ETH zürich

inf | Informatik Computer Science

From “REST-as-we-use-it” to Design Patterns

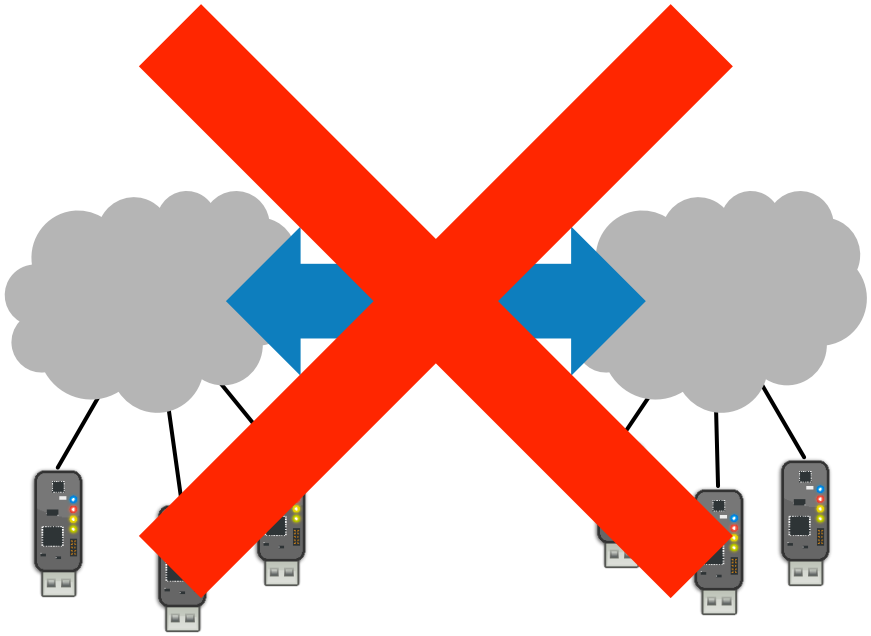
Matthias Kovatsch
kovatsch@inf.ethz.ch

IRTF Thing-to-Thing Research Group
Prague, Czech Republic

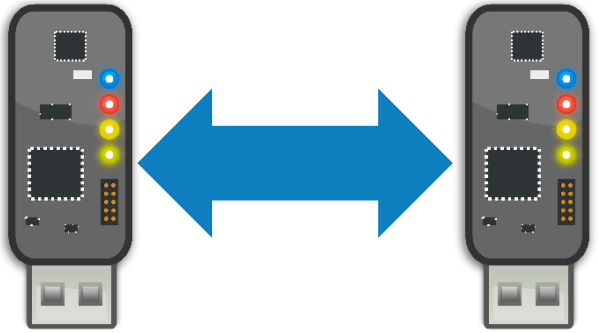
Saturday, 18 Jul 2015

REST for Thing-to-Thing Communication

Cloud-to-Cloud (with Things)

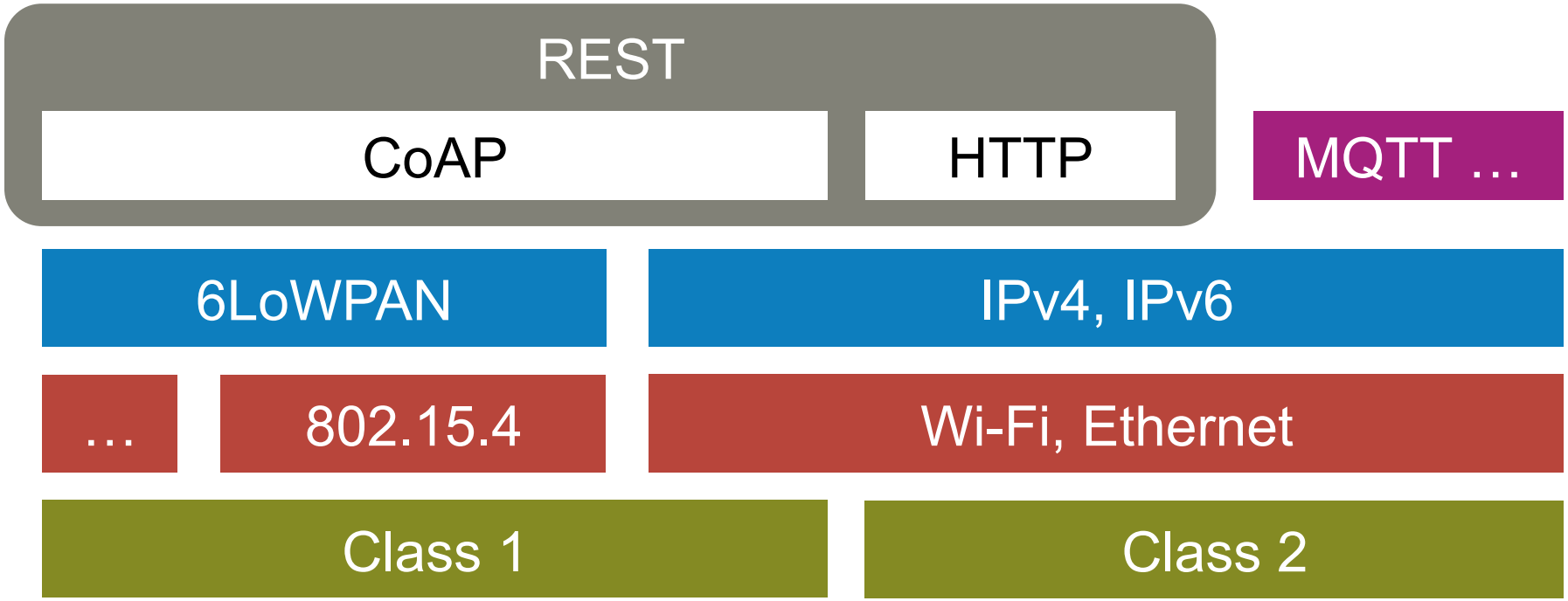


Thing-to-Thing



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

Typical Stack



Representational State Transfer (REST)

