

- Designed from a predominantly JavaScript/COAP viewpoint
- Fewer classes and properties than SSN
- Built on the idea of servients (compenents that are both user agents and servers)

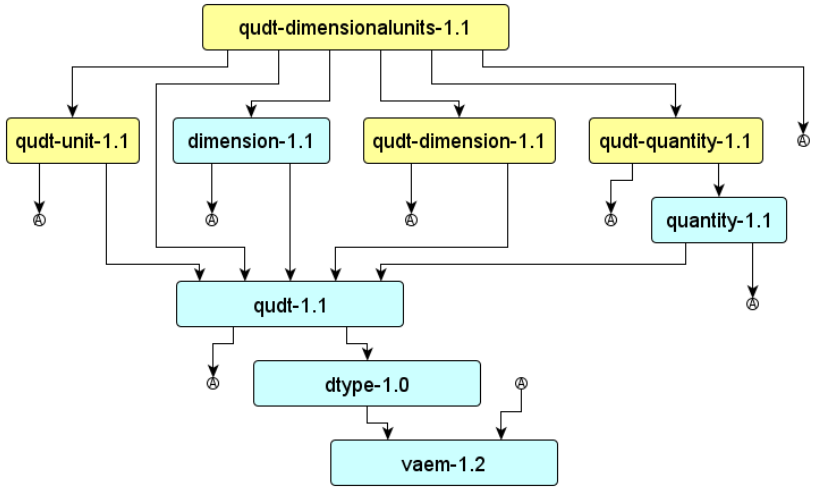
- Semantic description of the thing itself
- Basic notions for interaction: Properties, Actions, Events
- Forms, Linking

- Provides a JSON serialization (also JSON-LD)
- In WoT TD, the focus is on describing things, their interfaces and protocols
- Application behavior is scripted based on an API (object model)

Units of Measure, Quantity Kinds, Dimensions and Types (QUDT)

- Defines base units and derived units
- QUDT 1.1 is released, QUDT 2.0 in progress

- Originally from the NASA Exploration Initiatives Ontology Models (NExIOM) project
- Now managed by a non-profit organisation



http://linkedmodel.org/catalog/qudt/1.1/

The Unified Code for Units of Measure (UCUM)

- Based on the ISO 80000: 2009 Quantities and Units standards series (use of System International (SI) units)
- Focused on print publication, not on the web (no URIs)
- Restrictive license

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- **State vs. Event**
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

Motivation

- We would like to have a coherent knowledge representation
  - to be able to integrate data from different sources
  - to be able to query the integrated data
  - to have access to the current world state
- Unfortunately, there are incompatible views on knowledge representation in dynamic environments:
  - Logic-based formalisms have been developed to pick out objects and describe their properties (state of affairs)
  - REST builds on the idea of accessing and manipulating state
  - Protocols such as MQTT and ROS are based on the sending and processing of events via centralised message brokers

Protocol/Architecture

Linked Data

- Distinction between client (user agent) and server
- User agents invoke requests, servers answer with responses
- Server is persistent
- No central message broker needed

Event Processing

- No clear distinction between client and server
- All components can send messages and receive messages
- All components are persistent
- Some systems rely on a central message broker

Time

The New Media Reader, edited by Noah Wardrip-Fruin and Nick Montfort, book design by Michael Crumpton, 2003, http://www.newmediareader.com/, 50. Time Frames (from Understanding Comics), Scott McCloud, 1993

Aspect: State vs. Event

- State: empty

- Event: Leonard Cohen is born on September 21, 1934

```
[] a :Event ; :type :birth ; :date "1934-09-21" .
```

- State: Leonard Cohen

```
:lc :birthDate "1934-09-21" .
```

- Event: Leonard Cohen dies on November 7, 2016

```
[] a :Event ; :type :death ; :date "2016-11-07" .
```

- State: Leonard Cohen

```
:lc :birthDate "1934-09-21" .
:lc :deathDate "2016-11-07" .
```

https://www.youtube.com/watch?v=YD6fvzGIBfQ

Aspect of Data Representation

Linked Data

- Messages represent resource state
- Getting the combined current state of resources involves GETs on URIs and merging responses

Event Processing

- Messages represent events, but some messages represent state as well
- Getting the combined current state of resources requires integration of events

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- State vs. Event
- **System Architecture**
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

