SIEMENS

Andreas Scholz (andreas.as.scholz@siemens.com)

Beyond REST

Creating Automation Systems out of Things with REST

Beyond REST

Creating Automation Systems out of Things with REST

- Motivation / Approach
  - Take the view of an automation system designer / engineer
  - Take the building blocks that are available (e.g. CoAP) and try to build a REST based solution
  - Identify what “feels strange” and what is “missing”

- The presentation uses home automation examples
  - The hope is that this provides an easy to grasp common ground for discussions
  - The examples have been selected to **illustrate** typical interaction scenarios from industrial / energy automation systems – they might not necessarily be the optimal solution for home automation

SIEMENS

Beyond REST
Creating Automation Systems out of Things with REST

Take with a big grain of salt!

Properties of / assumptions about automation systems

- **Distribution**: It is worthwhile to separate between
  - "Local" automation tasks (industry automation, wind parks, home automation, etc) that operate on a LAN-like system
  - "Global" automation tasks (smart grid, logistics, etc) that operate on an Internet-like system

Focus of this presentation

- **Ownership**: It is worthwhile to separate between
  - Systems with a strong "owner" (factories, plants, power plants) that has a lot of control over the behavior of its components
  - "Unreliable" systems w.r.t. availability and usage (pv panels in the smart grid)

Focus of this presentation

- **Rate of Change**: It is worthwhile to separate between
  - Control, which evolves slowly. Changes occur, but often the system will run "as-is" for months or years. Changes occur in a controlled manner in re-engineering phases.
  - Data acquisition (optimization, predictive maintenance, etc) which is more dynamic

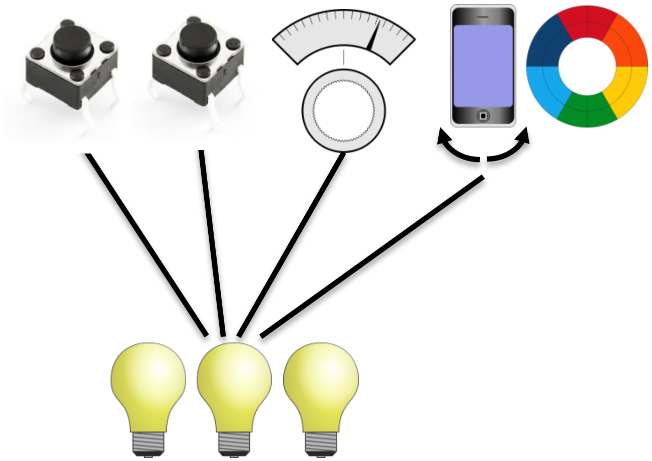
Focus of this presentation

- **Data Flow**: The vast majority of data flow stems from planned (often periodic) interactions. Hundreds of control loops can run "in parallel", coordinated by a hierarchy of higher level systems. Ad-hoc interactions are rare.
- **QoS**: The primary concern regarding QoS is achieving deterministic behavior (latency, jitter, etc) for well-known workloads. Scalability / elasticity is second.

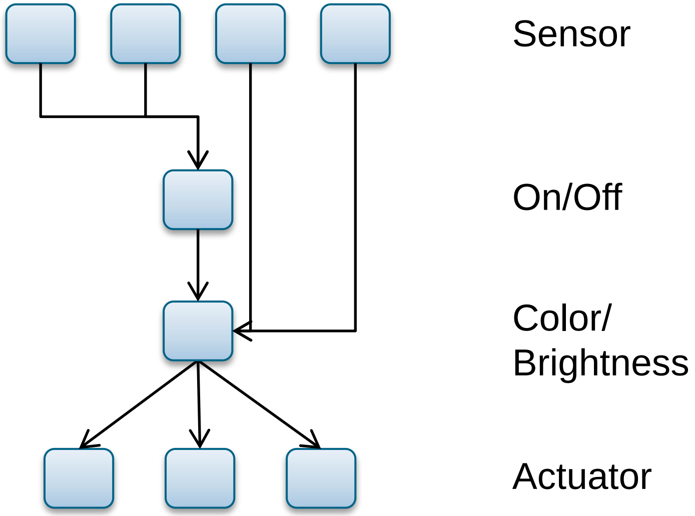
Beyond REST
Creating Automation Systems out of Things with REST