A set of Constraints

- Client-Server
- Stateless
- Cache
- Uniform Interface
- Layered System
- (Code-On-Demand)

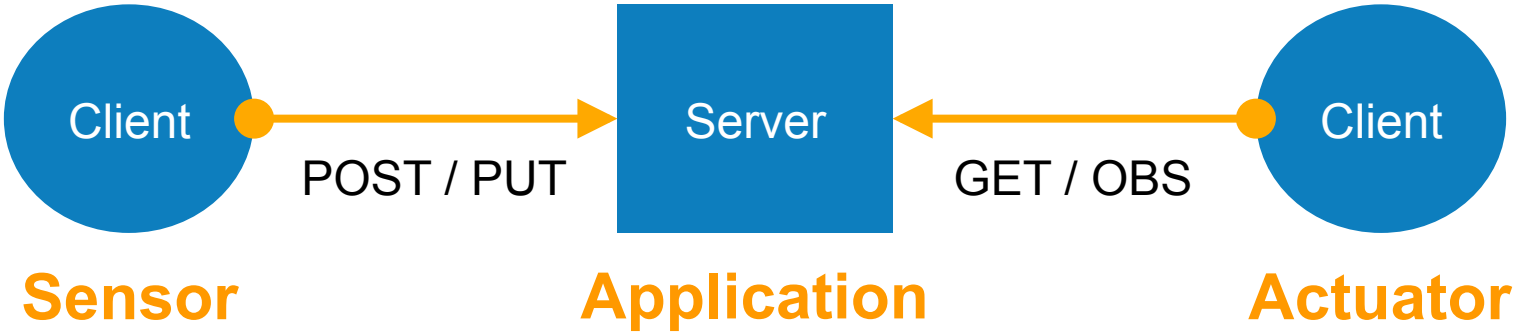
On a set of Elements

- User agents (client connectors)
- Origin servers (server connectors)
- Intermediaries (client + server conn.)
- Data (resources, representations, metadata)

Client-Server

Client model

- Traditional model for HTTP to push data from sensors



- Found in “CoAP Pub/Sub”
- Easy for normally-off devices

Client-Server

Server model

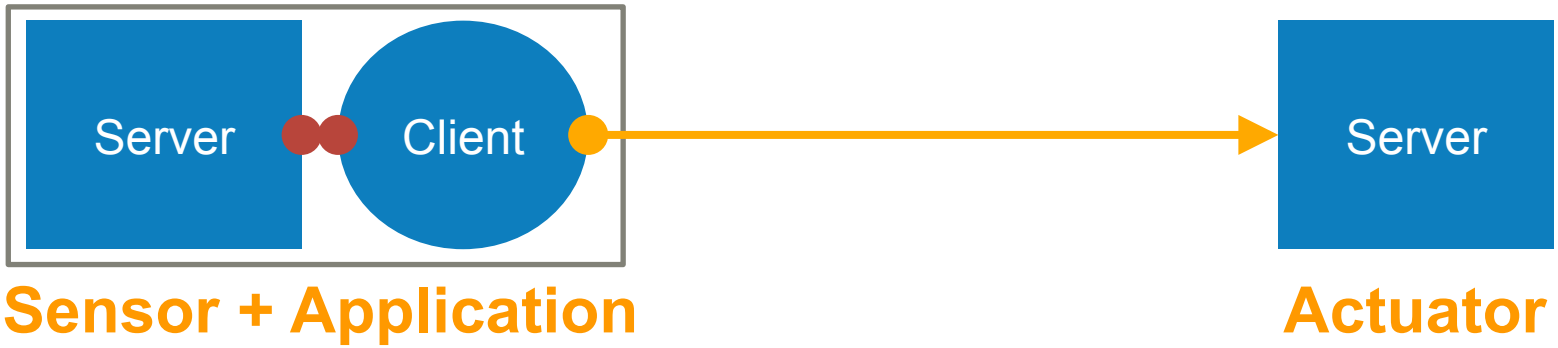
- CoAP Observe enables server push

- “Thin Server Architecture”
- The origin server has the data,
  the application (client) has the initiative

