Cardiff University, School of Computer Science and Informatics

# Edge-driven Real-time Vehicular Services Initial Plan CM3203

Samuil Velichkov
C1418948

February 5, 2018

Supervisor: Prof. Omer Rana
Moderator: Dr. Bailin Deng

# 1 Project Description

The ever-increasing data usage in information and communication technology, especially with the growing popularity of the Internet of Things(IoT), is making it challenging for existing cloud-based, centralised systems to draw actionable insights in near real-time. In cloud-centric IoT applications, the sensor data is extracted, accumulated and processed at the public or private clouds, leading to significant latencies. This is a particularly relevant problem with the gaining popularity of connected and autonomous vehicles which are predicted to generate and consume roughly 4 terabytes of data, every eight hours of driving by 2020[1]. With the increasing feature set and the vast number of connected vehicles predicted, a more decentralised approach is necessary and it is "Fog & Edge Computing" which will be in the forefront of solving this data deluge.

Instead of moving big data to cloud resources for processing, computation is brought closer to the source, in micro data centres located at the edge of the network, closer to the user. It is expected that such architecture will be integrated with established communication infrastructures, in road-side radio towers and in the vehicles themselves. By doing this, communication latency will be reduced and real-time vehicular services such as ride sharing, car sharing, navigation-related services and safety systems such as emergency assistance and any other car capabilities, which otherwise rely on a cloud or back-office connection, would be greatly improved, leading to a better quality of service(QoS). Connected and autonomous vehicles will be in the center of the "Fog & Edge Computing" movement and will drive the improvement of intelligent transportation systems.

In this project I will evaluate the advantages and limitations of using "Fog" and "Edge" nodes by creating a functional internal car network which replicates a real connected car's CAN bus (Controller Area Network)[2] and address different complexity and security challenges associated with this approach. The system will be designed to collect or emulate numerous sensor information such as speed, acceleration, proximity, location etc. and process this data locally using distributed nodes in an attempt to reduce the amount of information relayed to cloud services and simulate as closely as possible a real CAN bus. Using the attained results, I will compare and contrast the proposed architecture optimisation with a cloud-based approach. Issues around security, such as potential attack patterns relevant to vehicles, privacy in the context of this work and potential integration with Cardiff University's "Formula Student" race car will also be explored.

# 2 Project Aims and Objectives

To achieve the aim of the project, to simulate an internal car network, namely the CAN bus, "Raspberry Pi" System on a Chip(SoC) boards will be used in conjunction with various types of sensors, in order to mimic a vehicle's Electronic Control Units(ECUs)[3]. An example of an ECU would be the ECU that takes sensor readings from the powertrain, and sends them to the Instrument cluster, which itself does the processing, interpreting and displaying of the data. The goal is to create a complete simulator of the car network, with one "Raspberry Pi" acting as a Smart Connected Vehicle Gateway(SCV) and multiple others(ECUs) relaying to it data collected from sensors e.g. Fig 1. The SCV then uses this data and processes it locally, obtaining concise, insightful information about the car's e.g. speed, location, proximity to other vehicles and more, thus acting as an "Edge" node.

