Evaluating the Performance of CoAP, MQTT, and HTTP in Vehicular Scenarios

Roberto Morabito, Zakaria Laaroussi, Jaime Jiménez (Ericsson Research)

Reference at the paper: https://goo.gl/2r6RMJ
Outline

1. Introduction
2. Scenario
3. Testbed Setup
4. Performance Evaluation
5. Empirical Results
6. Conclusion and Future Perspectives
Introduction

• Purpose of the Study
  – Evaluating the performance (throughput and latency) of MQTT, CoAP, and HTTP in vehicular scenarios.
  – It can be considered a preliminary performance evaluation.
  – Empirically demonstrate whether there are performance differences when comparing edge-based service provisioning and cloud-based service provisioning.
Scenario

Edge–based service provisioning

Cloud–based service provisioning

Data Center

Base Station

Base Station

In-Car On Board Unit

Edge Server

Base Station
Testbed Setup – Implementation Plan
Testbed Setup – Current Status

- Edge Entity #1
- Data Center
- Vehicle-to-Vehicle Communication (X)
- Vehicle-to-Edge Communication (✓)
- Vehicle-to-Cloud Communication (✓)
Testbed Setup – More details (i)

- **Data Center**
  - Located in Lund (Sweden).
  - OpenStack environment.

- **Edge Entity**
  - Located in Jorvas (Finland).
  - Dell Precision T5500 server.
  - Connection with the base station through the mobile network of a Finnish operator.
  - No local breakout between base station and edge server (we fully rely on the network setup provided by the mobile network operator).
• **In-Car On-Board Unit**
  – Deployed in a general-purpose board, such as Raspberry Pi 3.
  – Connectivity is provided by a system combining a Sixfab base shield and a Quectel EC25 Mini PCIe 4G/LTE module. Both boards are in turn connected to the RPi3 through GPIO interface.
Testbed Setup – Ongoing studies

• **Cloud-based vs. Edge-based service provisioning** ✔
  – Shed light on the performance impact that a service provisioning provided from the cloud or from the edge introduces

• **Comparison among different application layer protocols** ✔
  – MQTT | CoAP | HTTP (through 4G connection)
  – Transmission of small-sized payloads

• **Service provisioning of large-sized payloads** ✔✖
  – The testbed is already set up for providing additional services to the vehicle (e.g., Multimedia contents)
  – Empirical evaluation must still be made

• **Testing and comparing different Radio interfaces** ✔✖
  – 4G vs. Wi-Fi
  – 802.11p for Car-to-Car communication
Performance Evaluation – Background

• **Main goals**
  – Evaluate two service provisioning approaches
    – Cloud-based vs. Edge-based
  – Application layer comparison
    – Transmission of small-sized messages
      – CoAP, MQTT, HTTP
  – Impact of additional factors
    – Vehicle’s speed
    – Number of clients
    – QoS in MQTT
# Performance Evaluation – Background*

<table>
<thead>
<tr>
<th>Criteria</th>
<th>HTTP</th>
<th>CoAP</th>
<th>MQTT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>Client/Server</td>
<td>Client/Server or Client/Broker</td>
<td>Client/Broker</td>
</tr>
<tr>
<td><strong>Abstraction</strong></td>
<td>Request/Response</td>
<td>Request/Response or Publish/Subscribe</td>
<td>Publish/Subscribe</td>
</tr>
<tr>
<td><strong>Header Size</strong></td>
<td>Undefined</td>
<td>4 Byte</td>
<td>2 Byte</td>
</tr>
<tr>
<td><strong>Message size</strong></td>
<td>Large and Undefined (depends on the web server or the programming technology)</td>
<td>Small and Undefined (normally small to fit in single IP datagram)</td>
<td>Small and Undefined (up to 256 MB maximum size)</td>
</tr>
<tr>
<td><strong>Semantics/Methods</strong></td>
<td>Get, Post, Head, Put, Patch, Options, Connect, Delete</td>
<td>Get, Post, Put, Delete</td>
<td>Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close</td>
</tr>
<tr>
<td><strong>Quality of Service (QoS) /Reliability</strong></td>
<td>Limited (via Transport Protocol - TCP)</td>
<td>Confirmable Message or Non-confirmable Message</td>
<td>QoS 0 - At most once QoS 1 - At least once QoS 2 - Exactly once</td>
</tr>
<tr>
<td><strong>Transport Protocol</strong></td>
<td>TCP</td>
<td>UDP, TCP</td>
<td>TCP (MQTT-SN can use UDP)</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>TLS/SSL</td>
<td>DTLS/IPSEC</td>
<td>TLS/SSL</td>
</tr>
<tr>
<td><strong>Default Port</strong></td>
<td>80/443 (TLS/SSL)</td>
<td>5683 (UDP)/5684 (DTLS)</td>
<td>1883/8883 (TLS/SSL)</td>
</tr>
</tbody>
</table>