Client-Server

Combined roles

- Any combination possible

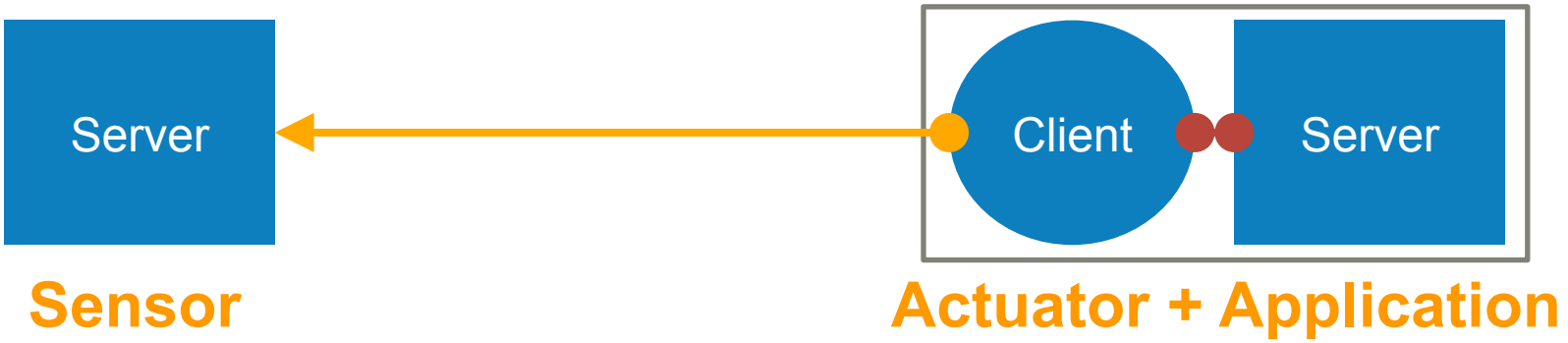


- For instance, server to configure client on sensor
- Enables direct Thing-to-Thing communication

Client-Server

Combined roles

- Any combination possible

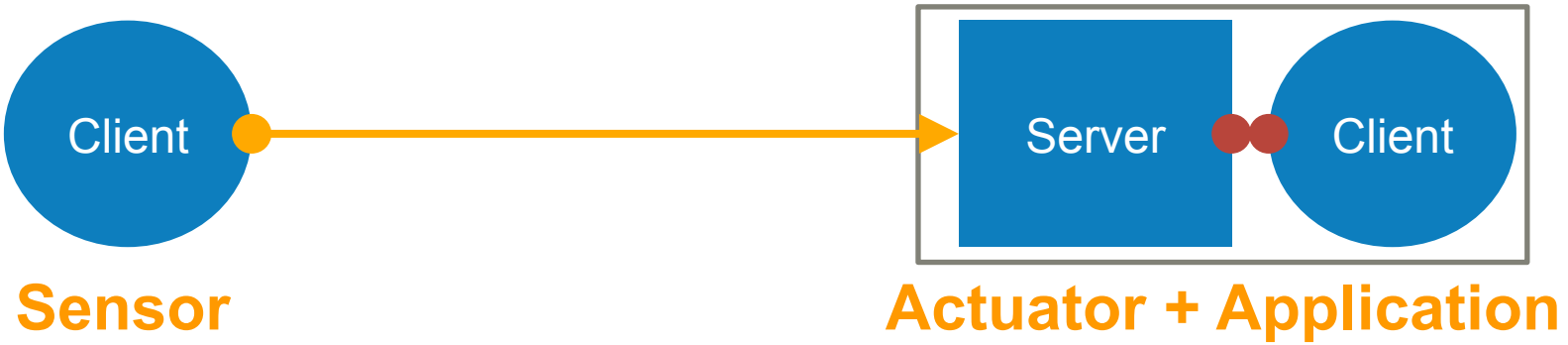


- Transferred data depends on where the application resides (client sending command vs client getting sensor value)