The Application: Single room lighting control, a Things–to–Things interaction

Thing View

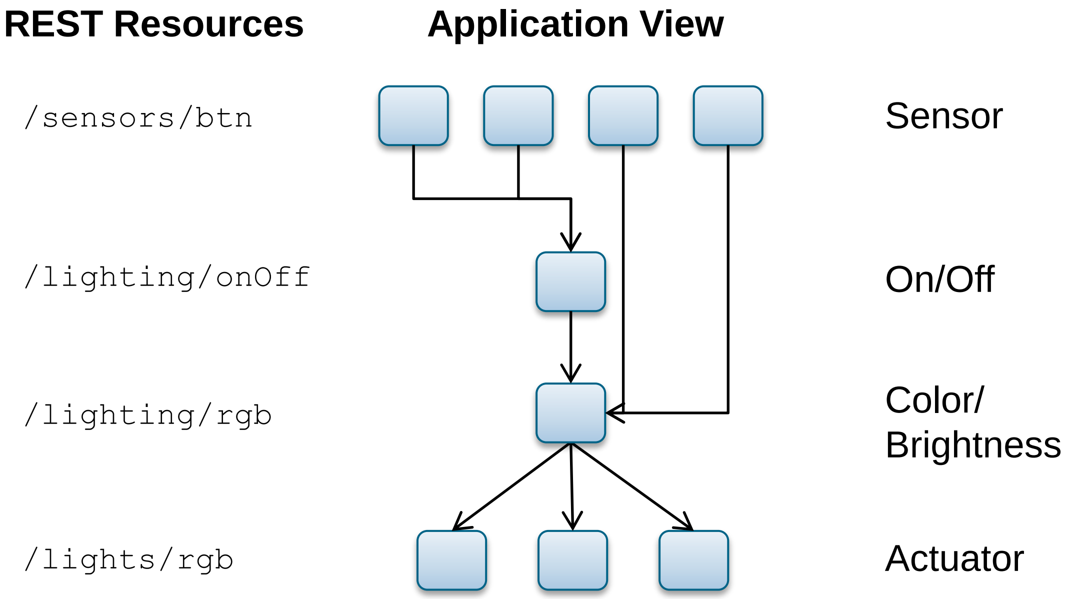


Application View



Beyond REST
Creating Automation Systems out of Things with REST

Mapping to REST - Resources



REST Resources	Application View
`/sensors/btn`	Sensor
`/lighting/onOff`	On/Off
`/lighting/rgb`	Color/ Brightness
`/lights/rgb`	Actuator

