



# Automated Synthesis of Software Artifacts for Middleware-layer Protocol Interoperability in the IoT

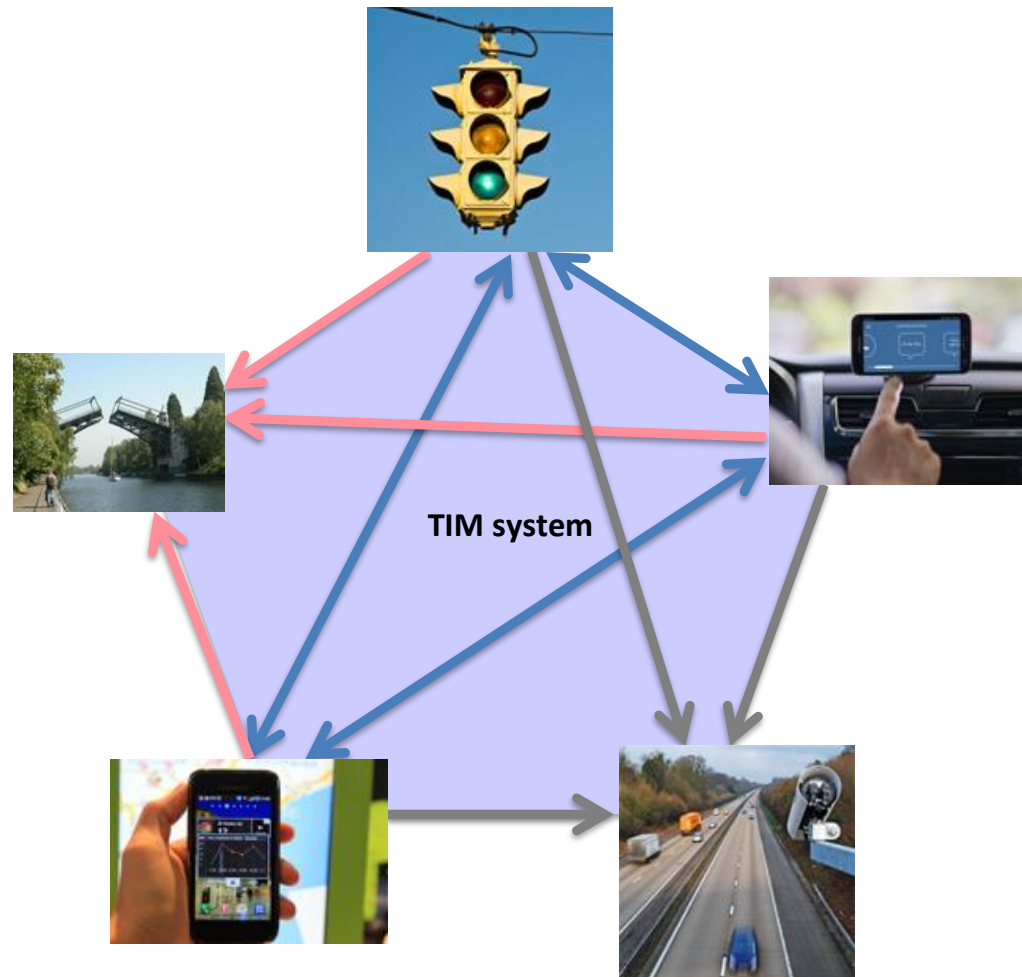
Georgios Bouloukakis

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

DSM Group, UC Irvine, June 2018