Client-Server

Combined roles

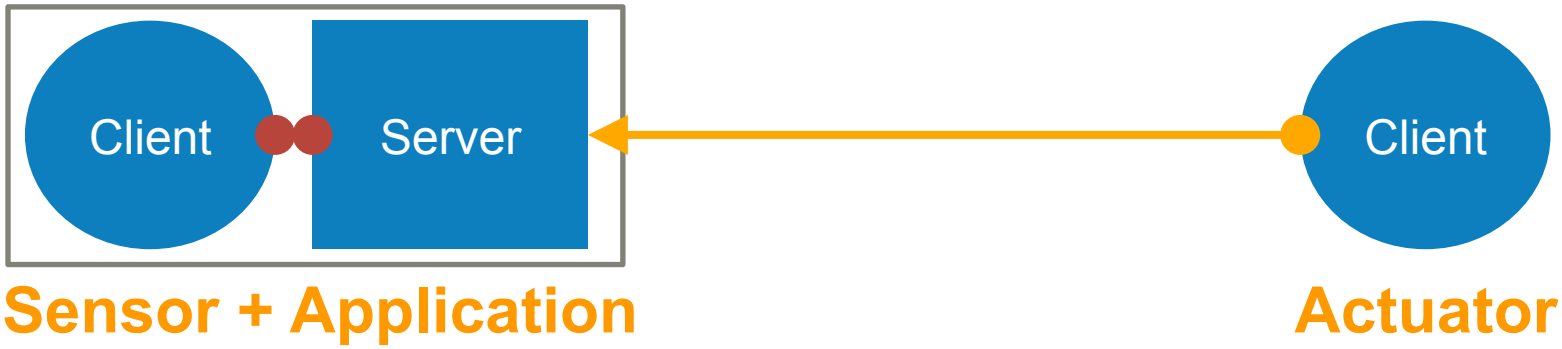
- Client model works correspondingly



Client-Server

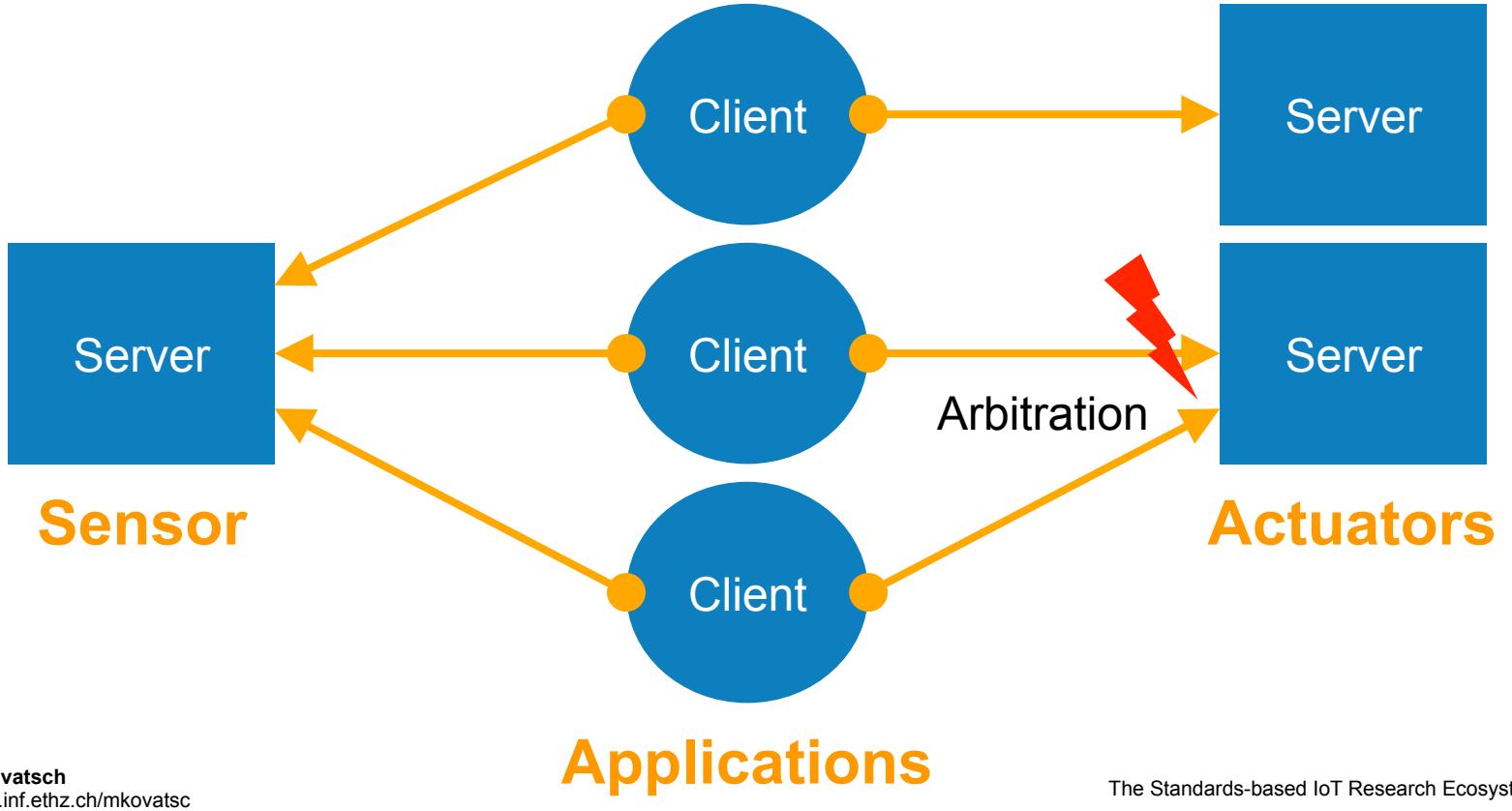
Combined roles

- Client model works correspondingly



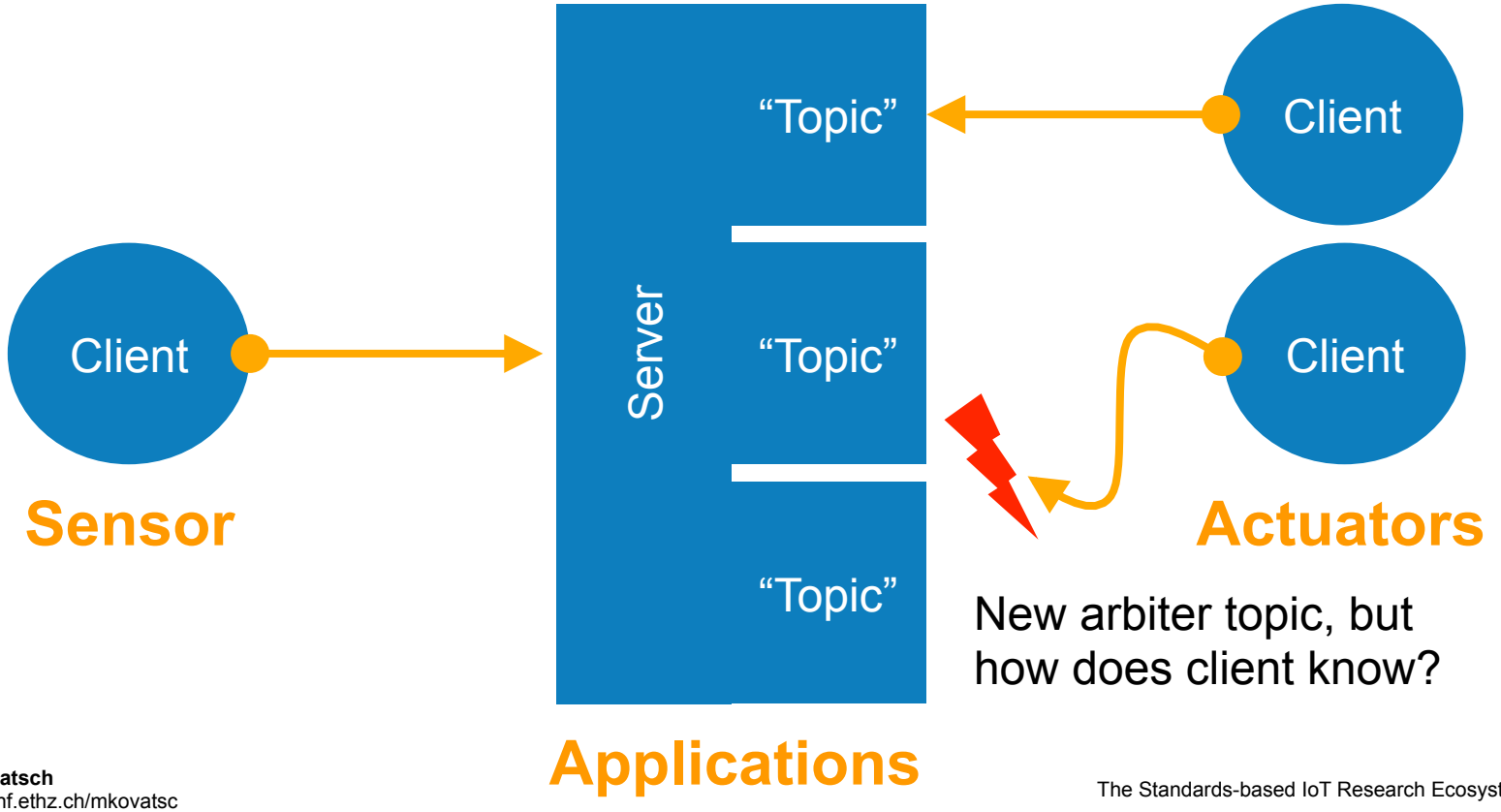
Client-Server

- Things should serve **multiple applications**

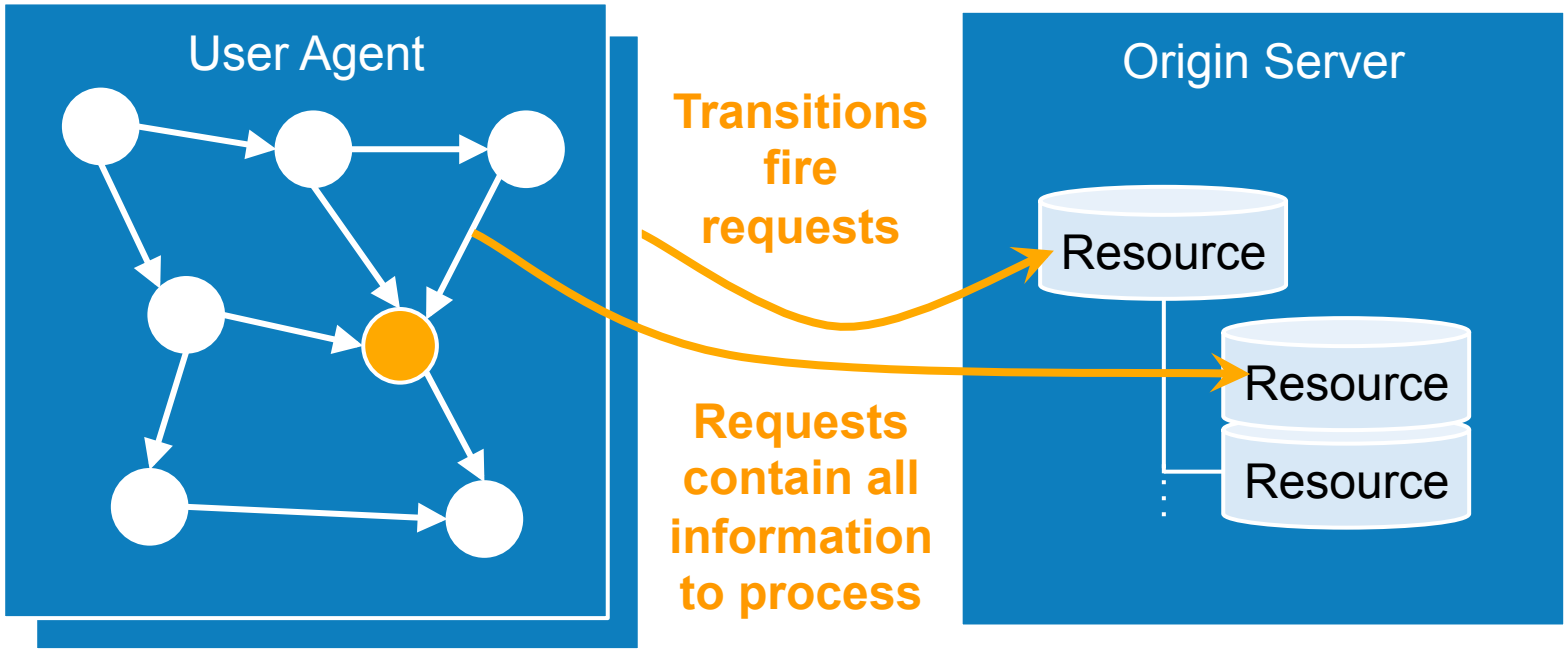