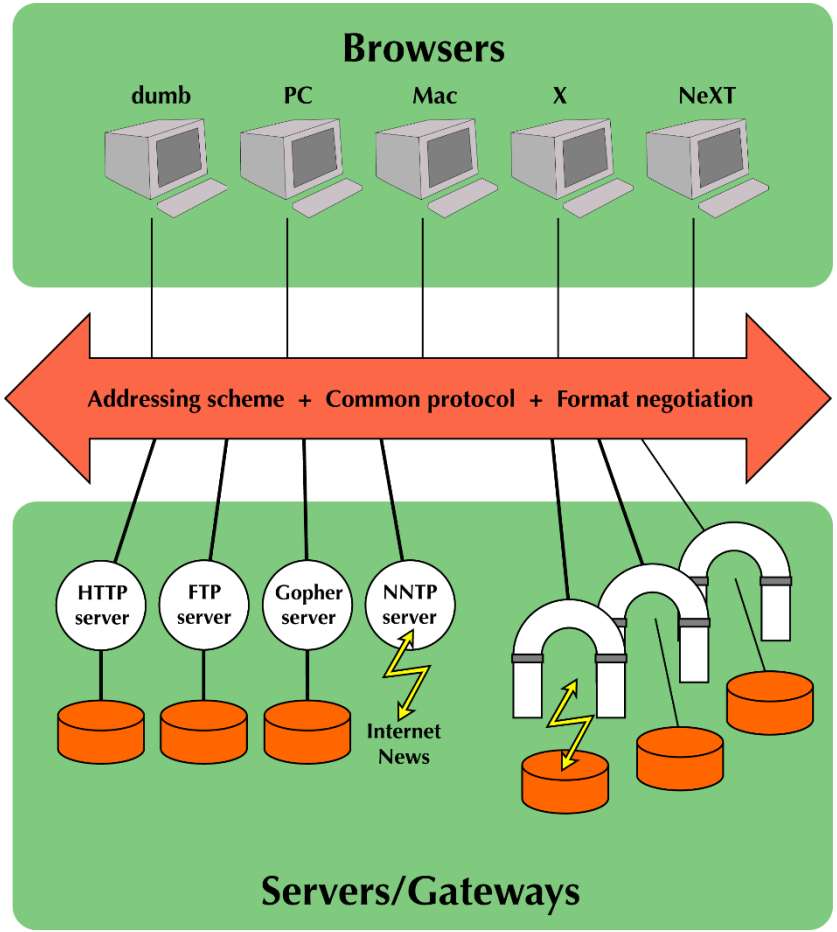
William of Ockham, 1287-1347

system architecture

The simplest ~~explanation~~ is usually the correct one.

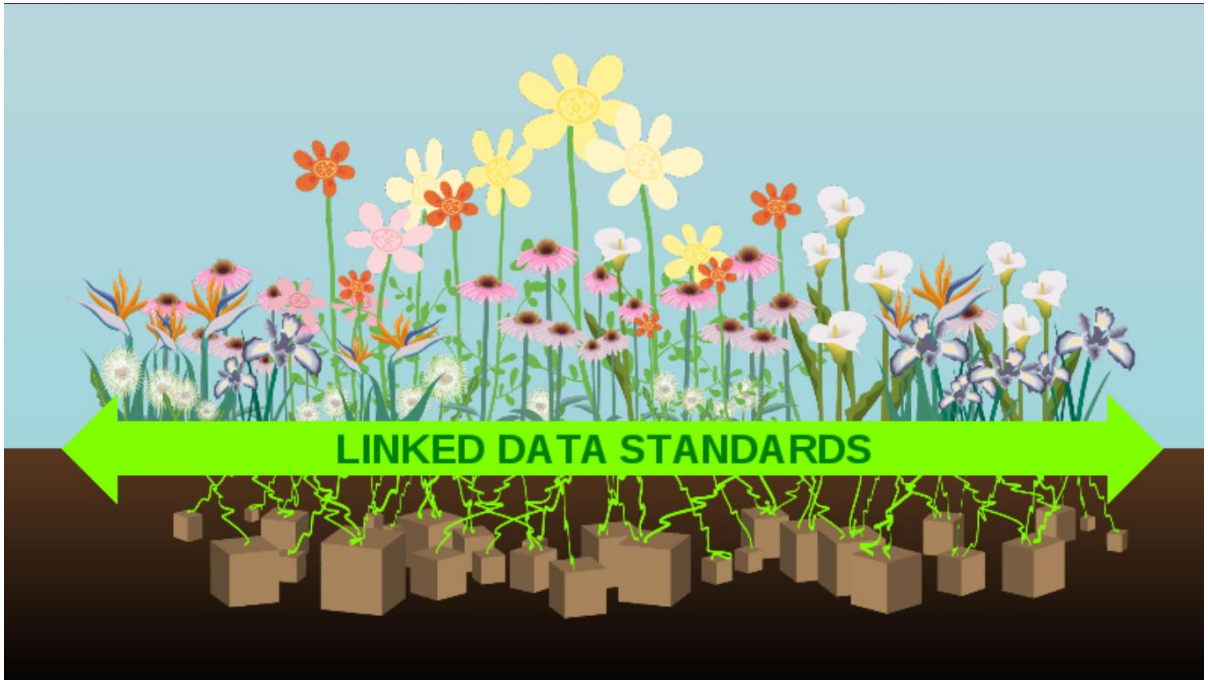
Web Architecture

- URI: RFC 1630 (1994), now RFC 3986
- HTTP: RFC 1945 (1996), now RFC 7230, 7231, 7232, 7234, 7235



