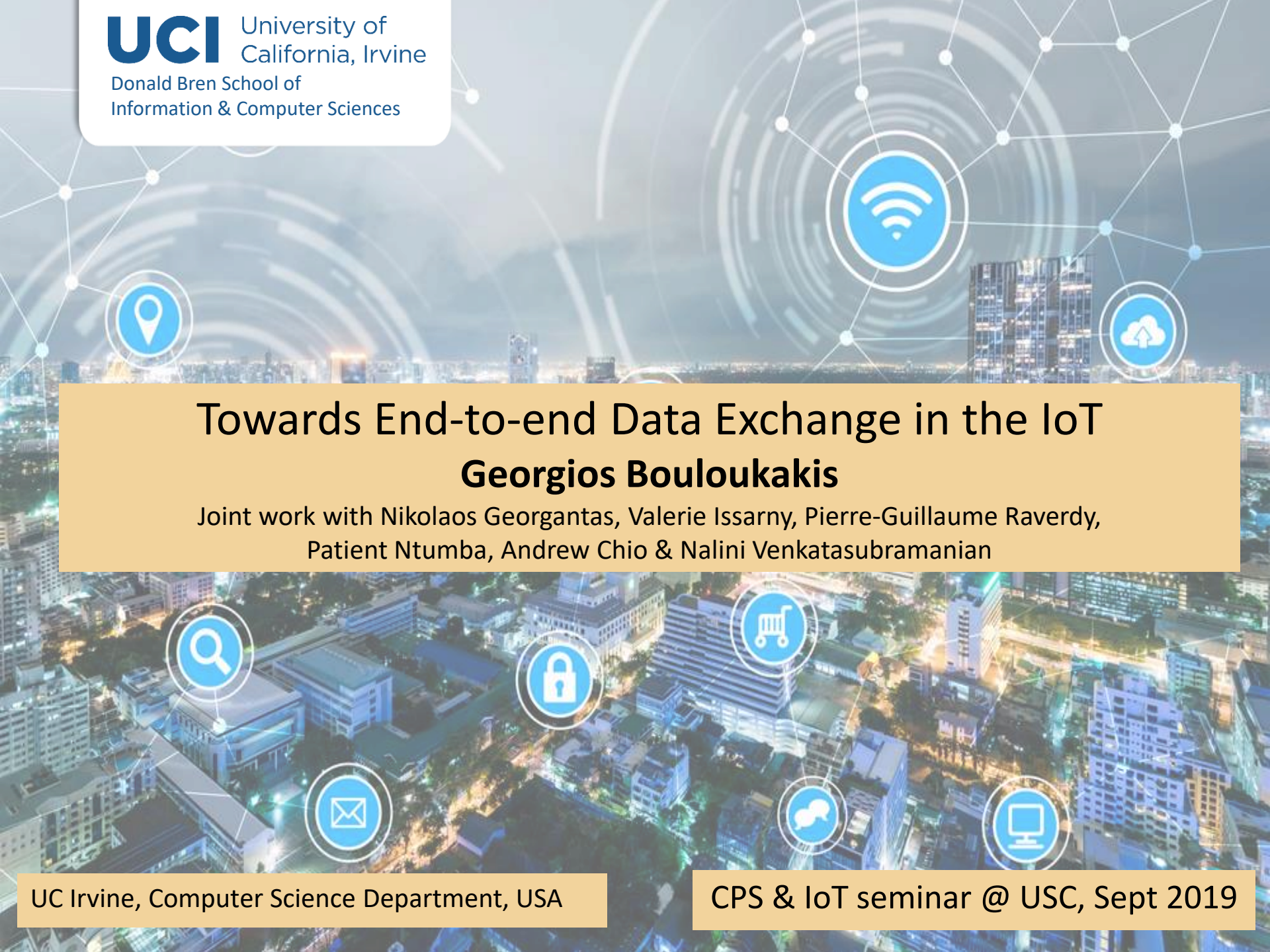


University of California, Irvine
Donald Bren School of Information & Computer Sciences

Towards End-to-end Data Exchange in the IoT

Georgios Bouloukakis

Joint work with Nikolaos Georgantas, Valerie Issarny, Pierre-Guillaume Raverdy, Patient Ntumba, Andrew Chio & Nalini Venkatasubramanian