Client-Server

- Also possible with client model



Stateless

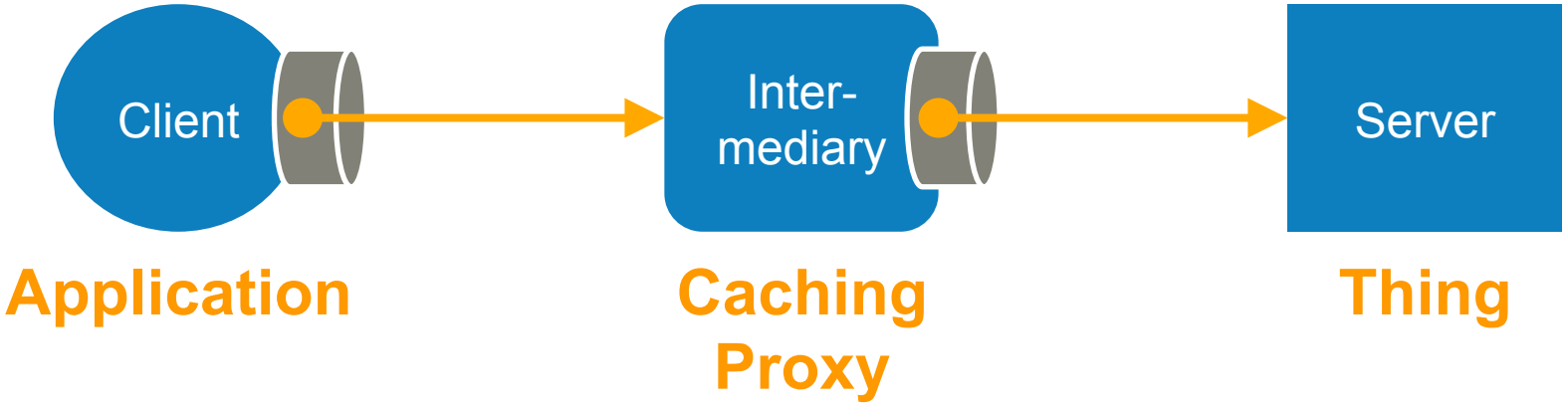


Only the clients keep their application state
(session/client state)

Servers store data that is independent from the individual client states
(resource state)

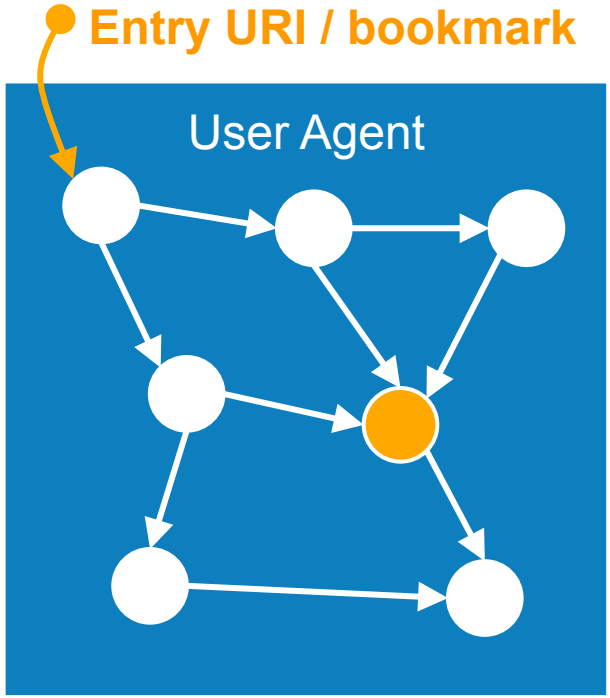
Cache

- Shield resource-constrained things