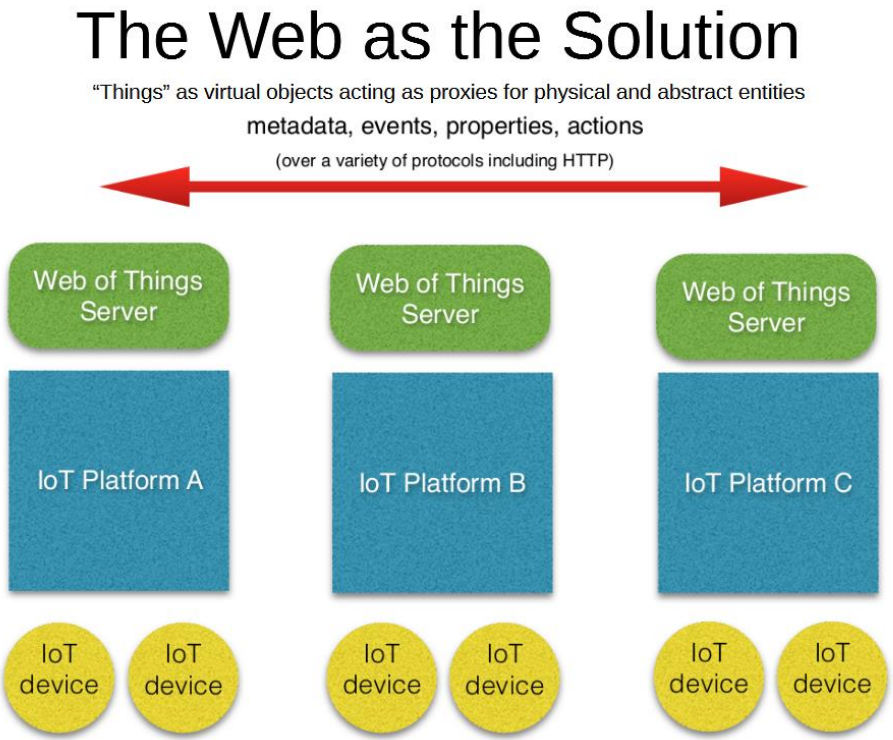
https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png
Redrawn from an image from 1990

Linked Data Architecture (2009)



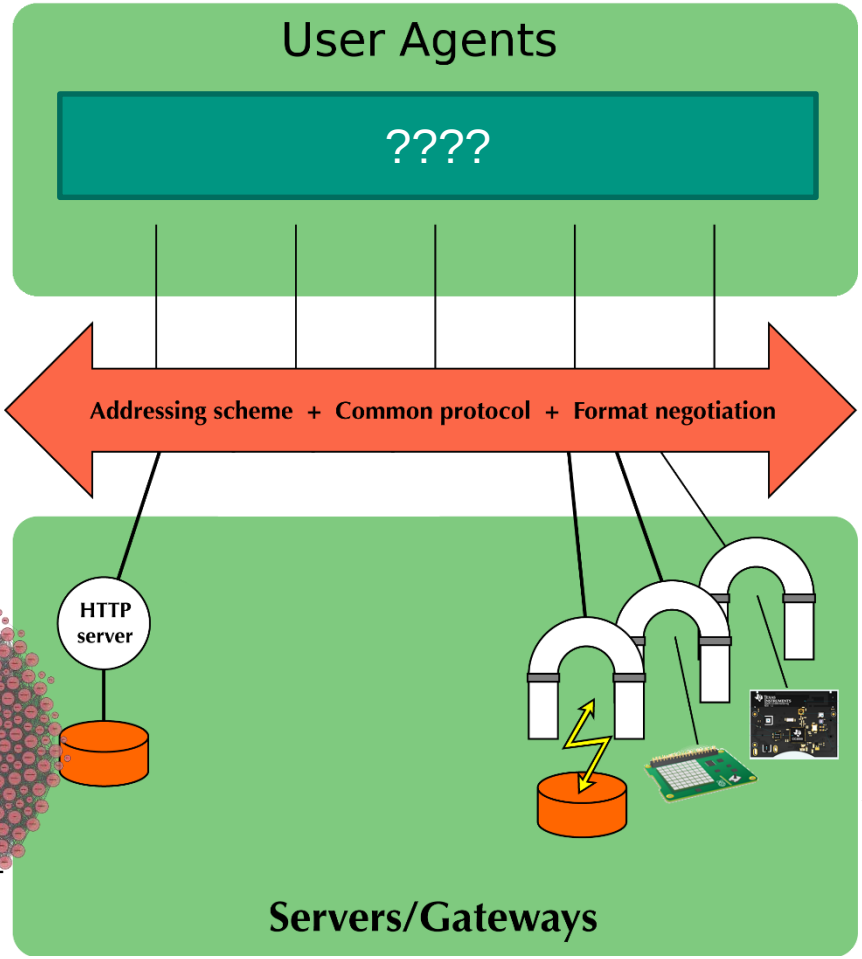
https://www.w3.org/2009/Talks/0204-ted-tbl/#(7)