# Enabling Emergent mobile systems in the IoT

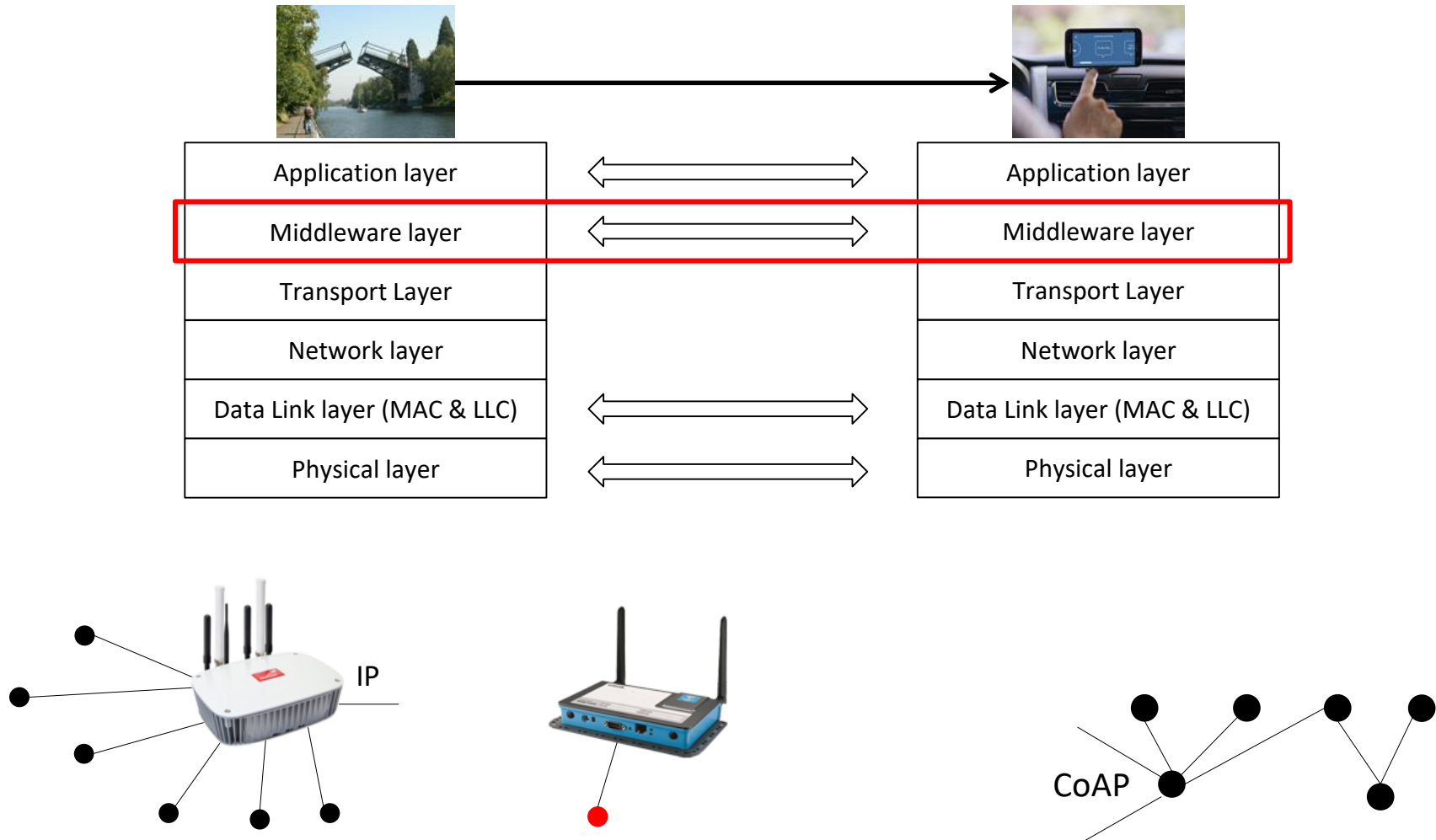
- Traffic Information Management (TIM) system:



**Heterogeneous**

**Dynamic**

# IoT heterogeneity at multiple layers



# Middleware protocols in the mobile IoT



DPWS

CoAP

MQTT

ZeroMQ

WebSockets

....