- Also helps with normally-off things (e.g., using Observe)

Uniform Interface



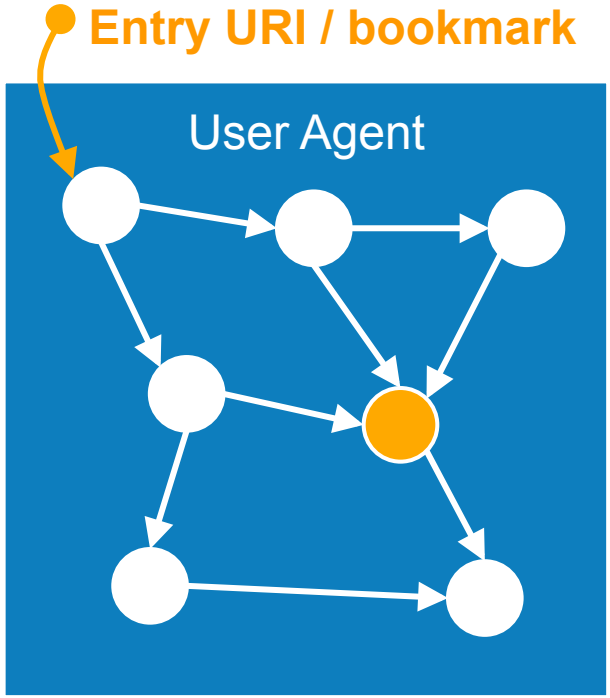
Application as finite-state machine (FSM) at the client

- Methods are standardized
- Hypermedia as the Engine of Application State
- We need proper **Internet Media Types** (e.g., Link-Format, SenML)
- Machine-readable forms?

