Fog Computing and the connected car

R. Vilalta, R. Casellas, R. Martínez, R. Muñoz

CTTC

WIRELESS WORLD RESEARCH FORUM

Why do we need a connected car?

Fog computing is…

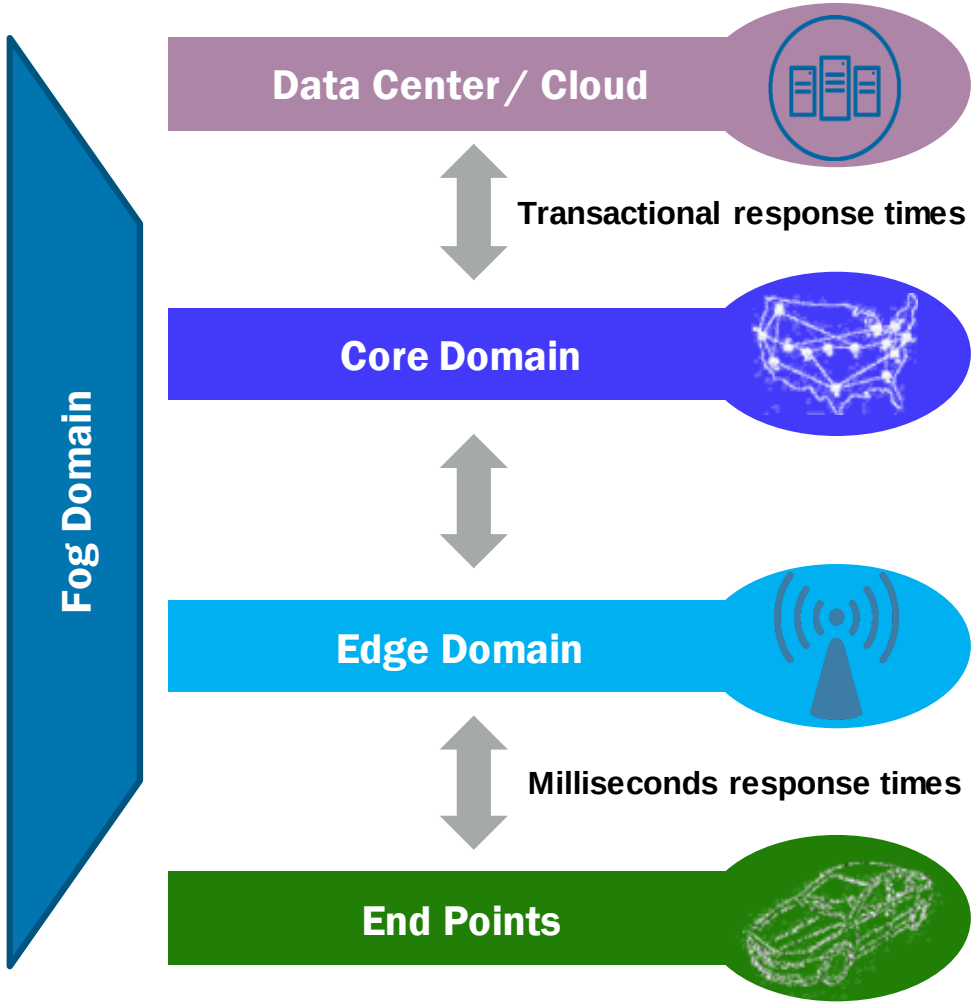
A system-level architecture to extend

COMPUTE