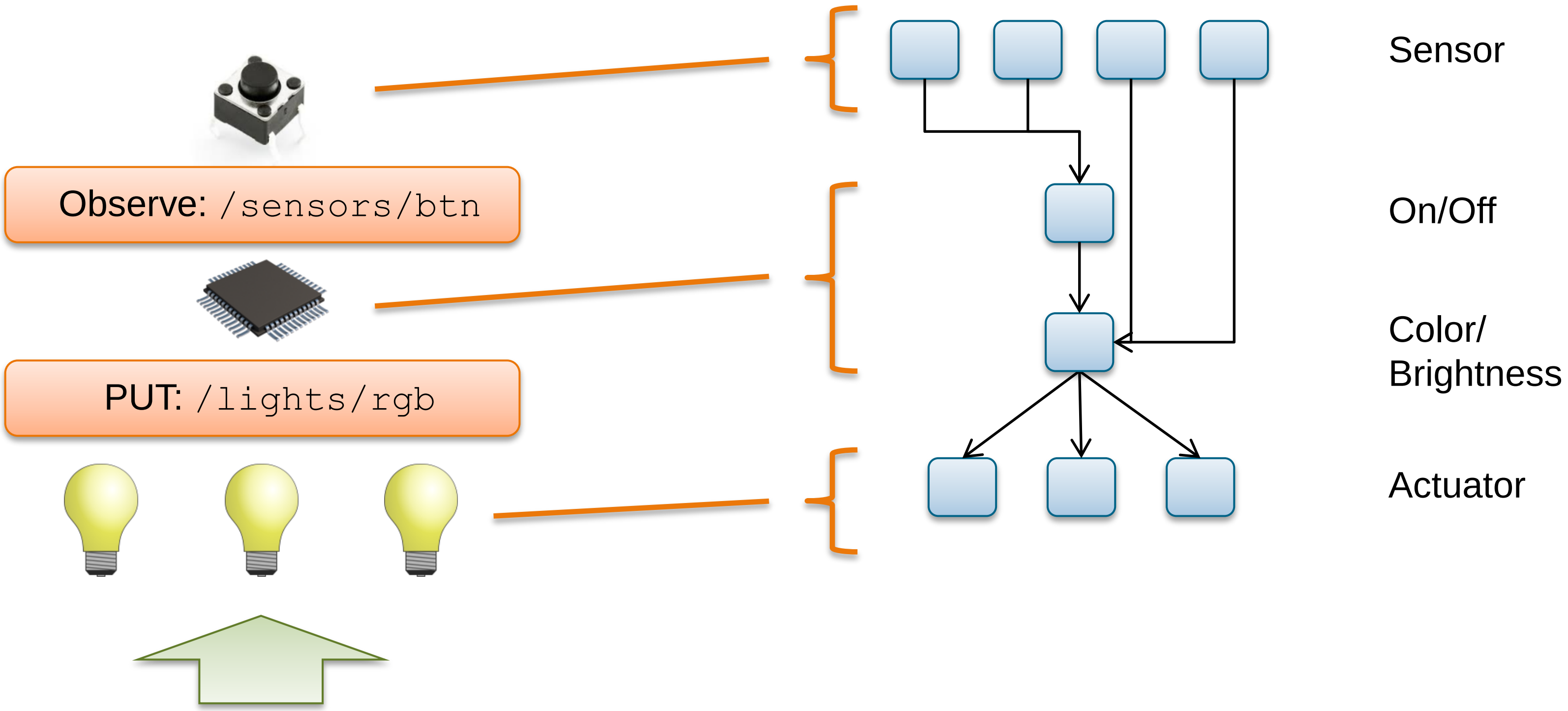
That feels “natural”

Beyond REST
Creating Automation Systems out of Things with REST

Mapping to REST - Interactions

REST Interactions Try One
“Control Node”

Application View



That feels “strange”
because its inconsistent

From an automation point of view the data flow between sensor and controller is “the same” as the flow between controller and actuator – why do we have different paradigms?

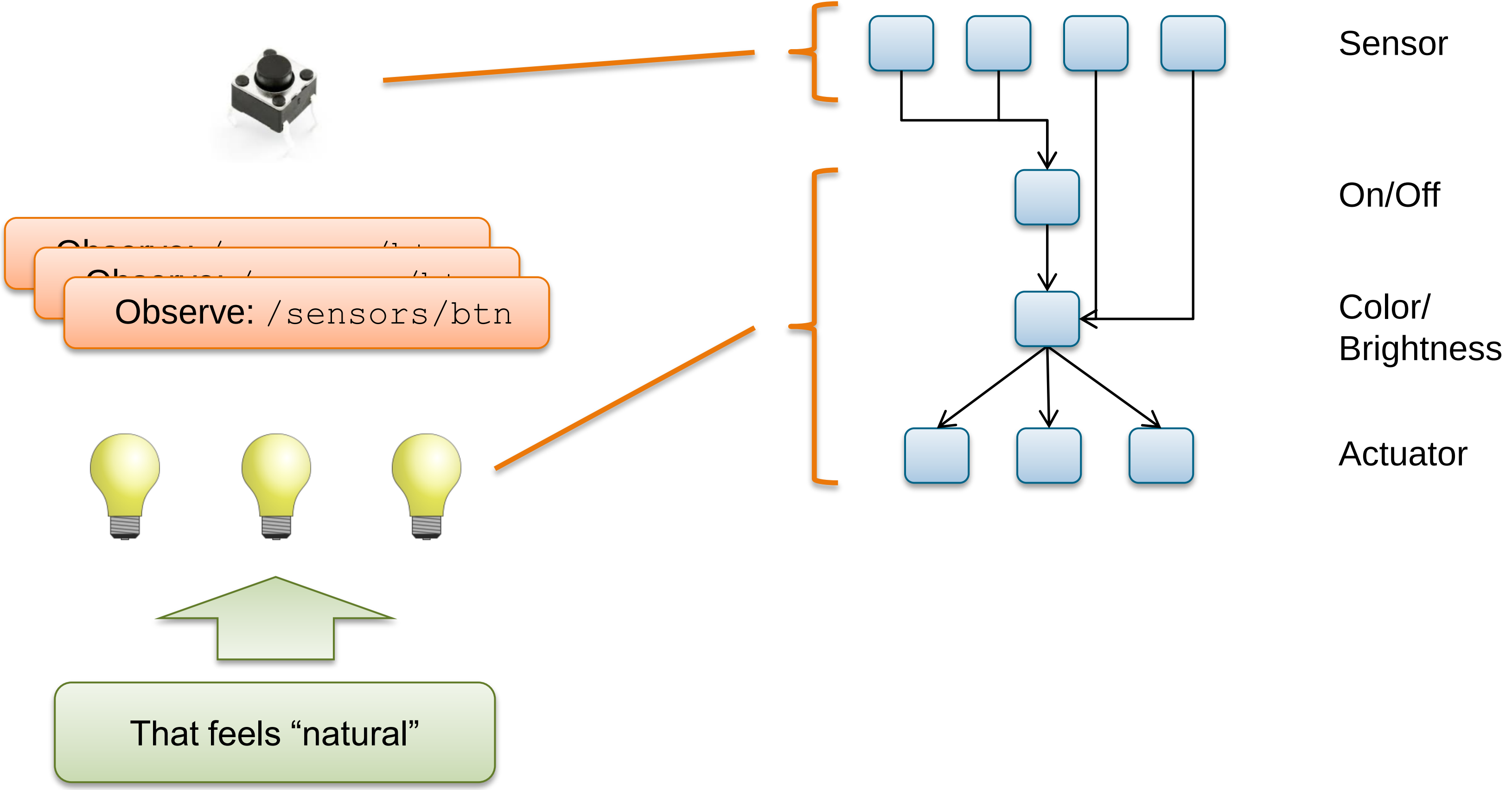
Beyond REST
Creating Automation Systems out of Things with REST