Client-server

Pub/sub

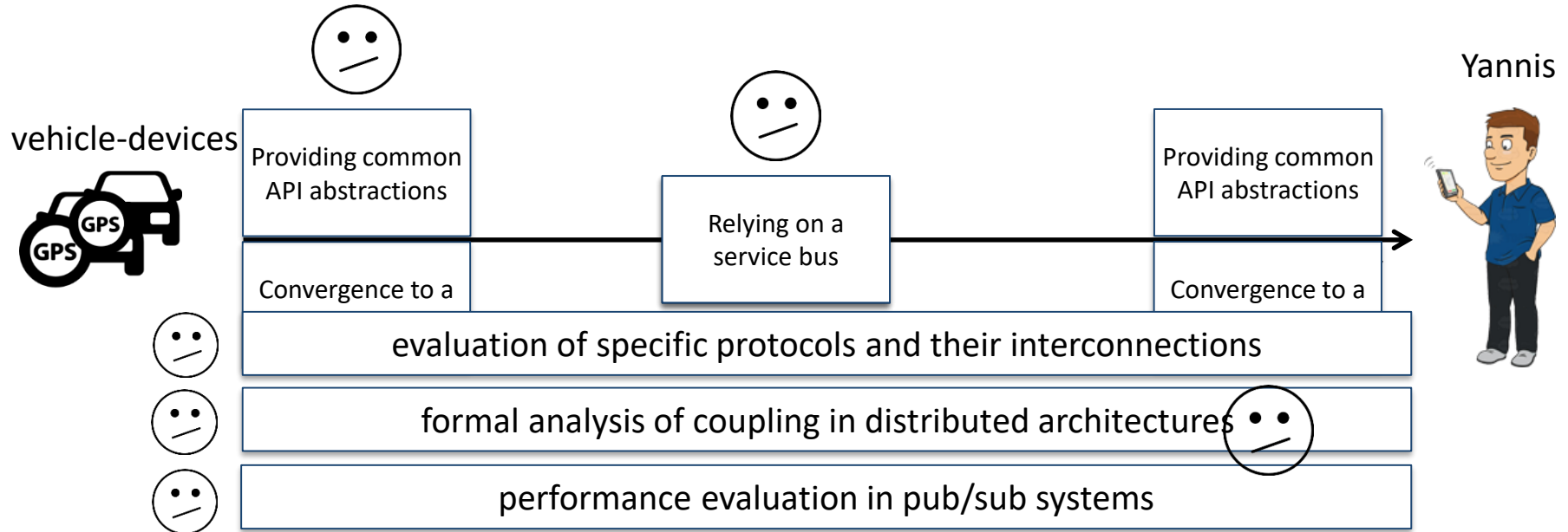
Streaming ....

reliable/unreliable

mobile connectivity

....

# Heterogeneous interconnections in the mobile IoT



- ❑ How to enable interconnections in the mobile IoT ?
- ❑ What is the end-to-end QoS of the interconnection ?

# Our proposed solution

vehicle-devices



Protocol X

systematic solution to  
interoperability

end-to-end  
performance analysis

Protocol Y

Yannis



e.g. CoAP server

- push-based
- resource
- data feeds lifetime
- unreliable

*functional semantics*

*QoS semantics*

e.g. MQTT subscriber

- pub/sub
- topic
- mobile connectivity
- reliable

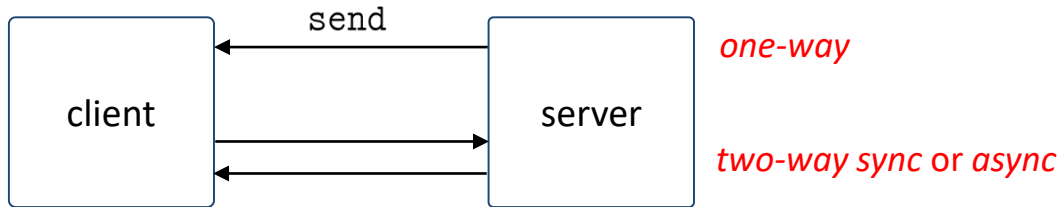
## Automated synthesis of interoperability artifacts:

- enables functional middleware-layer interoperability

## End-to-end performance modeling & analysis:

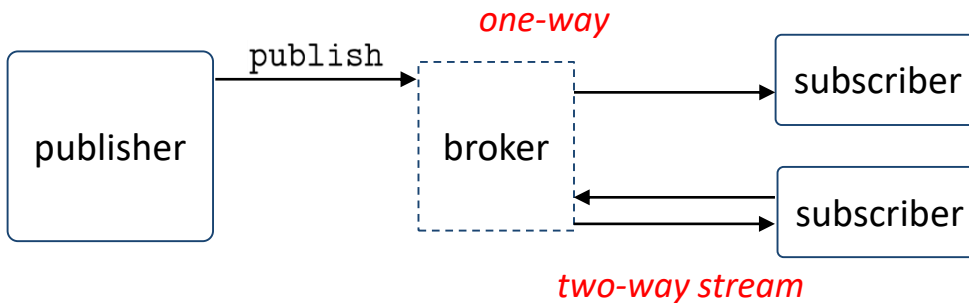
- evaluates the interoperability effectiveness

# Models for core interaction paradigms



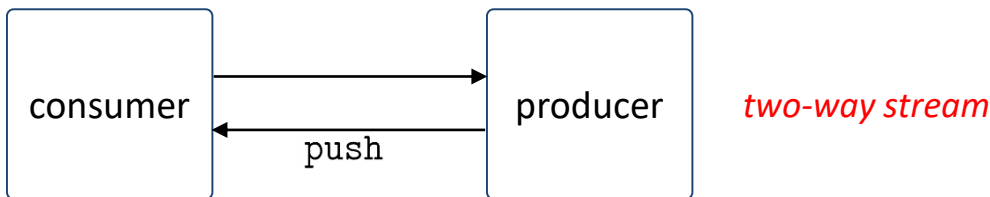
Client-Service (CS)

- Tight Time & Space Coupling



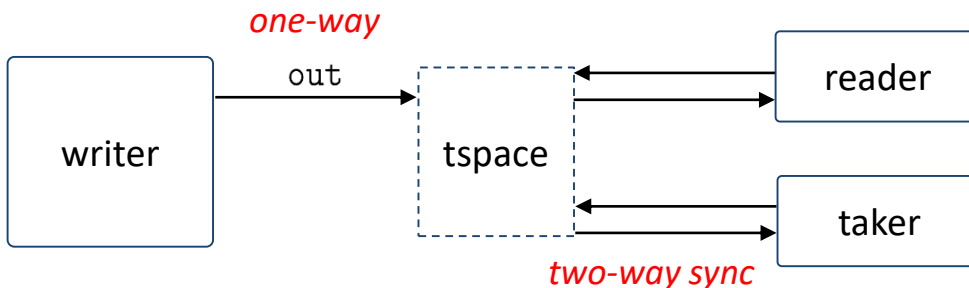
Publish-Subscribe (PS)

- Time & Space Decoupling



Data Streaming (DS)

- Tight Time & Space Coupling

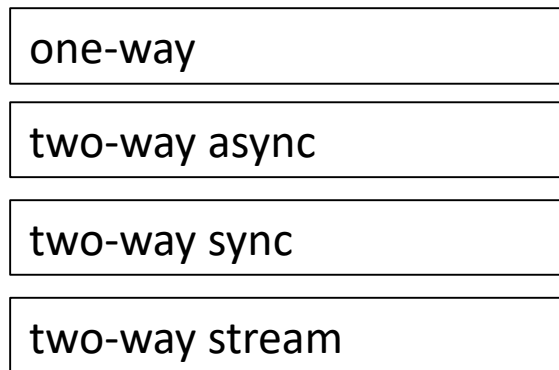


Tuple Space (TS)

- Time & Space Decoupling

# Generic Middleware (GM) connector model

- Our generic connector defines 4 basic interaction types:



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

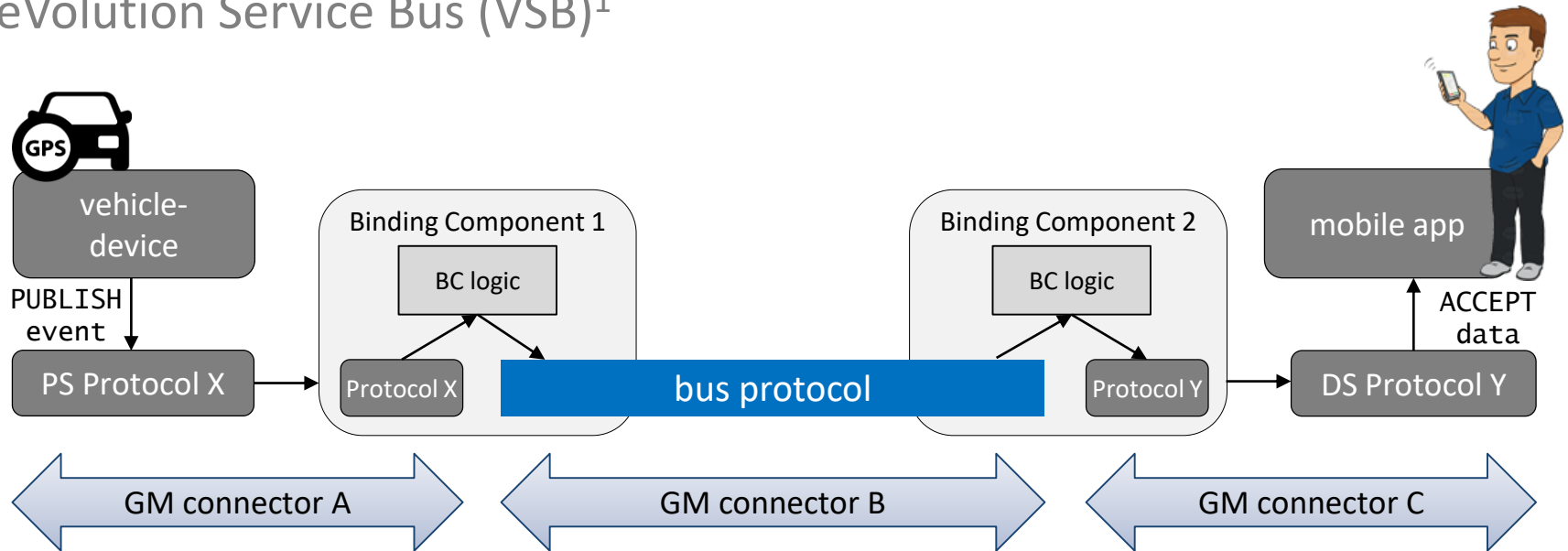
**post** and **get** primitives abstract CS, PS, DS and TS primitives