NETWORK

STORAGE

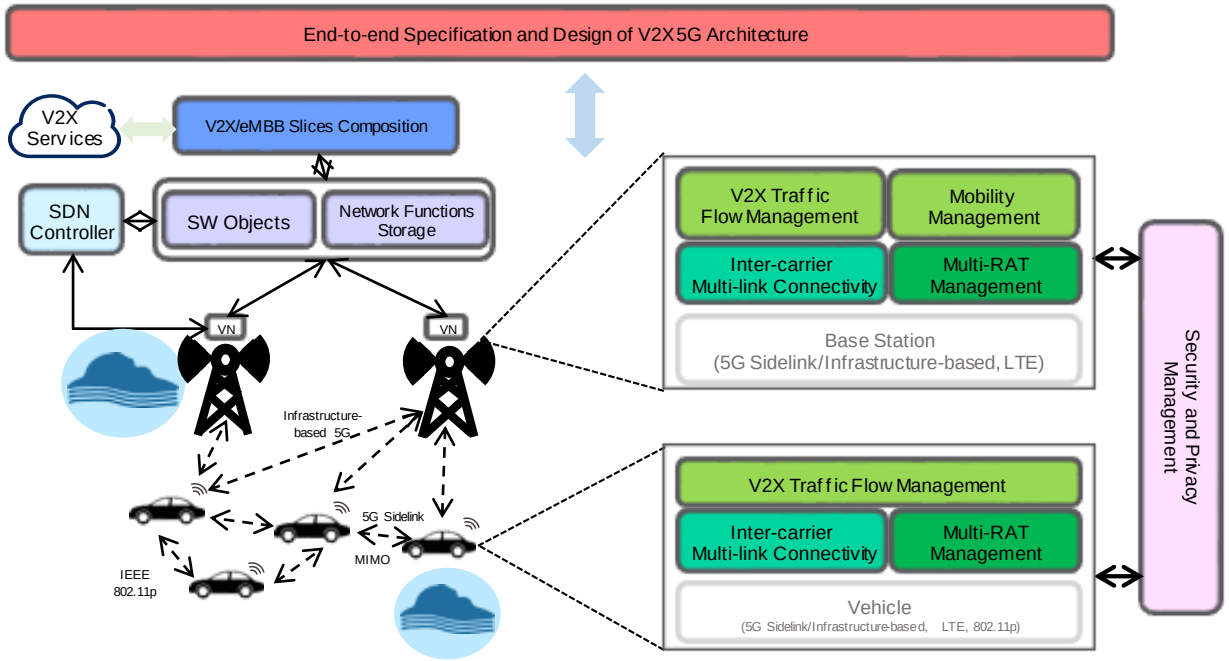
capabilities of Cloud to the Edge of the network



Fog computing is an enabler for the connected car

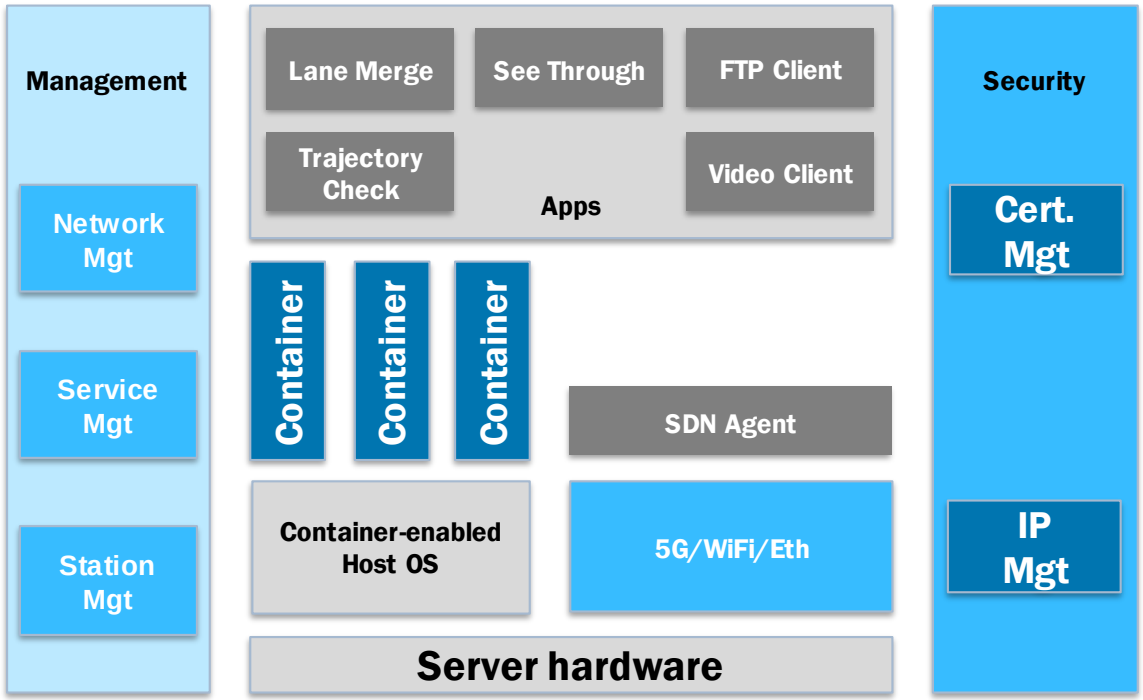
Fog Computing architecture. Where to locate a fog node?