Mapping to REST - Interactions

REST Interactions Try Two

"Dumb Switches, Smart Bulbs"

Application View



Observe: /sensors/btn

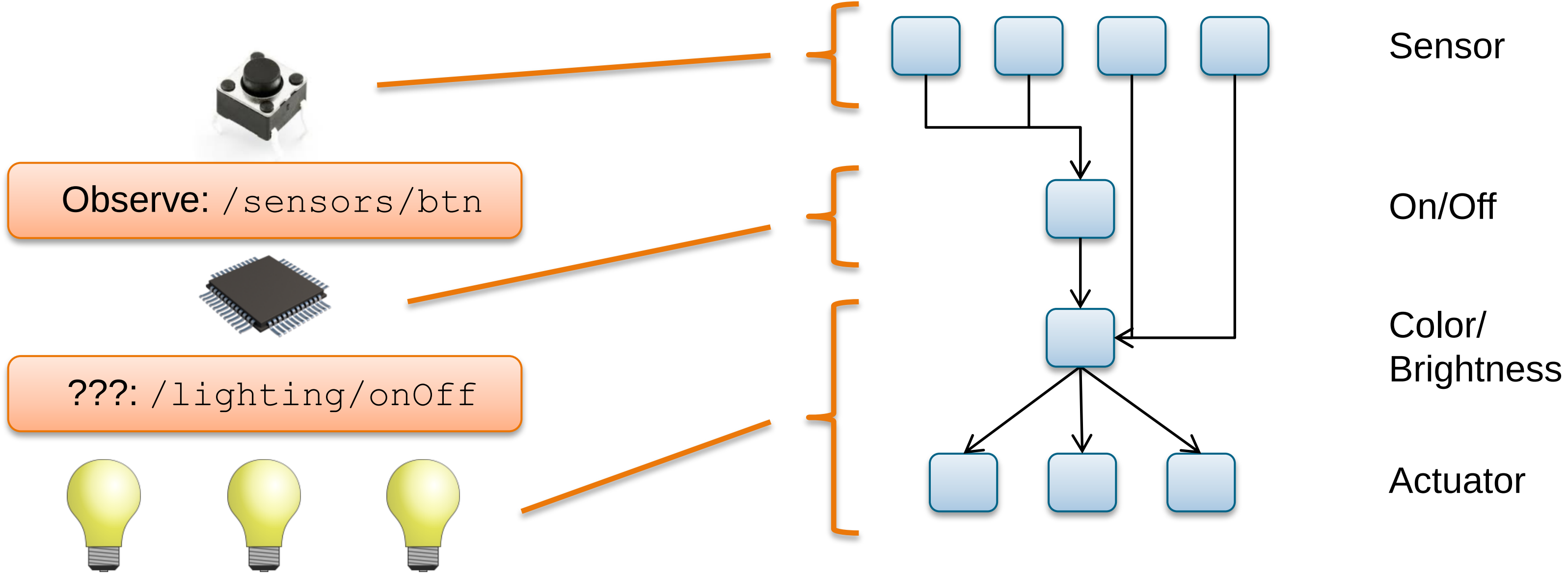
That feels "natural"