UC Irvine, Computer Science Department, USA

CPS & IoT seminar @ USC, Sept 2019

Enabling Data Exchange in IoT Smart Spaces

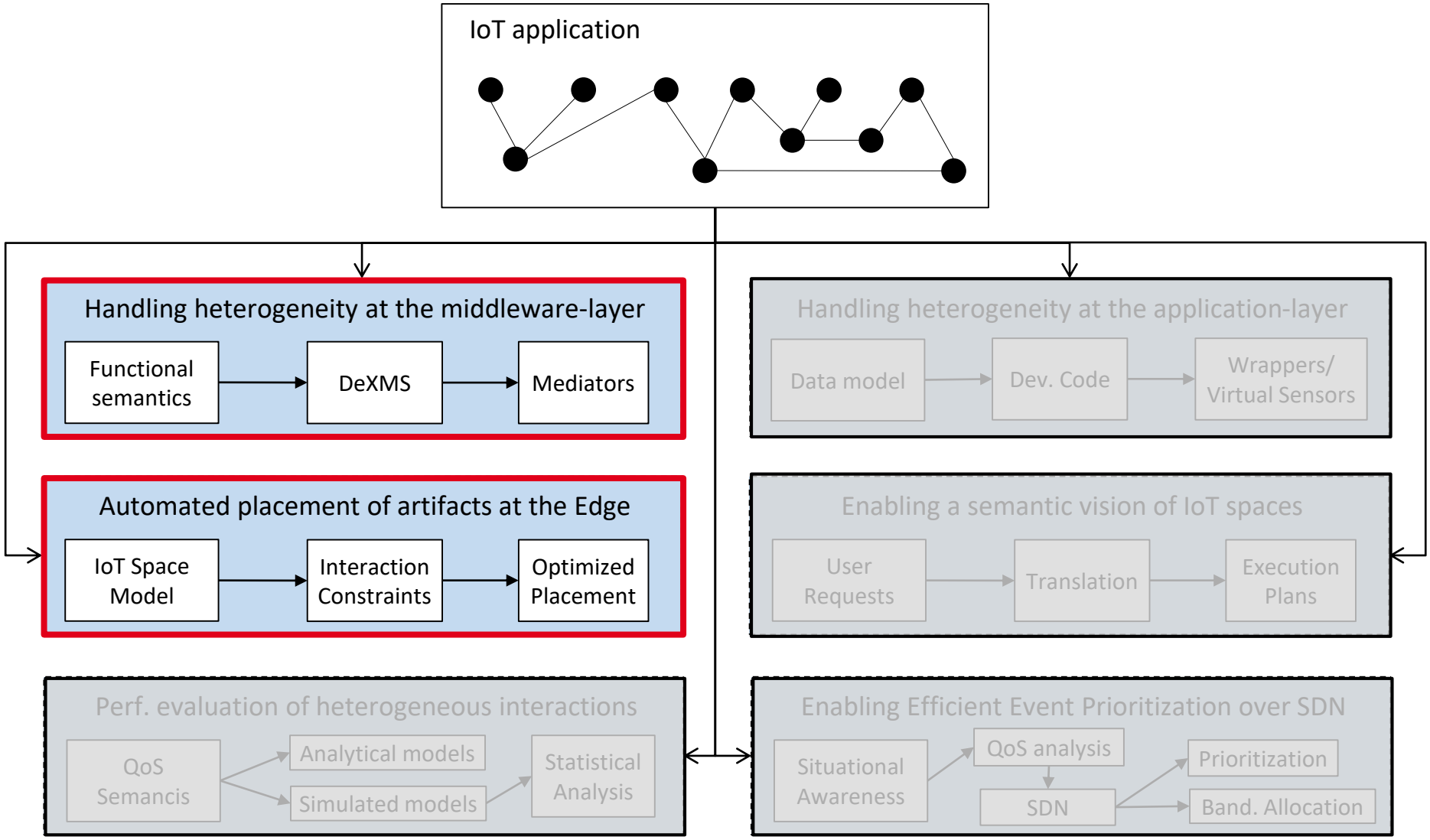
Dynamic

Heterogeneous

"what is the **occupancy** of the room 2065?

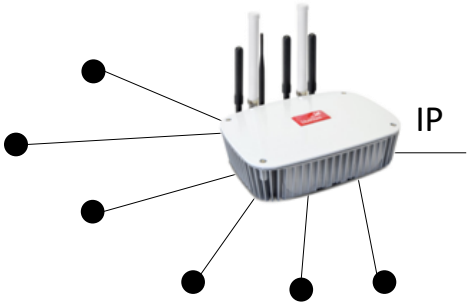
"**decrease** the **temperature** of those rooms with **occupancy** above 50% of their capacity?"