- Inside the car, in order to offer the various third party OAM services and applications on top of the same infrastructure
- On the BTS, where RAN information can be accessed in real time (MEC). Moreover, this location could allow the allocation of ITS services and applications near the edge of the network in order to provide low-latency. Multi-Access Mobile Computing



Fog Node architecture

- How can we integrate applications from different OAMs?
- Simplify vehicle control architecture, reduce control system weight and cost of software development
- Vehicular Control Unit based on containers



A Data Modelling Language for the Connected Car

- ETSI EN 302 637-2, "Specification of Cooperative Awareness Basic Service", September 2014.
  - ASN.1 ➔ JSON Encoding Rules (JER)
- 5GCAR: Cooperative Service Message (CSM) format The basic encoding is JSON.
- Example: Road User (RU) description:

How do we combine several data models from different technologies suc as NFV, MEC, and IoT?


```
{"type":"ru_description",
"subtype":"connected",
"origin":"self",
"timestamp":1504282117000,  // time in ms since 01/01/1970 UTC
"ru_type":"vehicle",  // or station_type like CAM ?
"position":["latitude":48.800534,"longitude":2.296025,"altitude":42],
"heading":145,  // degrees clockwise (0 & 360 means North)
"speed":15, // m/s
"acceleration":2.6,  // m/s²
"size":["length":4.5,  "width":2.1],  // m
"lane_position":1,  // counted from the rightmost lane in driving direction
"uuid":"468754354",  // message origin UUID (either road user or intelligent camera system)
"performance":["max_acceleration":4,  "max_decceleration":9],  // m/s²
"confidence":[" latitude":0.0001,"longitude:0.0001,"altitude":1,
"heading":0.5,"speed":0.1,"acceleration":0.05],  // accuracies of the provided information
"signature":"6578"}
```

YANG

- YANG is a data modelling language defined by the IETF, and specified in RFC 7950.
- A focus on readers and reviewers
  - Text-base: standards friendly
- Limited Scope, but extensible
- Ability to model config data, state data, RPCs, and notifications
- Experience gained by existing implementations.
- Running code, lots of support libraries.
- Together with RESTconf it might translate to: REST API + JSON (or XML) messages

Easy to read, easy to learn

```
container system {
    leaf host-name {
        type inet:host;
       description "Name of this host";
    }
    container services {
        container ssh {
            presence "Enables SSH";
            description "SSH service specific
configuration";
            // more leafs, containers and stuff here...
        }
    }
}
```

```
{
  “system”: {
    “host-name” : “fred”,
    “services” : {
      “ssh” : {}
    }
  }
}
```