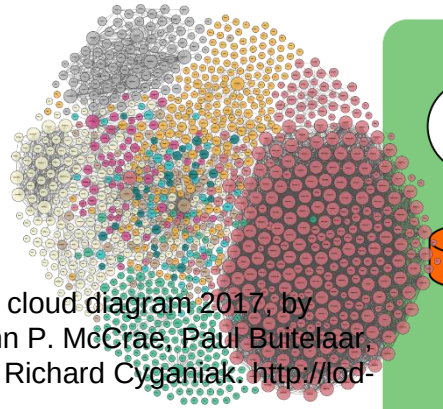
Web of Things Architecture (2015)



https://www.w3.org/2015/05/wot-framework.pdf

Web of Data/ Web of Things Architecture





Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

Adapted from https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png

Examples of Linked Data User Agents

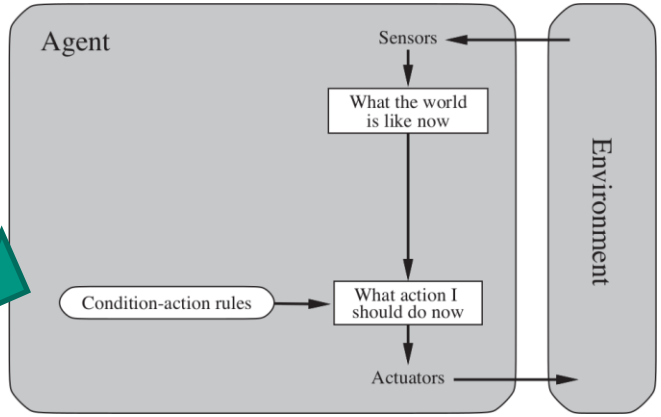
- Temperature recorder (for temperature of room, building, area…)
- Temperature display (for temperature of room, building, area…)
- Thermostat (for room temperature)
  - Setting temperature depending on who is in the room
- Heat alarm (for temperature of room, building, area…)
- Distance alarm (for theft detection)
- … (your ideas?)

- WoT „Recipes“ IFTTT „Applets“ provide similar functionality

Cognitive Architectures

- SOAR (initially: State, Operator, Apply, Result),
- ACT-R (Adaptive Control of Though – Rational)
- Goal: to create „intelligent agents“
- In the tutorial, we only consider user agents that are
  - „simple reflex agents“ (Russel & Norvig, see figure),
  - aka „tropistic agents“ (Genesereth & Nilson)
- We explain how to use rules to control the agent‘s behaviour
- Part I: safe HTTP methods (GET)
- Part II: unsafe HTTP methods

Russel and Norvig, Artificial Intelligence – A Modern Approach, Third Edition, 2010



User Agents

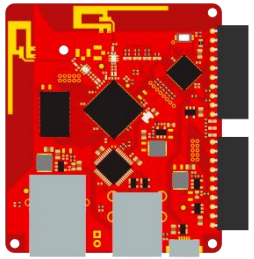
- We start with user agents that access world state and run queries over the combined world state
- Then, we consider user agents that perform reasoning over the world state for data integration
- Next, we consider user agents that follow links to discover new resources, combine state of different resources with reasoning and then run queries over the result
- Tobias will talk about user agents that are able to change the state of resources (actuators)

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- State vs. Event
- System Architecture
- **Query Evaluation on RDF Datasets (Principle #3)**
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

SPARQL Query



- Return the value and unit of the observation from the beacons:

```
PREFIX ssn: <http://www.w3.org/ns/ssn/>

SELECT ?x ?y
FROM <http://tessel2.lan/beacons/0>
FROM <http://tessel2.lan/beacons/1>
FROM <http://tessel2.lan/beacons/2>
WHERE {
   ?x ssn:hasValue ?y .
}
```

SPARQL FROM and FROM NAMED

Definition (Named Graph, RDF Dataset) *Let $\mathcal{G}$ be the set of RDF graphs and $\mathcal{U}$ be the set of URIs. A pair $\langle g, u \rangle \in \mathcal{G} \times \mathcal{U}$ is called a named graph. An RDF dataset consists of a (possibly empty) set of named graphs (with distinct names) and a default graph $g \in \mathcal{G}$ without a name.*

We assume the graph names are also addresses to locations with RDF documents.

Dave Beckett‘s roqet

- SPARQL processors operate on RDF datasets
- Many SPARQL processors operate on local RDF datasets
- A SPARQL „database“, to which you first import your data and then pose queries

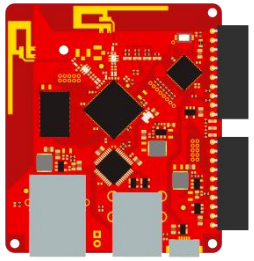
- There are query processing user agent
- E.g., roqet (http://librdf.org/rasqal/roqet.html) (for years)
- Roqet dereferences the URIs of the graphs in the RDF dataset during query time
- Possible to access current data and evaluate a query over the current data

Algorithm Agent Loop

```
1  input: integer delay
2  output: number of iterations
3  t := 0
4  while termination condition/criteria not true:
5              access data, evalute queries...
6              t := t + 1
7              output results (datasets, query results, request/response information...)
8              wait delay milliseconds
9  return t
```

.