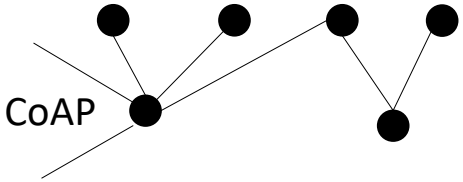
Most recent research



IoT heterogeneity at multiple layers

Application layer	⟺	Application layer
Middleware layer	⟺	Middleware layer
Transport Layer		Transport Layer
Network layer		Network layer
Data Link layer (MAC & LLC)	⟺	Data Link layer (MAC & LLC)
Physical layer	⟺	Physical layer



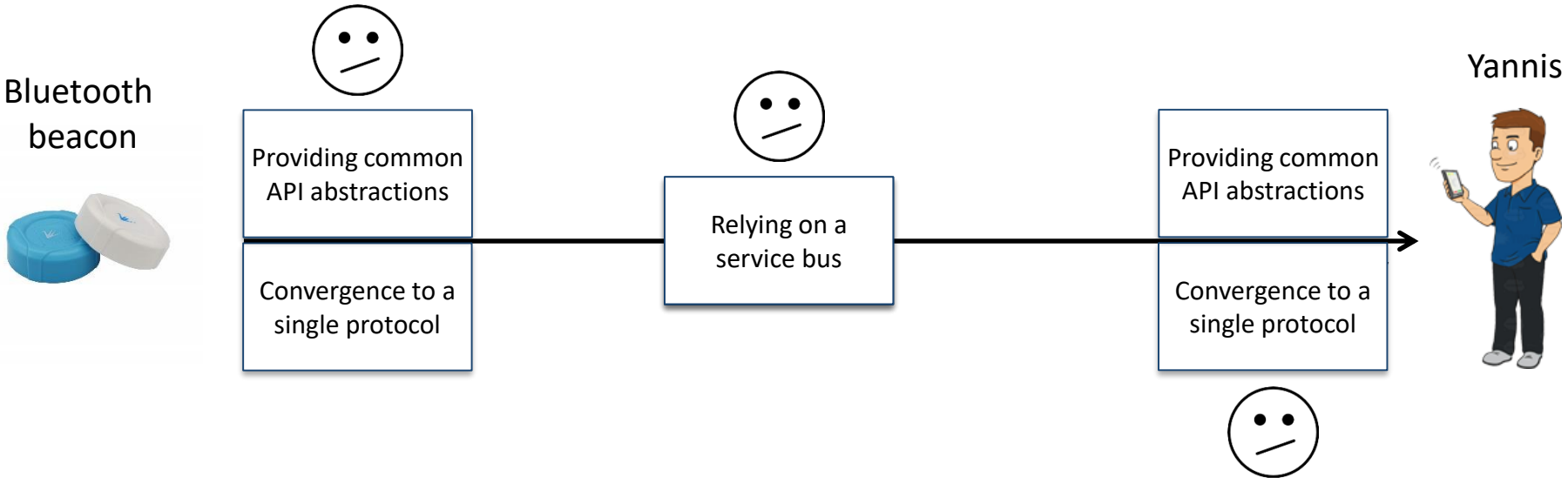


Middleware protocols in the mobile IoT

DPWS CoAP MQTT ZeroMQ WebSockets ….

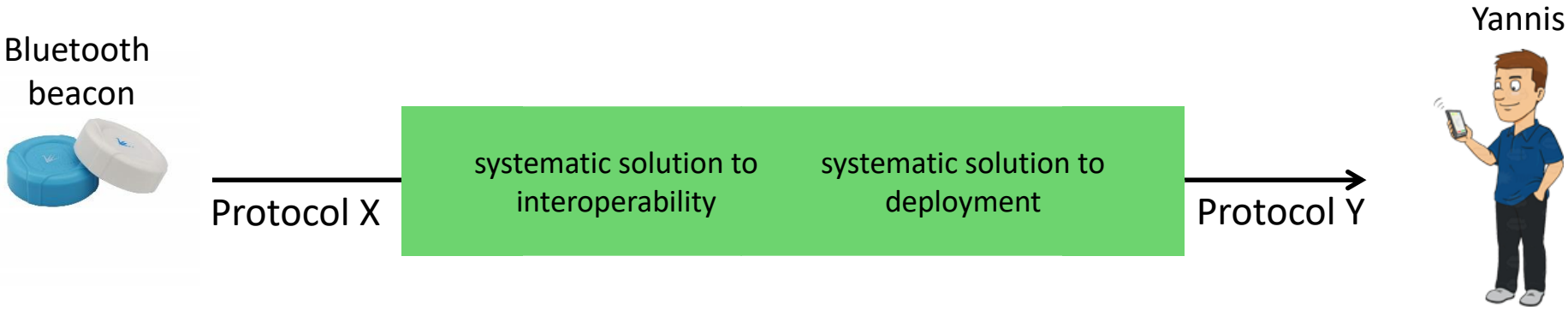
Client-server Pub/sub Streaming ….

Heterogeneous interconnections in the mobile IoT



- How to enable interconnections in the mobile IoT ?