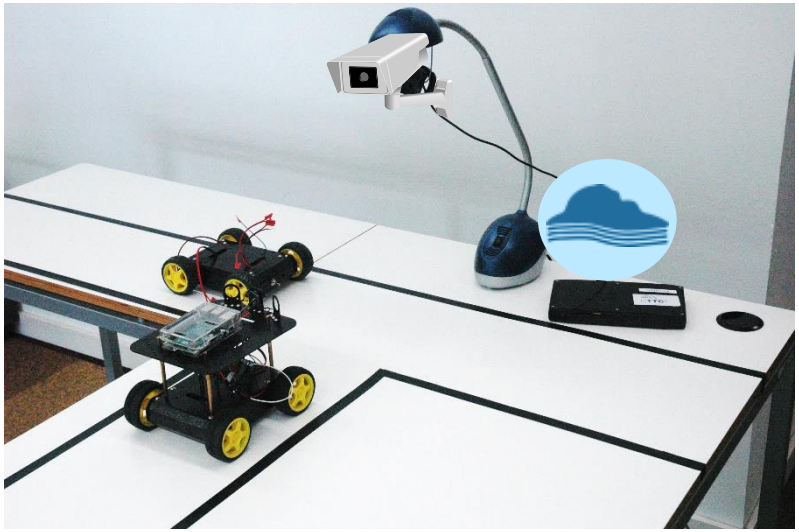
RESTCONF

- RESTCONF
  - Almost RFC
  - RESTful protocol to access YANG defined data
  - Representational State Transfer, i.e. server maintains no session state
  - URIs reflect data hierarchy
  - HTTP as transport
  - Data encoded with JSON (or XML)
  - Operations :

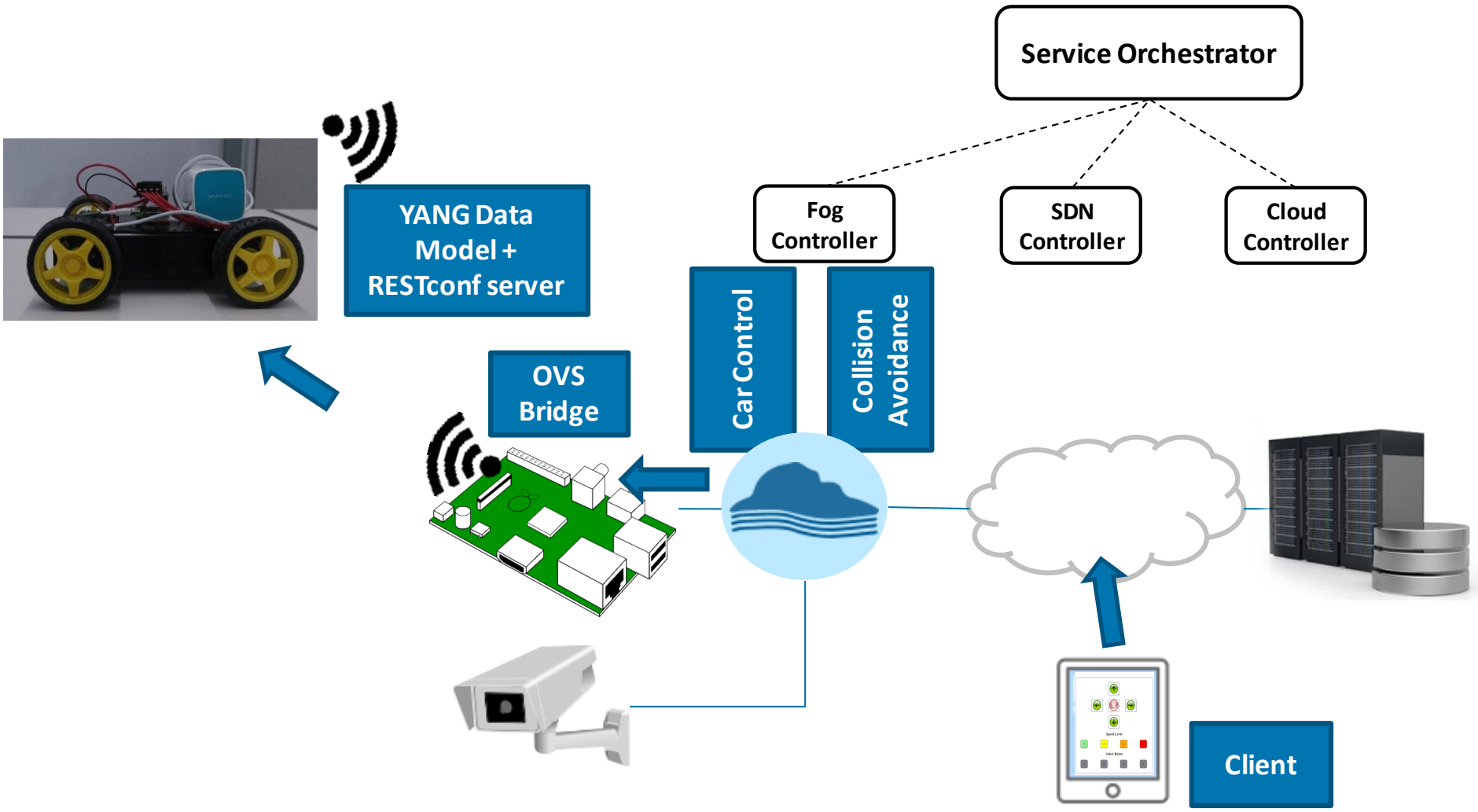
RESTCONF	Netconf
GET	<get-config>, <get>
POST	<edit-config> (“create”)
PUT	<edit-config> (“replace”)
PATCH	<edit-config> (“merge”)
DELETE	<edit-config> (“delete”)
OPTIONS	(discover supported operations)
HEAD	(get without body)

