Fog Computing and the connected car

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

  - ASN.1 ➔ JSON Encoding Rules (JER)
- 5GCAR: Cooperative Service Message (CSM) format  The basic encoding is JSON.
- Example: Road User (RU) description:

```json
{
  "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"
}
```

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

```json
{
  "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:

<table>
<thead>
<tr>
<th>RESTCONF</th>
<th>Netconf</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>&lt;get-config&gt;, &lt;get&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>&lt;edit-config&gt; (&quot;create&quot;)</td>
</tr>
<tr>
<td>PUT</td>
<td>&lt;edit-config&gt; (&quot;replace&quot;)</td>
</tr>
<tr>
<td>PATCH</td>
<td>&lt;edit-config&gt; (&quot;merge&quot;)</td>
</tr>
<tr>
<td>DELETE</td>
<td>&lt;edit-config&gt; (&quot;delete&quot;)</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>(discover supported operations)</td>
</tr>
<tr>
<td>HEAD</td>
<td>(get without body)</td>
</tr>
</tbody>
</table>
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

- Service Orchestrator
- YANG Data Model + RESTconf server
- OVS Bridge
- Car Control
- Collision Avoidance
- Fog Controller
- SDN Controller
- Cloud Controller
- Client
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
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

Select Robot

1  2  3  4

Speed Level

1  2  3  4
**Message workflow**

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>REF</em></td>
<td>10.1.16.59</td>
<td>10.1.2.202</td>
<td>HTTP</td>
<td>GET /index.php?comando=FORWARD HTTP/1.1</td>
</tr>
<tr>
<td>0.013118095</td>
<td>10.1.2.202</td>
<td>172.24.1.5</td>
<td>HTTP</td>
<td>PUT /restconf/config/robotApp/ HTTP/1.1</td>
</tr>
<tr>
<td>0.097723150</td>
<td>172.24.1.5</td>
<td>10.1.2.202</td>
<td>HTTP</td>
<td>HTTP/1.0 200 OK (application/json)</td>
</tr>
<tr>
<td>0.098776603</td>
<td>10.1.2.202</td>
<td>10.1.16.59</td>
<td>HTTP</td>
<td>HTTP/1.1 200 OK (text/html)</td>
</tr>
<tr>
<td><em>REF</em></td>
<td>10.1.16.59</td>
<td>10.1.2.202</td>
<td>HTTP</td>
<td>GET /index.php?speed=8 HTTP/1.1</td>
</tr>
<tr>
<td>0.004280350</td>
<td>10.1.2.202</td>
<td>10.1.16.59</td>
<td>HTTP</td>
<td>HTTP/1.1 200 OK (text/html)</td>
</tr>
<tr>
<td><em>REF</em></td>
<td>10.1.16.59</td>
<td>10.1.2.202</td>
<td>HTTP</td>
<td>GET /index.php?comando=STOP HTTP/1.1</td>
</tr>
<tr>
<td>0.014779584</td>
<td>10.1.2.202</td>
<td>172.24.1.5</td>
<td>HTTP</td>
<td>PUT /restconf/config/robotApp/ HTTP/1.1</td>
</tr>
<tr>
<td>0.105489993</td>
<td>172.24.1.5</td>
<td>10.1.2.202</td>
<td>HTTP</td>
<td>HTTP/1.0 200 OK (application/json)</td>
</tr>
<tr>
<td>0.106600629</td>
<td>10.1.2.202</td>
<td>10.1.16.59</td>
<td>HTTP</td>
<td>HTTP/1.1 200 OK (text/html)</td>
</tr>
</tbody>
</table>