Our proposed solution



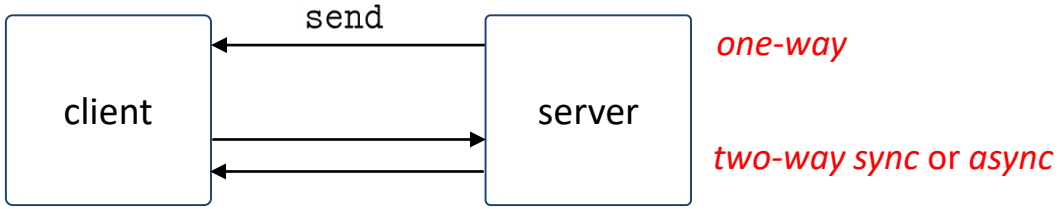
Automated synthesis of interoperability artifacts:

- enables functional middleware-layer interoperability

Automated placement and deployment at the Edge:

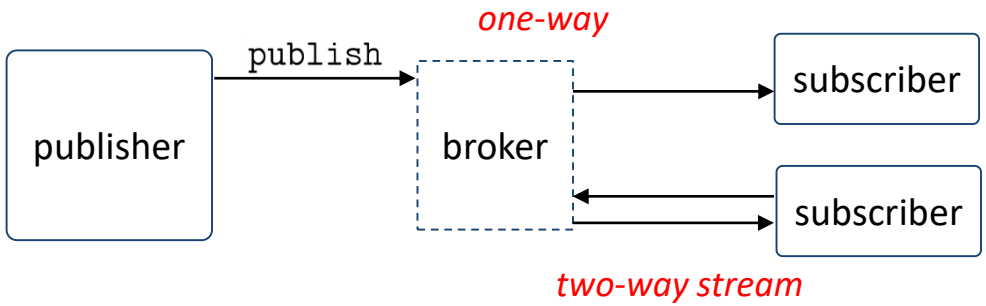
- enables the deployment of interoperability artifacts at the Edge

Models for core interaction paradigms



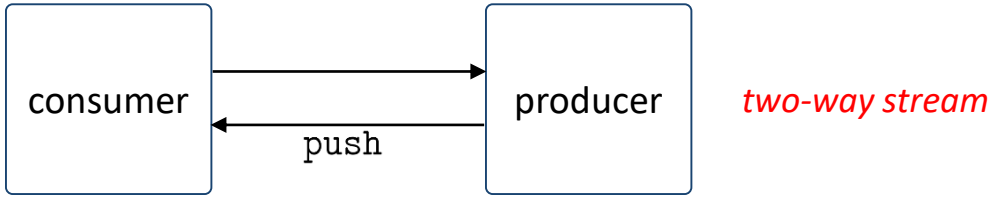
Client–Service (CS)

- Tight Time & Space Coupling



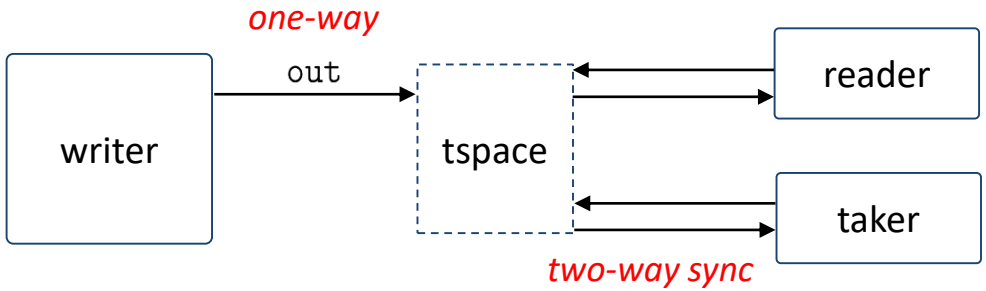
Publish-Subscribe (PS)

- Time & Space Decoupling



Data Streaming (DS)

- Tight Time & Space Coupling



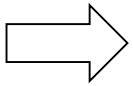
Tuple Space (TS)

- Time & Space Decoupling

Data eXchange (DeX) connector model

- Our generic connector defines 4 basic interaction types:

one-way
two-way async
two-way sync
two-way stream



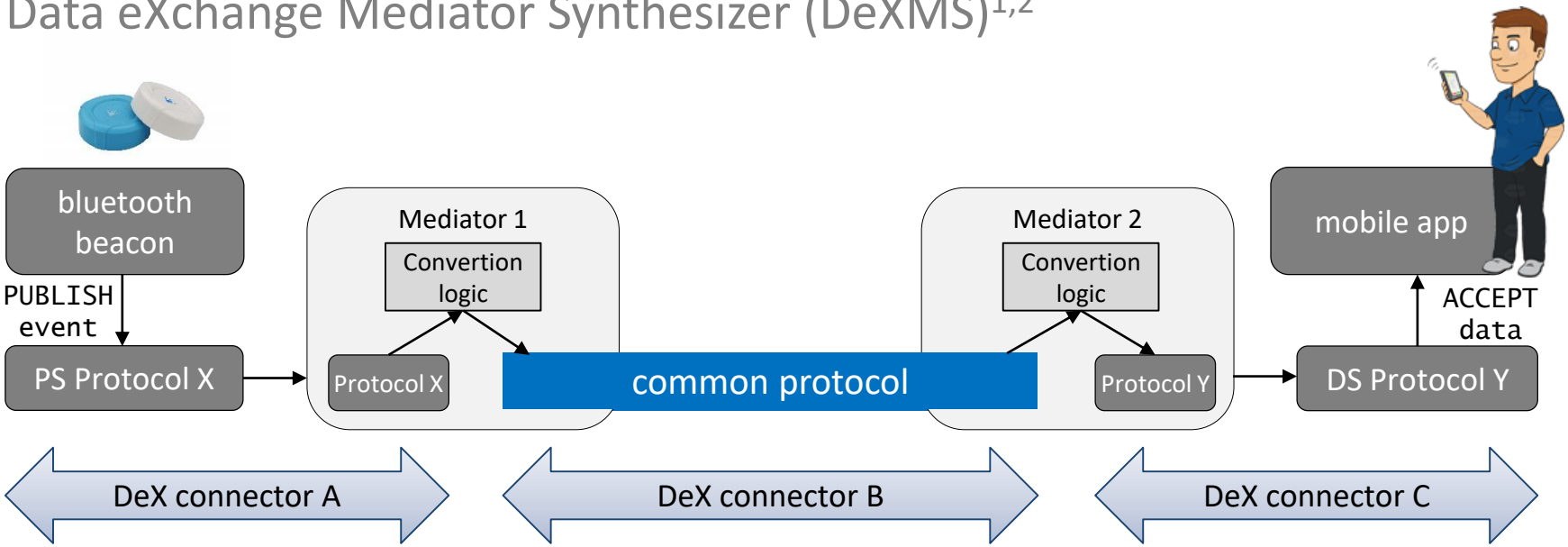
each interaction is represented as combination of **post** and **get** primitives

post and **get** primitives abstract CS, PS, DS and TS primitives

We rely on the DeX abstraction to introduce our middleware protocol interoperability solution

Our middleware protocol interoperability solution

- Data eXchange Mediator Synthesizer (DeXMS)[1,2]



- Mediator architecture: relies on DeX for automated Mediator synthesis
- Primitives & data conversion between the common protocol and the Things' protocols
- A universal way to describe the Things' I/O required

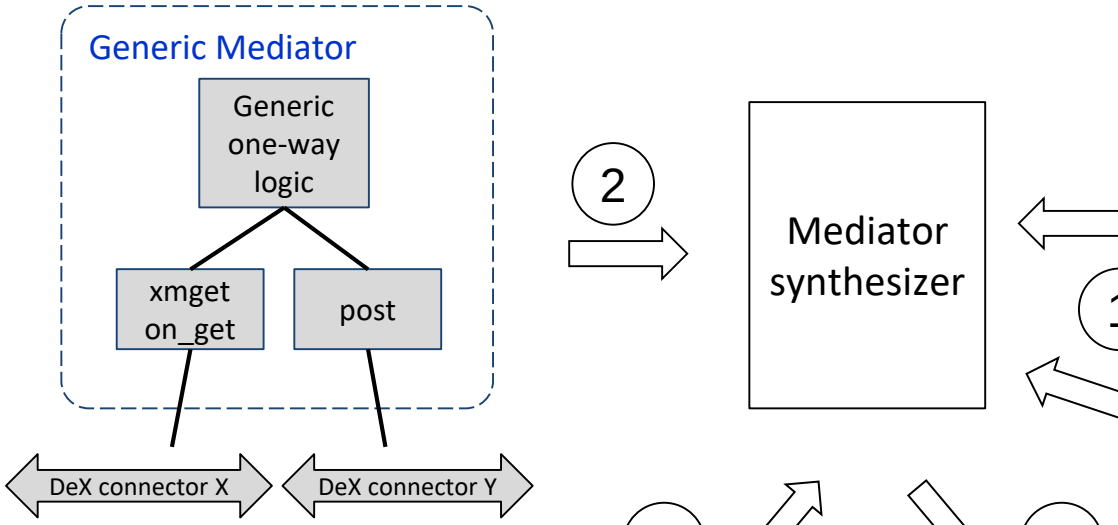
[1] G. Bouloukakis et al., FGCS, 2019
[2] G. Bouloukakis et al., ICSOC, 2016