Our toy demo proposal

- Objective: Demonstrate advances in:
  - V2X Communications
  - Network infrastructure
  - Fog computing
- Use case(s):
  - Similar to lane merging
  - Focused in Collision Avoidance.



Proposed architecture and experimental assessment



Conclusion

- Presented the concept of fog computing as an enabler of the connected car.
- Architecture for Fog Computing
- Presented a Proof-of-concept for a remotely-controlled car using YANG modeling for the data associated to the car, and operating an SDN/NFV network with dynamic service provisioning.

Thank you! Questions?

ricard.vilalta@cttc.es
http://networks.cttc.es

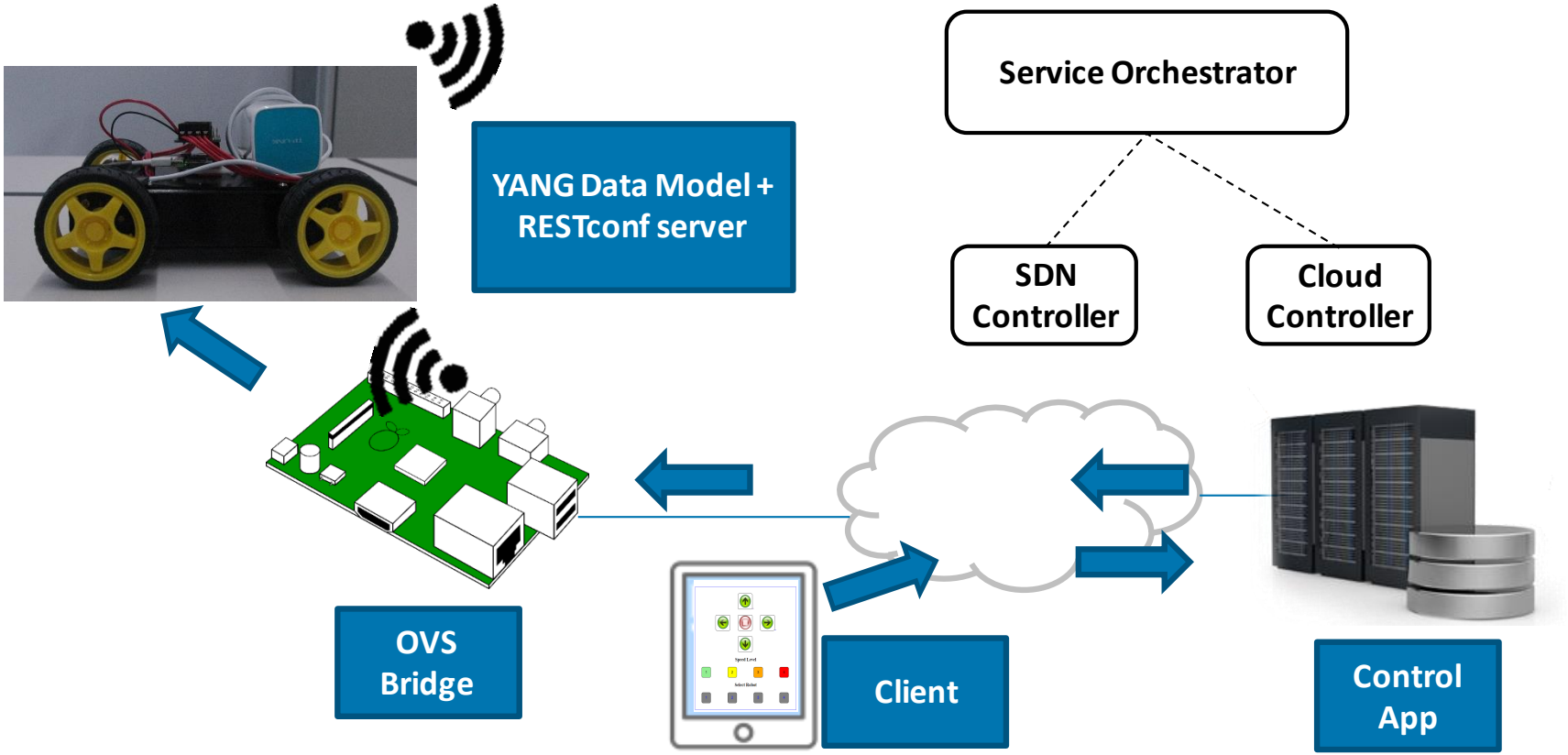
CTTC

WIRELESS WORLD
RESEARCH FORUM

AENOR
ER
Gestión I+D+i
UNE 166002
IDI-0024/2011

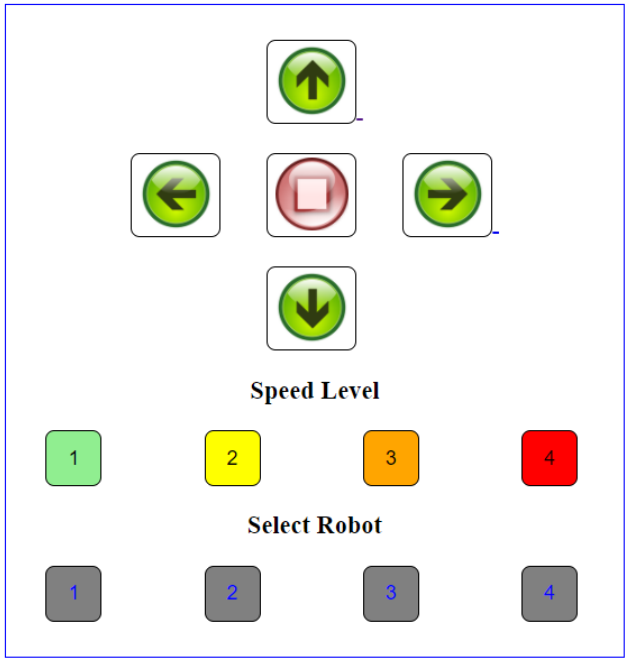
WE HAVE STARTED WITH YANG AND CLOUD COMPUTING