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



# Our middleware protocol interoperability solution

## ➤ eVolution Service Bus (VSB)<sup>1</sup>

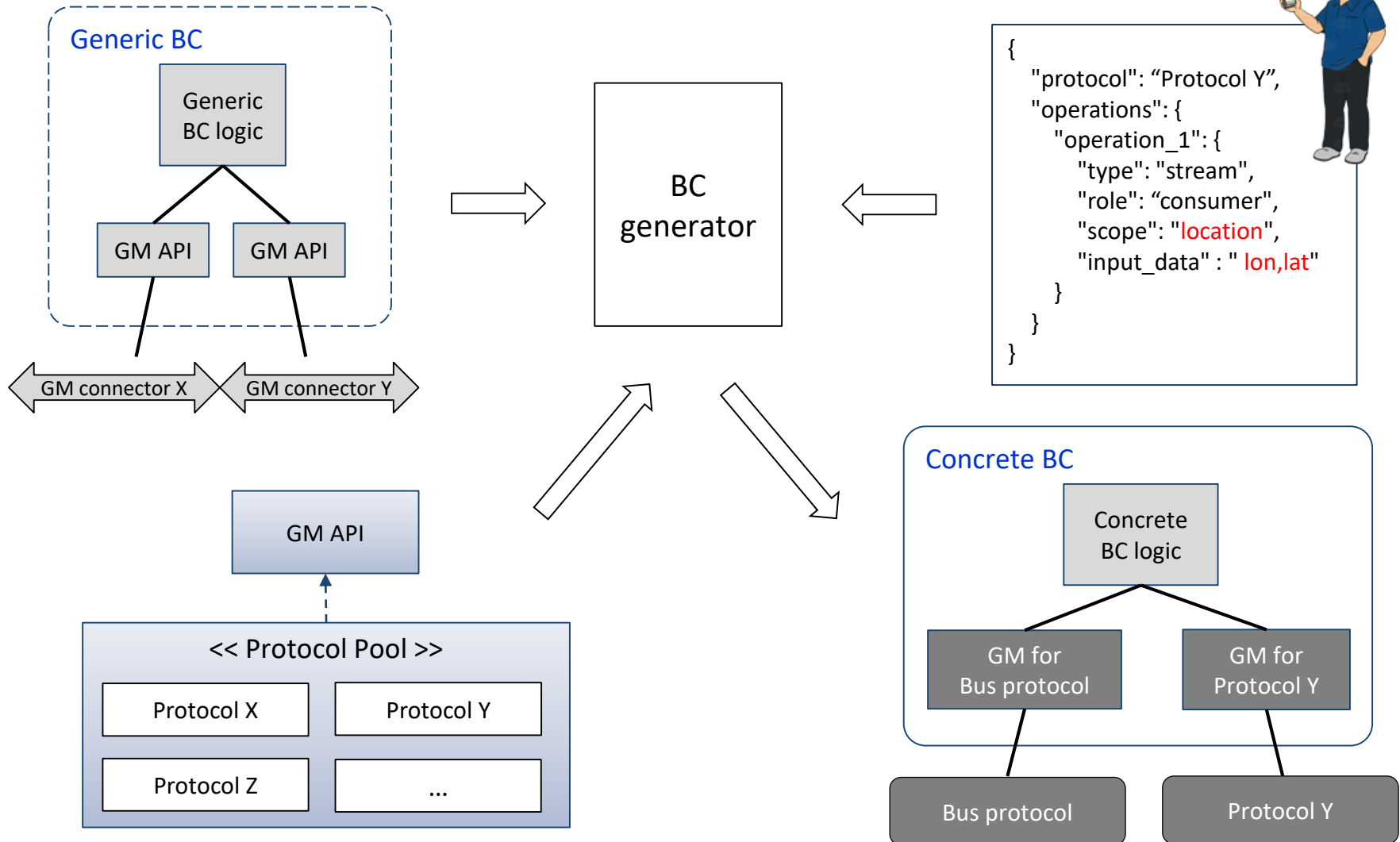


- ❑ BC architecture: relies on GM for automated BC synthesis
- ❑ Primitives & data conversion between the bus protocol and the Things' protocols
- ❑ A universal way to describe the Things' I/O required