Beyond REST
Creating Automation Systems out of Things with REST

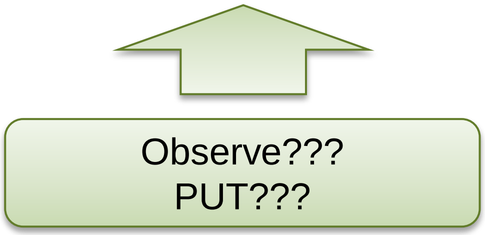
Mapping to REST - Interactions

REST Interactions Try Three
“Control Node, Smart Bulbs”

Application View


Observe: `/sensors/btn`

???: `/lighting/onOff`

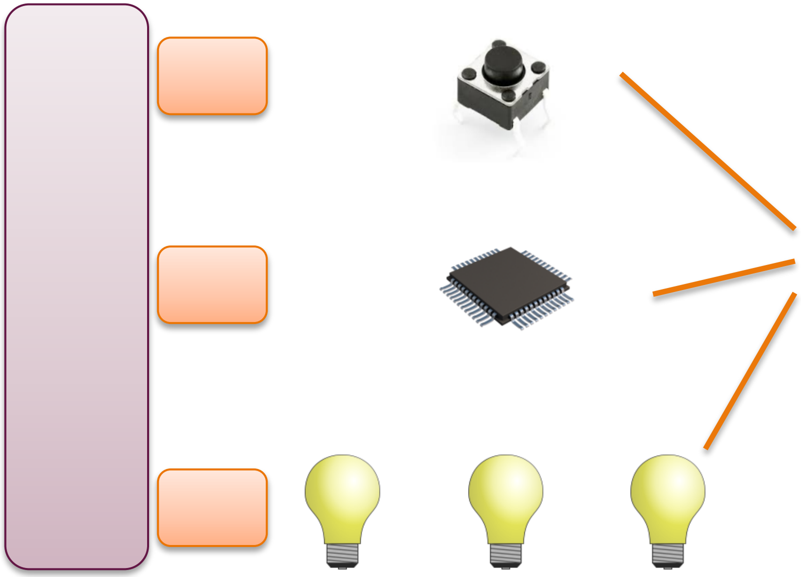

Observe???
PUT???

Hmm where should I put the `/lighting/onOff` endpoint? Is that an “ouput” of the controller? Or is that an “input” of the bulb? Or do we have both resources?

Beyond REST
Creating Automation Systems out of Things with REST

Mapping to REST - Interactions

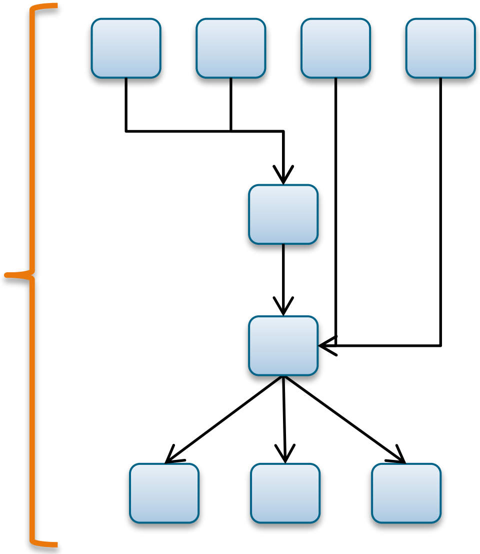
REST Interactions Try Four
“Pub / Sub”