SPARQL Query



- Return true if beacon 0 is a certain distance away from the Tessel (approximating distance via RSSI value):

```
PREFIX ssn: <http://www.w3.org/ns/ssn/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

ASK
FROM <http://tessel2.lan/beacons/0>
WHERE {
   <http://tessel2.lan/beacons/0#id> ssn:hasProperty ?prop .
   ?prop ssn:hasValue ?y .
   FILTER ((xsd:integer(?y)) > -60)
}
```

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- State vs. Event
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- **Query Evaluation with Entailment (Principle #3)**
- Link Traversal Query Evaluation (Principle #4)
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

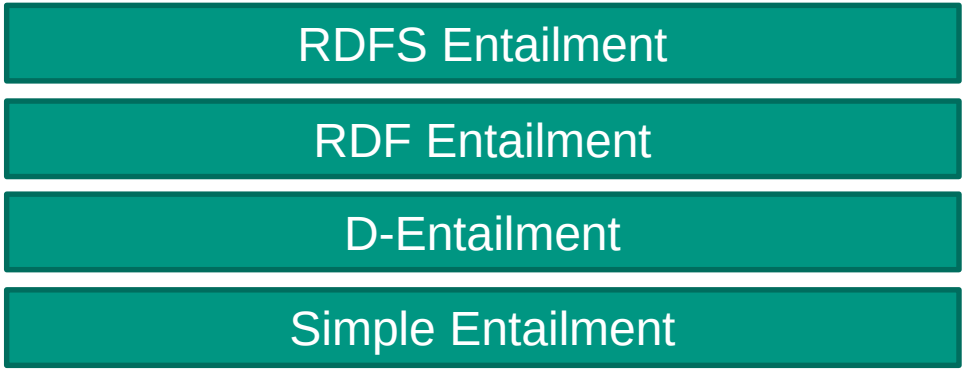
3: Provide Useful Information in RDF/RDFS

- RDF and RDFS have a model-theoretic semantics
- Requires appropriate implementation (datatypes, inference rules…)

- We could even do a bit of OWL (including owl:sameAs)

- E.g., sosa:ObservableProperty rdfs:subClassOf td:Property .

Layering of Entailment



- RDF 1.1 moved datatype entailment (D-entailment)
- Now, datatype entailment comes first, then RDF entailment, then RDFS entailment
- In the following, we present the entailment patterns for RDF and RDFS (and omit axiomatic triples)

RDF Entailment Patterns

- S is a graph, D is the set of supported datatypes (at least rdf:langString and xsd:string)
- Patterns in conjunction with RDF data model:

	if *S* contains	then *S* RDF-entails recognising *D*
rdfD1	?x ?p "sss"^^ddd .	?x ?p _:n . _:n rdf:type ddd .
rdfD2	?x ?p ?y .	?p rdf:type rdf:Property .

- Patterns in conjunction with generalised RDF data model:

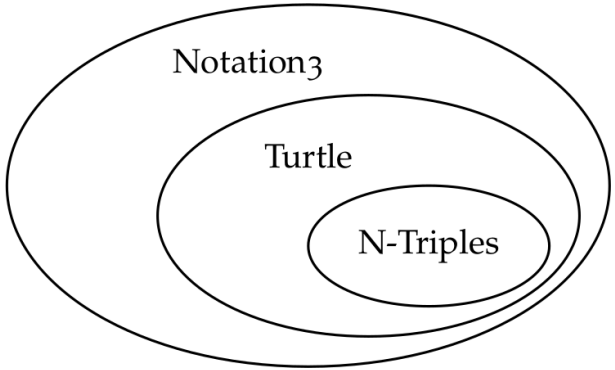
	if *S* contains	then *S* RDF-entails recognising *D*
GrdfD1	?x ?p "sss"^^ddd . for ddd in D	"sss"^^ddd rdf:type ddd .

RDFS Entailment Patterns

	if *S* contains	then *S* RDFS-entails recognising *D*
rdfs1	any URI ddd in D	ddd rdf:type rdfs:Datatype .
rdfs2	?p rdfs:domain ?x . ?y ?p ?z .	?y rdf:type ?x .
rdfs3	?p rdfs:range ?x . ?y ?p ?z .	?z rdf:type ?x .
rdfs4a	?x ?p ?y .	?x rdf:type rdfs:Resource .
rdfs4b	?x ?p ?y .	?y rdf:type rdfs:Resource .
rdfs5	?x rdfs:subPropertyOf ?y . ?y rdfs:subPropertyOf ?z .	?x rdfs:subPropertyOf ?z .
rdfs6	?x rdf:type rdf:Property	?x rdfs:subPropertyOf ?x .
rdfs7	?p2 rdfs:subPropertyOf ?p1 . ?x ?p2 ?y .	?x ?p1 ?y .
rdfs8	?x rdf:type rdfs:Class .	?x rdfs:subClassOf rdfs:Resource .
rdfs9	?x rdfs:subClassOf ?y . ?z rdf:type ?x .	?z rdf:type ?y .
rdfs10	?x rdf:type rdfs:Class .	?x rdfs:subClassOf ?x .
rdfs11	?x rdfs:subClassOf ?y . ?y rdfs:subClassOf ?z .	?x rdfs:subClassOf ?z .
rdfs12	?x rdf:type rdfs:ContainerMembershipProperty .	?x rdfs:subPropertyOf rdfs:member .
rdfs13	?x rdf:type rdfs:Datatype .	?x rdfs:subClassOf rdfs:Literal .