Hypermedia as the Engine of Application State

1. Discovery through Entry URI
   `coap://<all-coap-nodes>/.well-known/core?rt=rd*`
2. Get list of links to things in `application/link-format`
   `coap://rd.example.com/rd-lookup/res`
3. Select the designated thing from the list of links
   `<coap://thing1.example.com/temp>;rt=temperature`
4. Repeat following links

Uniform Interface

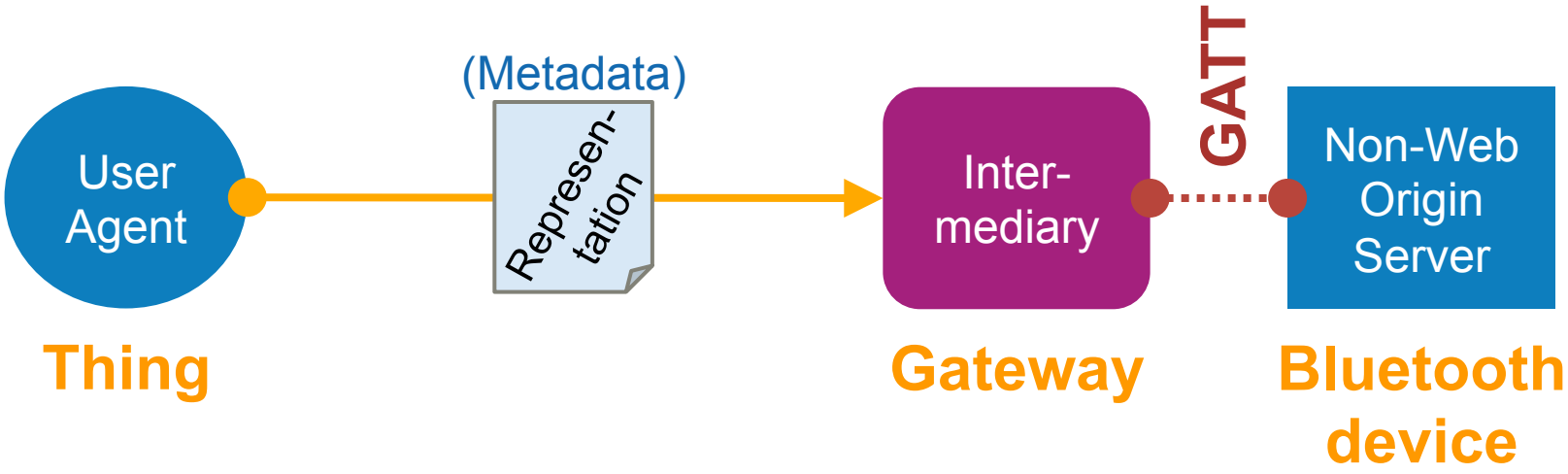


Application as finite-state machine (FSM) at the client

- Methods are standardized
- Hypermedia as the Engine of Application State
- We need proper **Internet Media Types** (e.g., Link-Format, SenML)
- Machine-readable forms?