Automated Mediator synthesis

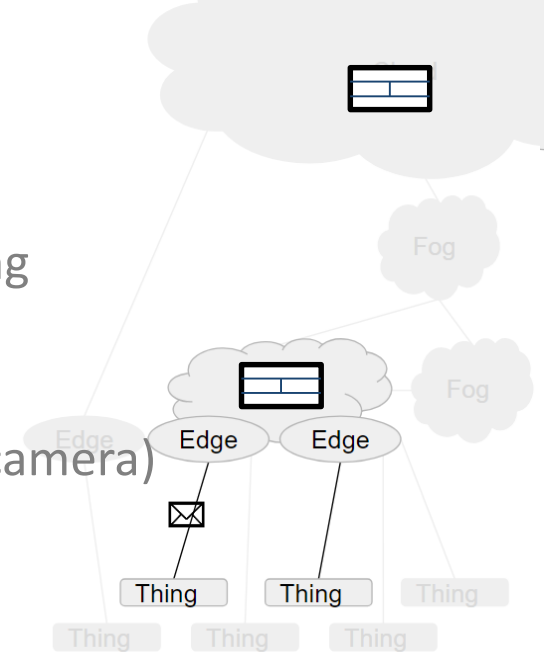
- Generic Interface Description Language (DeXIDL) & Generic Mediator

```
"protocol": "MQTT",
"operations": {
  "operation_1": {
    "type": "stream",
    "role": "consumer",
            "interaction_type":"one_way"
    "scope": "get_occupancy",
    "input_data" : "capacity,room"
  }
}
```

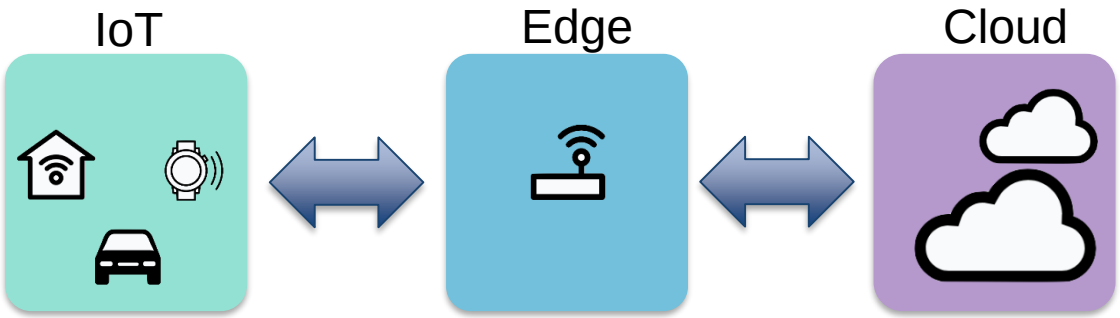
REST (common protocol)