Notation3 Syntax

- Notation3 syntax is a superset of Turtle syntax
- N3 adds variables (prefixed with a question mark „?“)
- N3 adds graph quoting
  - Subject or object can consist of triple patterns



- Example

```
@prefix : <#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
{ ?x :p ?y } log:implies { ?y :q ?x . } .
```

Derivation Rule Abstact Syntax

Definition (Derivation Rule) *Let q be a graph pattern and g a derivation template pattern. A derivation rule is a pair $\langle q, g \rangle$ and has the form* `{` q `} => {` g `} .` *All variables of g must also occur in q (the rule is called safe).*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

# rdfD2
{ ?xxx ?aaa ?yyy . } => { ?aaa rdf:type rdf:Property . } .
```

Query Processing User Agent with Entailment (forward-chaining)

- Input: Dataset URIs, Query (as SPARQL algebra expression), Ruleset (as SPARQL algebra expression), L2V datatypes map
- Output: Query results (or error)

- Construct RDF dataset for local processing
- Materialise RDF dataset to default graph
- Evaluate SPARQL algebra expression over local (materialised) RDF dataset
- Return query results

Query Processing User Agent with Entailment (backward-chaining)

- Input: Dataset URIs, Query (as SPARQL algebra expression), Ruleset (as SPARQL algebra expression), L2V datatypes map
- Output: Query results (or error)

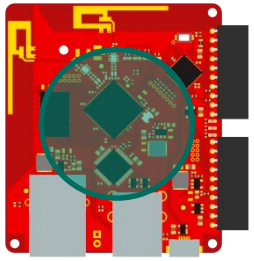
- Construct RDF dataset for local processing
- Rewrite Query to take into account Ruleset
- Evaluate SPARQL algebra expression over local RDF dataset
- Return query results

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- State vs. Event
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- **Link Traversal Query Evaluation (Principle #4)**
- Half-Time Summary

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

4: Links to other URIs (Recap)



- A User Agent performing a HTTP GET on `http://tessel.lan/` leads to:

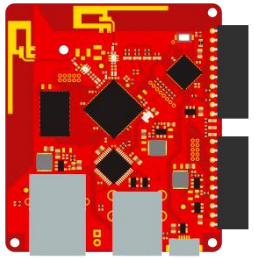
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/woto#> .

<#id> rdf:type :Platform ; rdfs:comment "Andreas' Tessel2" ;
      :hosts <beacons/#id> , <leds/#id> .
```

- A HTTP GET on `http://tessel.lan/beacons/` leads to:

```
@prefix : <http://example.org/woto#> .

<#id> :hosts <0#id> , <1#id> , <2#id> , <3#id> .
```

3: Provide Useful Information in RDF (Recap)

- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/3#id` leads to:




```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/0#id> rdf:type :Sensor ;
                :hasProperty <#prop1> , <#prop2> .
<#prop1> rdf:type :ObservableProperty ;
       :measure :RSSI ; :value "-52" ; :unit "dbm" .
<#prop2> rdf:type :ObservableProperty ;
       :measure :temperature ; :value "23" ; :unit "Celcius" .
```

Link Traversal User Agent Model

1. The user agent starts its interaction based on a specified seed URI.
2. The user agent performs HTTP requests[1] on URIs and parses the response message.
3. Based on the response, the user agent has the choice as to which URIs to dereference next.
4. The user agent decides which link(s) to follow and initiates a new request/new requests.

[1]We restrict HTTP requests to GET requests for now.

- Follow one link to arrive at a path through the web of information resources (e.g., depth-first search)
- Follow all links to arrive at a tree of information resources (e.g., breadth-first search)

Link Traversal Query Processing User Agent

- Input: Query (as SPARQL algebra expression)
- Output: Query results

- Construct RDF dataset for local processing from URIs in Query
- Evaluate SPARQL algebra expression over local RDF dataset
- Return query results

How to Specify which Links to Traverse?

- Function that maps current state to additional requests
- Implicitely encoded in the algorithm/system
- Or: encode the function using rules

Request Rule Abstract Syntax

Definition (Request Template Pattern) *A request template pattern is a HTTP request, in which the start line S consists of a tuple* $\langle M, t, V \rangle$*, where* M *is the HTTP method,* $t \in \mathcal{U} \cup \mathcal{V}$ *is the request target, which can be a URI or variable, and* V *is the HTTP version.*

Definition (Request Rule) *Let* q *be a graph pattern and* r *a request template. A request rule is a pair* $\langle q, r \rangle$ *and has the form* `{ q } => { r } .` *All variables in* r *must also occur in* q *(the rule is called safe).*

Example Traversal Rules

```
@prefix http: <http://www.w3.org/2011/http#> .
@prefix httpm: <http://www.w3.org/2011/http-methods#> .
@prefix : <http://example.org/woto#> .

{ [] http:mthd httpm:GET ;                          # access index (entry) page
   http:requestURI <http://tessel.lan/> . }

{
  ?x :hosts ?y .                                    # match :hosts triples
} => {
  [] http:mthd httpm:GET ;                          # access linked documents
     http:requestURI ?y .
} .
```

Run with Linked Data-Fu, http://linked-data-fu.github.io/, Steffen Stadtmüller, Sebastian Speiser, Andreas Harth, Rudi Studer. "Data-Fu: A Language and an Interpreter for Interaction with Read/Write Linked Data". WWW 2013, Rio de Janerio, Brasil.