Layered System

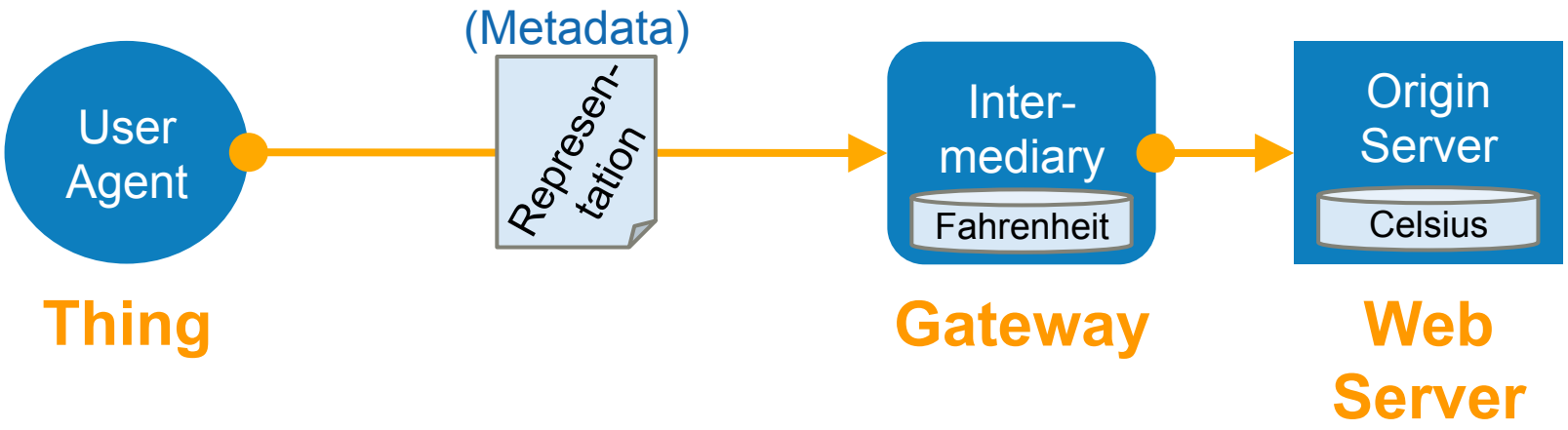
- Helps to integrate “legacy” systems



- Standard mappings to other protocols help integration

Layered System

- Can provide adaptors such as converters or aggregators



- Can also add features such as Conditional Observe

Representational State Transfer (REST)

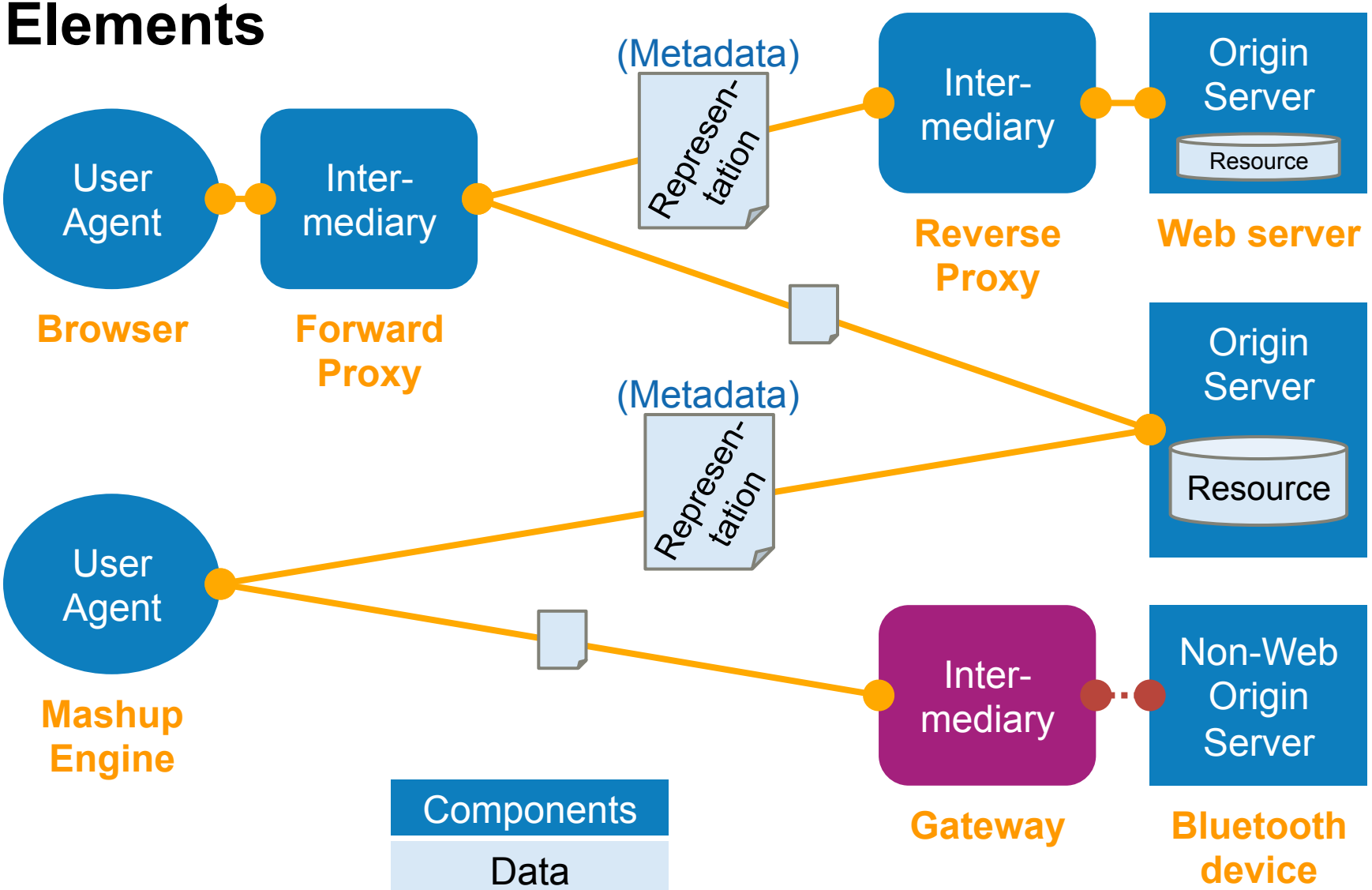
A set of Constraints

- Client-Server
- Stateless
- Cache
- **Uniform Interface**
- Layered System
- (Code-On-Demand)

On a set of Elements

- User agents (client connectors)
- Origin servers (server connectors)
- Intermediaries (client + server conn.)
- Data (resources, representations, metadata)

Elements



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

ETH zürich

inf Informatik Computer Science

Questions?

Matthias Kovatsch

kovatsch@inf.ethz.ch

https://github.com/mkovatsc/

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