The *Where* and *How* Problem

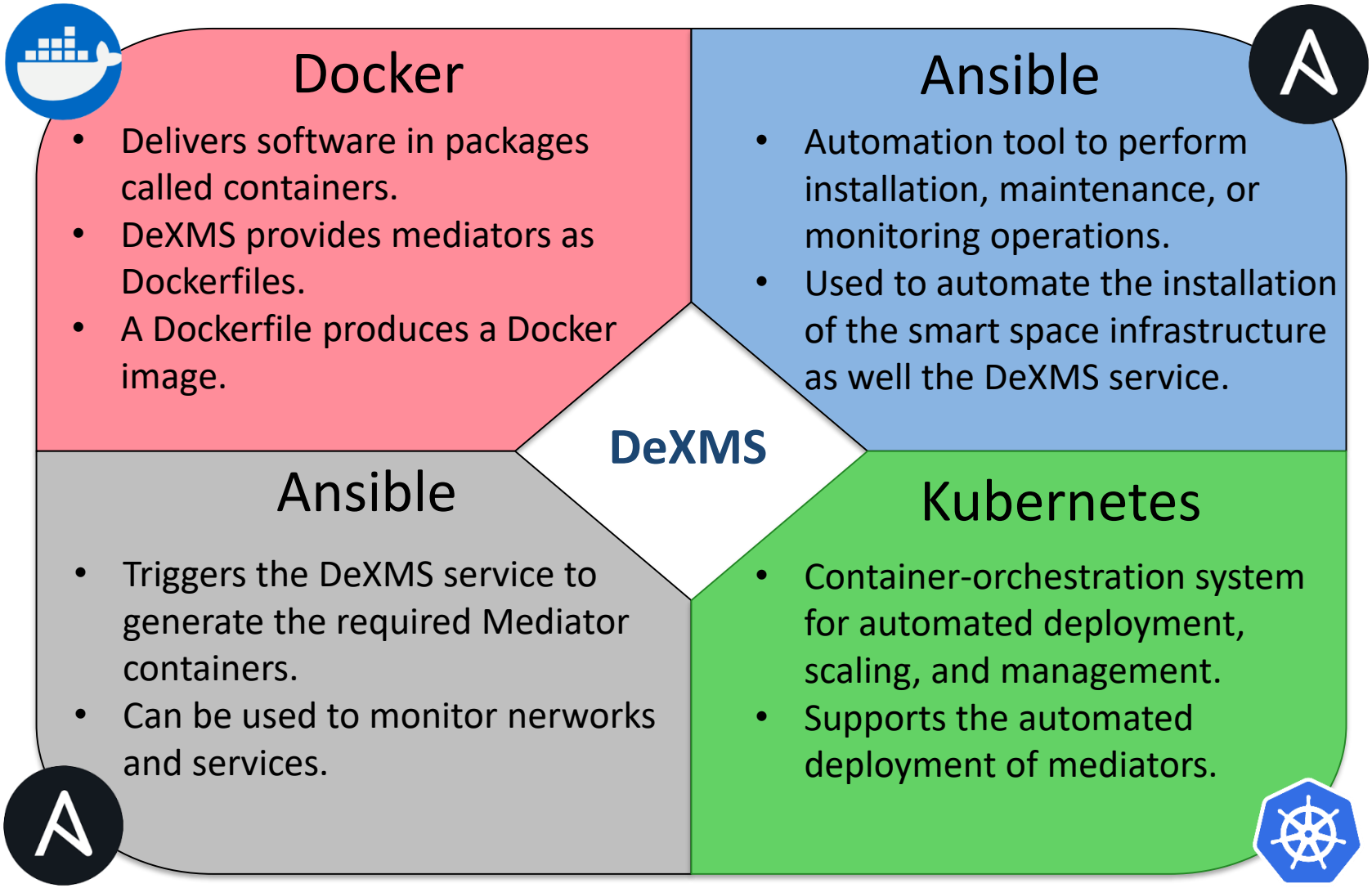


- *Where* to place mediators: Cloud, Edge, and Fog Computing
- Obvious solution: The Edge and Fog
  - Things push data to the cloud to be analyzed (e.g., 4k camera)
    - Use artifacts at the Edge/Fog to filter these data
  - Timeliness, data privacy, etc



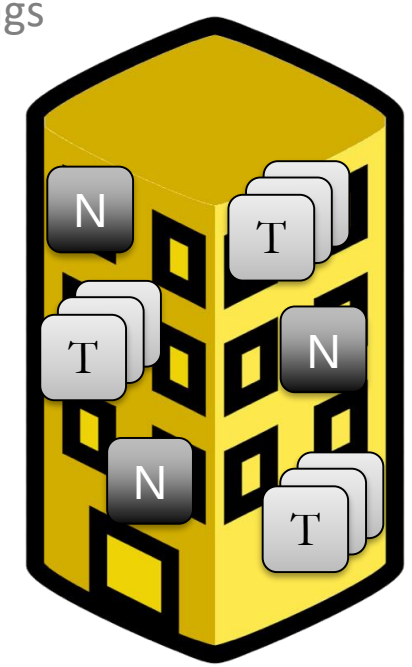
- Work in progress:
  - Systematic solution to automate the deployment of mediators at the Edge
  - Utilize this solution to deploy mediators and other artifacts in the I3 platform

Mediators at the Edge



The *How* Problem

- *How* to place mediators?
  - Related Problem: Operator Placement
  - Compute a "cost space"[1] to represent *Things* and *Physical Nodes*
    - E.g., a smart building with heterogeneous Things
  - Place mediators in an optimized manner