CoAP MQ

Freedom of choice

Application View



Sensor

On/Off

Color/ Brightness

Actuator

That feels “natural”
… and like a single point of failure

Abstraction level introduced by CoAP MQ hides the client/server differences

Beyond REST

Creating Automation Systems out of Things with REST

Looking at CoAP MQ I am wondering whether we can (and should) decouple the role of “orchestration” from the role of “data forwarding”

From an automation point of view it would be ideal to have a way to model/ describe the communication paths from a system point of view and “download” this information to the devices. This would include the creation of brokers, proxies and other “data forwarders” during run

- Several constrained devices? Introduce a broker
- Non-constrained devices? Use peer-to-peer observes
- Constrained device and many observers? Introduce a proxy in between

Do we need a more “system centric” view?

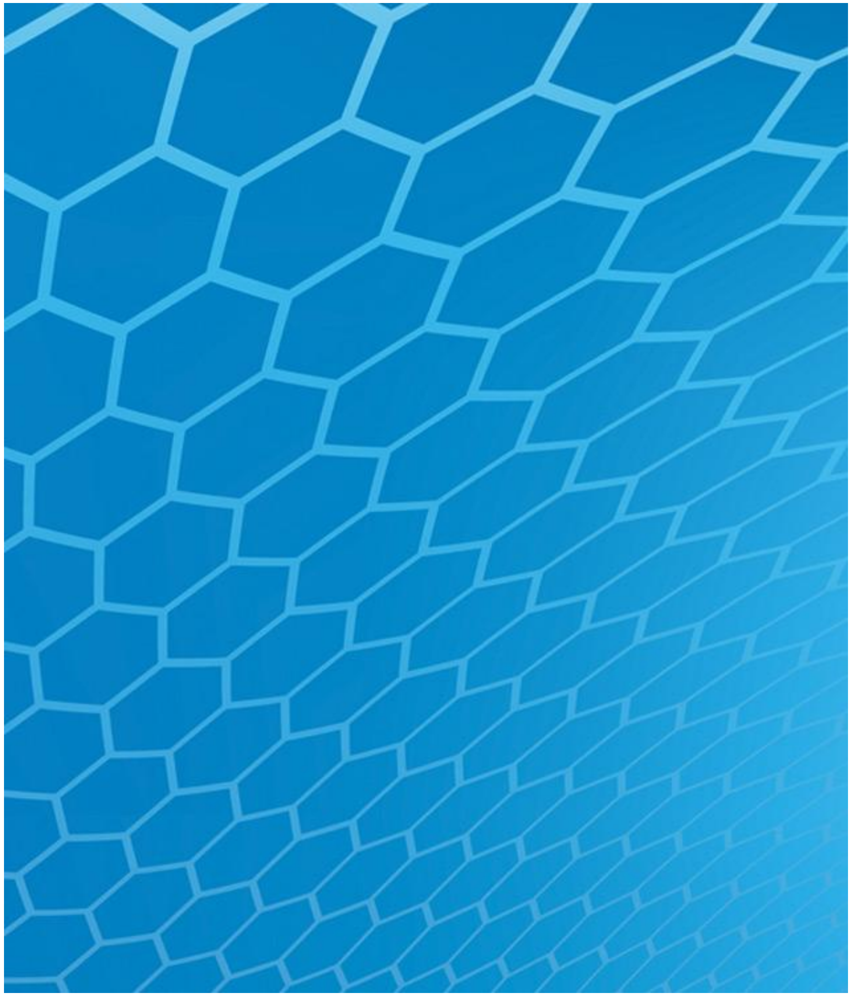
Remember the assumptions about
“Distribution”
“Ownership”
“Rate of Change”

Benefits:

- We could leverage existing CoAP interaction paradigms and avoid a layer of “middleware” on top of CoAP that degrade CoAP to a data pipe / rpc mechanism and avoid parallel communication “silos”
- We could express interactions that involve multiple resources on different devices (I need the switch and the brightness sensor) and dynamically create “aggregators”/ reverse proxies that collect data from multiple devices and re-publish them under a new resource
- The description format could be extended to support the specification of QoS parameters for the interactions. This might be a promising way to exploit the capabilities provided by software defined networks

Beyond REST

Creating Automation Systems out of Things with REST



Andreas Scholz
Principal Research Scientist
Siemens CT RTC NEC

E-mail:
andreas.as.scholz@siemens.com