First experimental set-up: Using YANG + Cloud computing



Proof of Concept: The YANG-based cloud connected car

- Web-based UI

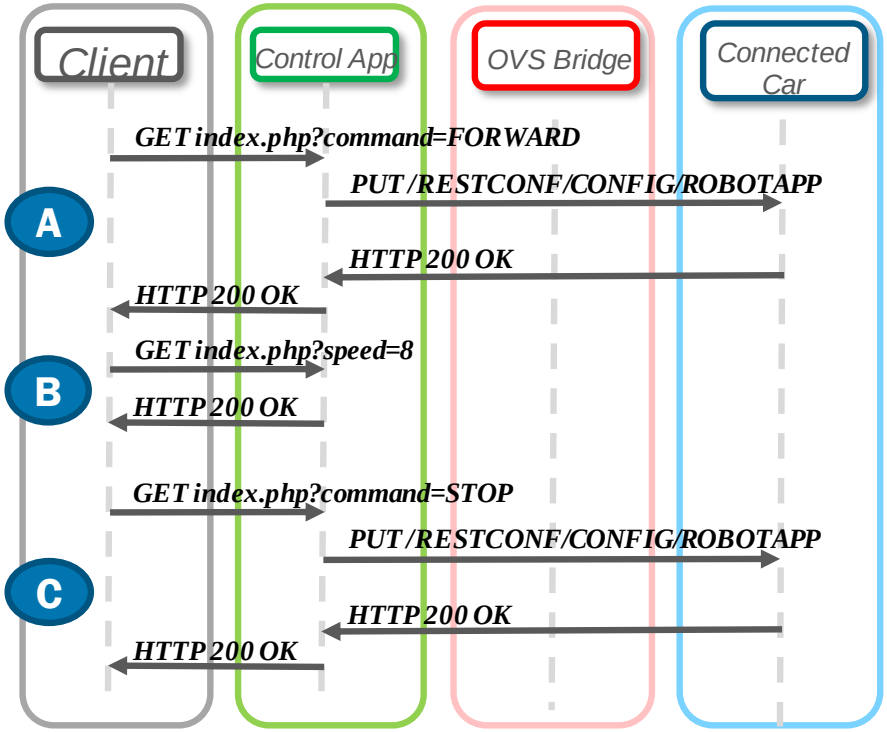


- JSON commands

```
▿ Object
  ▿ Member Key: robotId
      String value: robot1
      Key: robotId
  ▿ Member Key: speed
      Number value: 5
      Key: speed
  ▿ Member Key: command
      String value: FORWARD
      Key: command
```

Message workflow

- Three Scenarios:
  - Scenario A: Move the connected car
  - Scenario B: Speed selection
  - Scenario C: Stop the connected car



Wireshark capture

	Time	Source	Destination	Protocol	Info
A	*REF*	10.1.16.59	10.1.2.202	HTTP	GET /index.php?comando=FORWARD HTTP/1.1
	0.013118095	10.1.2.202	172.24.1.5	HTTP	PUT /restconf/config/robotApp/ HTTP/1.1
	0.097723150	172.24.1.5	10.1.2.202	HTTP	HTTP/1.0 200 OK (application/json)
	0.098776603	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)
B	*REF*	10.1.16.59	10.1.2.202	HTTP	GET /index.php?speed=8 HTTP/1.1
	0.004280350	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)
C	*REF*	10.1.16.59	10.1.2.202	HTTP	GET /index.php?comando=STOP HTTP/1.1
	0.014779584	10.1.2.202	172.24.1.5	HTTP	PUT /restconf/config/robotApp/ HTTP/1.1
	0.105489993	172.24.1.5	10.1.2.202	HTTP	HTTP/1.0 200 OK (application/json)
	0.106600629	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)