M

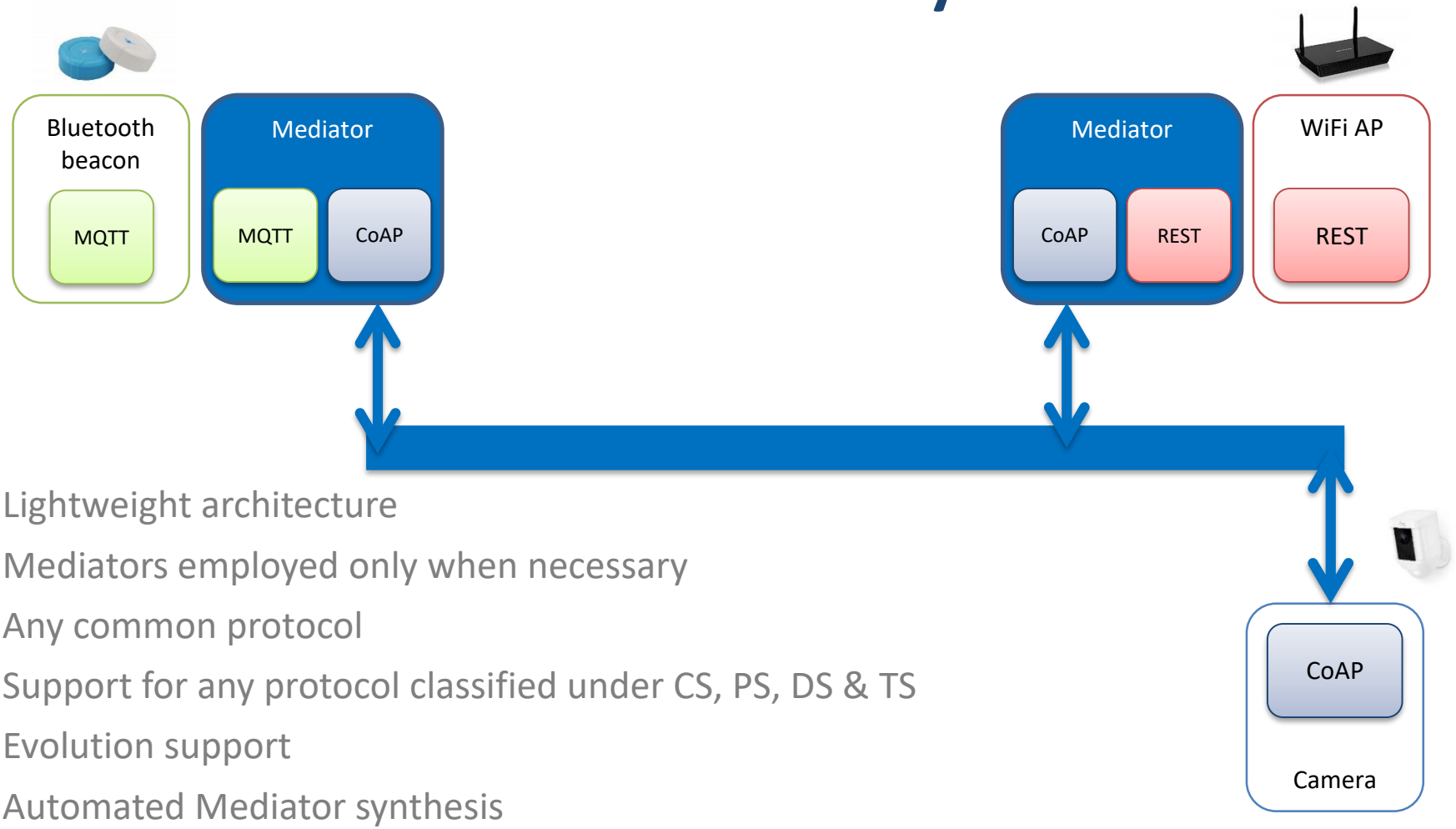
M

Criteria: distance, energy, bandwidth, latency, availability, etc

Optimization techniques[2,3]: constraint programming solvers, heuristics, linear programing, genetic programming, etc.

[1] P. Pietzuch et al., ICDE, 2006
[2] V. Issarny et al., ICDCS, 2019
[3] A. Chio et al., ARM, 2019

DeXMS novelty



- Lightweight architecture
- Mediators employed only when necessary
- Any common protocol
- Support for any protocol classified under CS, PS, DS & TS
- Evolution support
- Automated Mediator synthesis
- 75-96 % person-hours reduction when using DeXMS
- Work in progress: enabling application-layer data exchange[1]

[1] R. Yus et al., Buildsys, 2019

Software artifacts and adoption

- DeXMS is part of the zefxis[1] platform (https://gitlab.inria.fr/zefxis):
  - Mediator generator: https://gitlab.inria.fr/zefxis/DeXMS
  - Eclipse plugin for defining Things' DeXIDLs: https://gitlab.inria.fr/zefxis/DeX-IDL
  - Web console: https://gitlab.inria.fr/zefxis/IoT-web-console
- Demos:
  - Mediator generation: https://youtu.be/UgfM3810RS8 (ICSOC 2016)
  - Web console installation: https://youtu.be/IGjZ5u3QYOw (ICWE 2018)
  - Fire Detection scenario: https://youtu.be/SJeiqJkBhls (ICWE 2018)
- DeXMS is used as a core component in H2020 CHOReVOLUTION, UCI TIPPERS and Inria/UCI MINES projects.

[1]zefxis in ancient Greek (ζεῦξις) means connection or coupling

Publications

- G. Bouloukakis, N. Georgantas, P. Ntumba, V. Issarny, "Automated Synthesis of Mediators for Middleware-layer Protocol Interoperability in the IoT", FGCS Journal, 2019.
- R. Yus, G. Bouloukakis, S. Mehrotra, N. Venkatasubramanian, “Abstracting Interactions with IoT Devices Towards a Semantic Vision of Smart Spaces”, ACM Buildsys, November 2019, New York, USA
- V. Issarny, B. Billet, G. Bouloukakis, D. Florescu, C. Toma, “LATTICE: A Framework for Optimizing IoT System Configurations at the Edge”, ICDCS 2019, July 2019, Dallas, Texas, USA
- A. Chio, G. Bouloukakis, C.H. Hsu, S. Mehrotra, N. Venkatasubramanian. “Adaptive Mediation for Data Exchange in IoT Systems”, 18th ARM Workshop 2019, Davis, CA, USA

Thank you!