Link Traversal Query Processors

- Hartig et al. ISWC 2009
  - Index nested loops joins
  - Iterator model (bolted on top of ARQ/Jena)
- Harth et al. WWW 2010
  - Repeated evaluation
  - Improve source index over time
- Ladwig and Tran ISWC 2010
  - Architecture for link-traversal query processor (push, SHJ)
  - But no recursion
- Fionda, Gutierrez and Pirro WWW 2012 (also: NautiLOD)
  - No notion Information Resource Graph
  - Function calls on results (no REST state manipulation)
  - But: recursion on graph traversal paths
- Hartig and Perez 2016: LDQL
  - Language to specify link traversal
  - Keep link traversal specification and query specification separate

Link Traversal Query Processing User Agent

- Input: Query (as SPARQL algebra expression), link expansion specification
- Output: Query results

- Construct RDF dataset for local processing from Query
- Evaluate SPARQL algebra expression over local RDF dataset
- Return query results

- Loop: do repeated query evaluation

Agenda – Part I (Andreas)

- Introduction
- The Four Linked Data Principles
- Vocabularies and Ontologies
- State vs. Event
- System Architecture
- Query Evaluation on RDF Datasets (Principle #3)
- Query Evaluation with Entailment (Principle #3)
- Link Traversal Query Evaluation (Principle #4)
- **Half-Time Summary**

- Part II: Traversing and Reasoning on Read-Write Linked Data (Tobias)

Learning Goals of the Tutorial – Part I

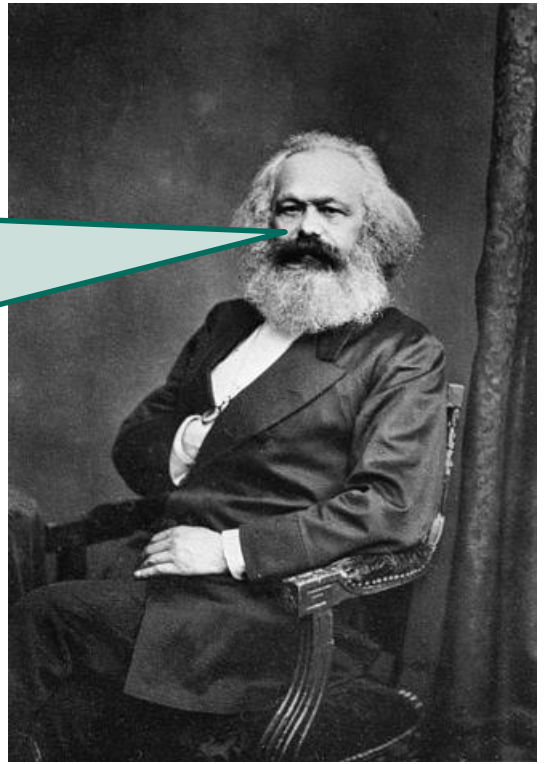
- Read RDF in Turtle syntax
- Recount the Linked Data principles
- List and briefly explain different vocabularies proposed for the Web of Things
- Explain the architecture of IoT systems based on web architecture
- Run SPARQL queries over live sensor data
- Perform reasoning over world state
- Run link-traversal specifications over live sensor data