Performance Evaluation – Background

• Application later protocols – Setup
  – MQTT
    Mosquitto ([https://mosquitto.org/](https://mosquitto.org/))
    MQTT benchmark tool ([https://github.com/krylovsk/mqtt-benchmark](https://github.com/krylovsk/mqtt-benchmark))
  – HTTP
    Apache HTTP Server Project ([https://httpd.apache.org/](https://httpd.apache.org/))
    ab benchmark tool ([https://httpd.apache.org/docs/2.4/programs/ab.html](https://httpd.apache.org/docs/2.4/programs/ab.html))
  – CoAP
    libcoap ([https://libcoap.net/](https://libcoap.net/))
    CoAPBench ([https://www.eclipse.org/californium/](https://www.eclipse.org/californium/))
Empirical Results – Impact of vehicle’s speed

- No strict dependence between vehicle's speed increase and throughput/latency variation.
- CoAP is outperforming both MQTT (with QoS 2) and HTTP.
- Edge-based service provisioning shows performance benefits when compared to the cloud-based approach.
Empirical Results – Impact due to number of connected clients.

- Average throughput delivered to each client decreases when the number of connected clients grows
- CoAP still the most efficient protocol (10% throughput decrease)
Empirical Results – Impact of **QoS in MQTT**

<table>
<thead>
<tr>
<th>QoS 0</th>
<th>QoS 1</th>
<th>QoS 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels" alt="Image" /></td>
<td><img src="https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels" alt="Image" /></td>
<td><img src="https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels" alt="Image" /></td>
</tr>
</tbody>
</table>

- **QoS 0**
  - Best-effort delivery.
  - No guarantee of delivery.
  - Recipient does not acknowledge receipt of the message and the message is not stored and retransmitted by the sender.

- **QoS 1**
  - It guarantees that a message is delivered at least one time to the receiver.
  - The sender stores the message until it gets a packet from the receiver that acknowledges receipt of the message.
  - Message can be sent or delivered multiple times.

- **QoS 2**
  - It guarantees that each message is received only once by the intended recipients.
  - Safest and slowest QoS.
  - Guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver.

* [https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels](https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels)
Empirical Results – Impact of QoS in MQTT.

- Higher QoS produces a throughput reduction in the order of 40%.
- Higher QoS slows down the message transmission by approximately 75%
Conclusion

- **Conclusion**
  (small-sized messages transmission...to be kept in mind...)
  - CoAP outperforms MQTT (both QoS 1 and QoS 2) and HTTP, both from throughput and latency perspective in different case studies.
  - In MQTT, the choice of the QoS has effects in the produced performance.
  - Tangible performance gains when exploiting edge-based service provisioning.
Conclusion and Future Perspectives

- **Future Work**
  - Large-sized workloads.
  - Evaluate the impact of Object Security for Constrained RESTful Environments (OSCORE).
  - Additional network interfaces.
  - Different placement for the edge entity.

As a general rule, the choice of using a protocol over another highly depends on the use case under examination and the workloads intrinsically generated within them, without also neglecting the implementation choices of different service providers.