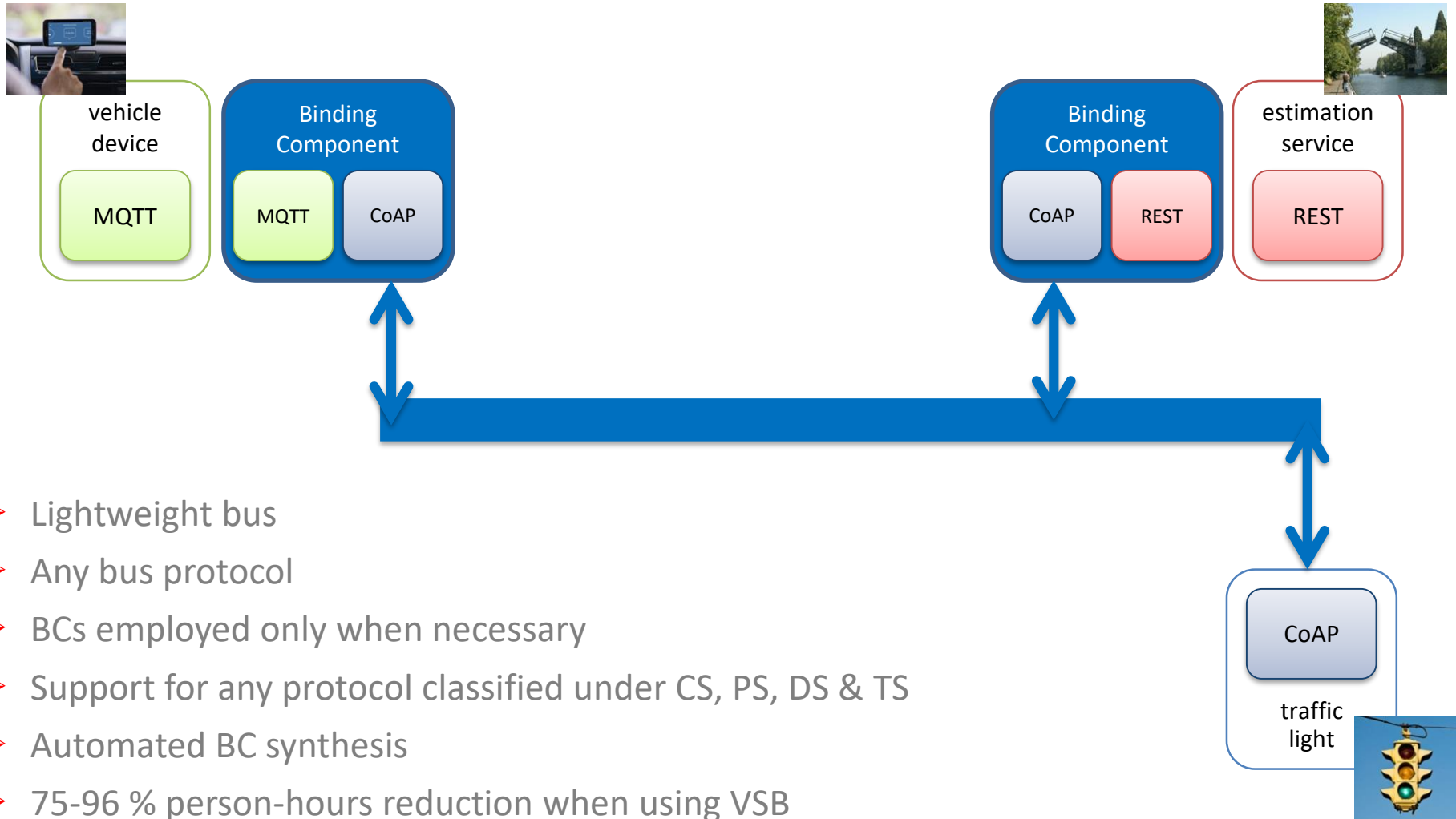
<sup>1</sup> G. Bouloukakis et al., ICSOC, 2016

# Automated BC synthesis

## ➤ Generic Interface Description Language (GIDL) & Generic BC



# VSB novelty



- Lightweight bus
- Any bus protocol
- BCs employed only when necessary
- Support for any protocol classified under CS, PS, DS & TS
- Automated BC synthesis
- 75-96 % person-hours reduction when using VSB
- Evolution support

# Software artifacts and adoption

- VSB is part of the zefxis<sup>1</sup> platform (<https://gitlab.inria.fr/zefxis>):
  - BC generator: <https://gitlab.inria.fr/zefxis/BCgenerator>
  - Eclipse plugin for defining Things' GIDLs: <https://gitlab.inria.fr/zefxis/GIDL>
  - Web console: <https://gitlab.inria.fr/zefxis/loT-web-console>
- Demos:
  - BC generation: <https://youtu.be/UgfM3810RS8>
  - Web console installation: <https://youtu.be/IGjZ5u3QYOW>
  - Fire Detection scenario: <https://youtu.be/SJeiqJkBhls>
- VSB is used as a core component in H2020 CHOReVOLUTION project



# Thank you!