Data Access and Integration on Devices and Sensors

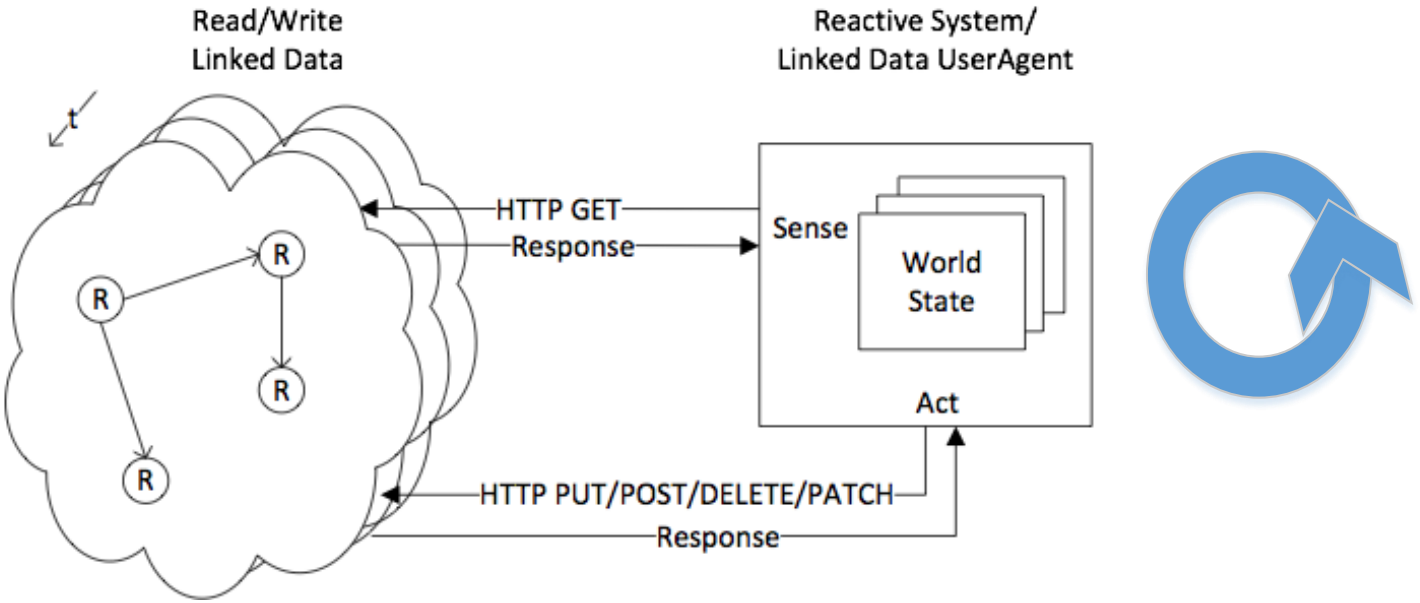
- Now you can easily do data integration on Linked Data, no matter whether data is „static“ or „dynamic“
- For some APIs, devices, sensors/actuators you need wrappers, but they are reasonably easy to build
- You need to keep in mind to use a resource-oriented interface on the web (and use a state-based view)

Karl Marx (1818 – 1883)

The ~~philosophers~~ ontologists have only interpreted the world, in various ways. The point, however, is to change it.

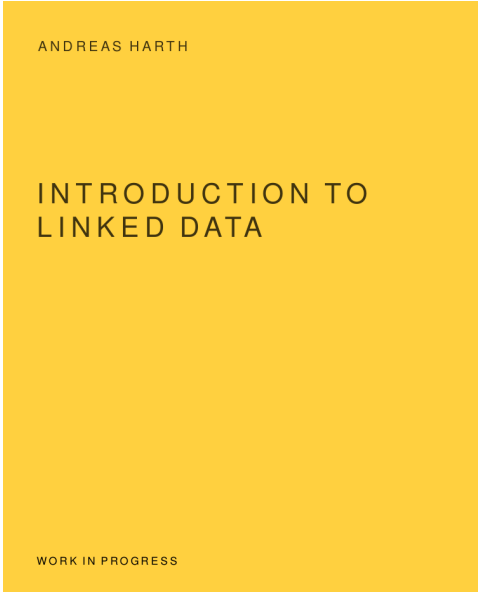


Summary of First Half, Outlook to Second Half



Commercial Break

- Some figures taken from the upcoming book „Introduction to Linked Data“

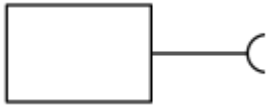


BACKUP

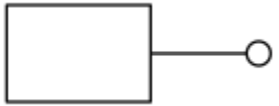
Uniform Network Interface

- Components A and B, data flows from A (source/producer) to B (sink, consumer)

- REST assumes request/response communication pattern between components with client connector and server connector
  - Clients emit requests, receive response



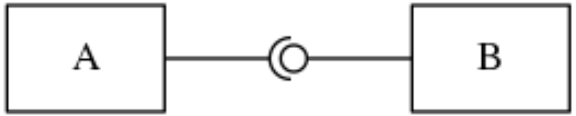
  - Servers answer to incoming requests with a response



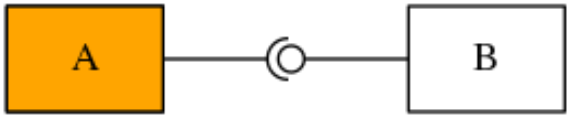
Network Interface: Push vs. Pull

Push

- A is client, B is server
- A emits PUT request
- At A: B.put(value)

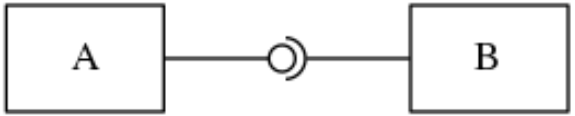


- Loop at A

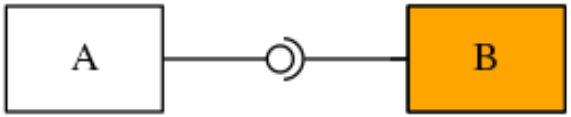


Pull

- A is server, B is client
- B emits GET request
- At B: value = A.get()



- Loop at B



Bi-directional Communication in HTTP

- Stream processor acts as server (which processes incoming HTTP requests)
- For publish/subscribe, each component is both client and server (a transducer or servient)