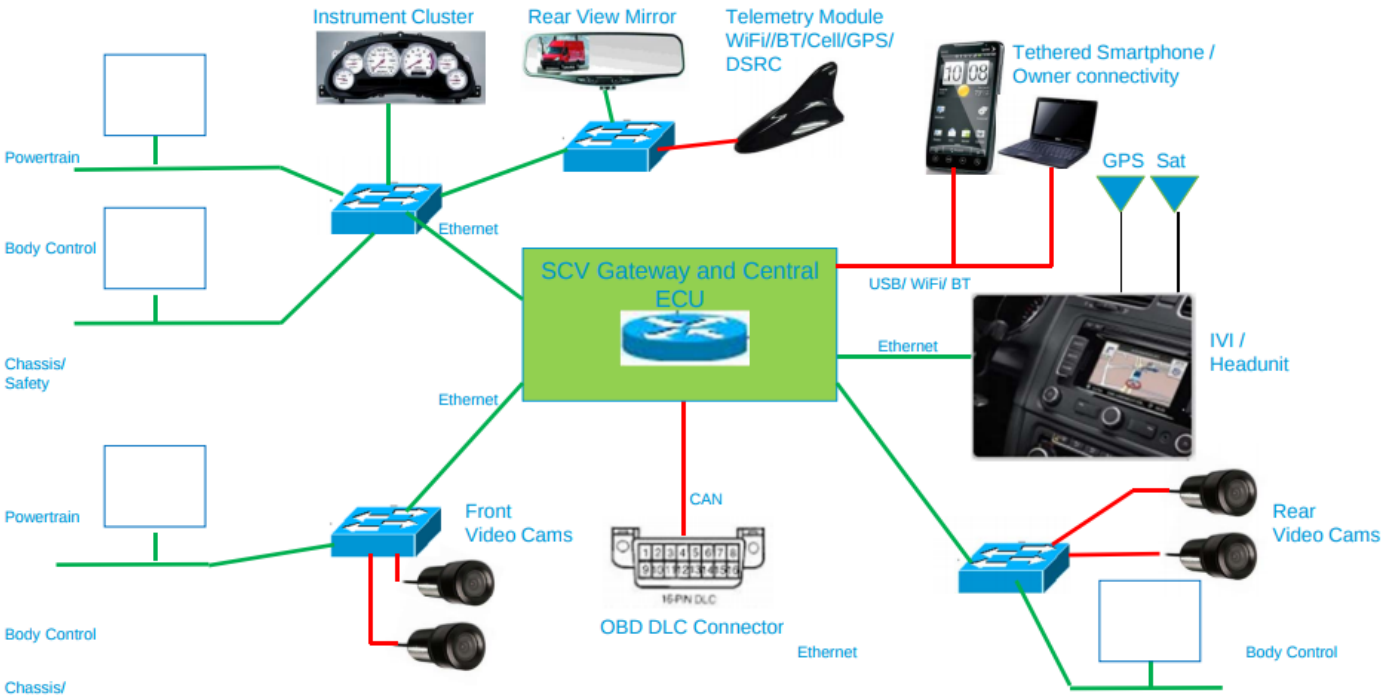


Figure 1: Visualisation of Vehicle CAN bus with ECU Consolidation[4]

Some of this compiled and filtered information is then relayed to the appropriate "Fog" nodes. In a real field test, these nodes would be represented by nearby cell towers that contain small servers or micro data warehouses and gather information from all "Edge" nodes (cars) on the road and process it further. They then send the concise data to cloud services (e.g. improving geo-location data, traffic information, accidents etc.).

- Research the topic of "Fog and Edge Computing"
  - Fully understand what it is.
  - What implications are there?
  - How it affects current and future communications infrastructures?
  - Use the attained knowledge to create the design and structure of the system accordingly.
- Configure multiple Raspberry Pis
  - Test the Automotive Grade Linux[5] distribution.
  - Set up networking to communicate them with each other.
  - Pick out sensors to attach to the boards.
- Implement a simulator of a car's CAN bus, OBD(On-board diagnostics) and ECU's
  - Research the workings of the CAN bus and how the in-vehicle components communicate.
  - Design an architecture mimicking the environment.
  - Determine how many Raspberry Pis will be needed for a good overall simulation of the active ECUs in a car.
  - Use Java and any other necessary languages/frameworks to develop a framework for simulating the separate ECUs and SCV gateway. Package it appropriately so it can be extended in the future.
  - Supplement the framework with a web-based interface to nicely visualise the processed data from the individual nodes.
  - Investigate the potential usefulness of implementing the car simulator with the Cardiff University's "Formula Student" race car project.
- Investigate Attack Surface and Privacy of the designed simulator
  - Find possible weak points.
  - Uncover and list potential attack patterns relevant to vehicles.
  - Research virtualisation possibilities of separate components in the system.
- Evaluate the advantages and limitations of using "Fog" and "Edge", distributed architecture
  - Clearly state the benefits and drawbacks of such an approach referring to the results of the simulation.
  - Show the implications and potential opportunities this approach gives rise to.

# 3 Work Plan

On a weekly basis, short meetings will be scheduled with my supervisor for guidance and tracking of the progress on the project. Depending on the availability of the supervisor and necessity or urgency, meetings may be altered to fortnightly ones. Each week I will be working on a set of tasks which lead up to milestones and I have set the appropriate deliverables. During progression with the project I will be taking notes and compiling them into a report. Using the following weekly plan, my aim is to keep track of said tasks, milestones, deliverables and special review meetings:

**Week 1**: Jan 29 - Feb 4

- Investigate existing Fog & Edge computing solutions and how they affect current and future communications infrastructures
- Complete Initial Plan
- Meet with supervisor to discuss Initial Plan
- Amend Initial Plan if necessary

**Week 2**: Feb 5 - Feb 11

- Submit initial Plan
- Research existing in-vehicle, connected car communications solutions
- Background research on Fog & Edge computing hardware solutions
- Background research on Fog & Edge computing simulators (iFogSim)

**Week 3**: Feb 12 - Feb 18

- Background research on Fog & Edge computing infrastructure
- Begin designing the architecture for the proposed system
- **Milestone**: Finalize requirements for the simulator, e.g. sensors to use, Raspberry Pi count, processing to be done on central node, potential virtualisation of components

**Week 4**: Feb 19 - Feb 25

- Setup Raspberry Pis with Automotive Grade Linux distribution
- Set up networking to communicate them with each other
- Pick out sensors to attach to the boards and make sensor probes
- Take pictures of the devised system

**Week 5**: Feb 26 - Mar 4

- Create a GitHub repository to store the developed scripts and framework
- Create first draft of the simulator framework
- Collect sensory data from the ECUs

**Week 6**: Mar 5 - Mar 11

- Test and record the network throughput and efficiency of this first revision
- Store compiled sensory information locally into logs/NoSql database
- Improve framework as necessary
- Update repository

**Week 7**: Mar 12 - Mar 18

- Improve framework as necessary
- Test and record new iteration throughput and efficiency
- Update repository

**Week 8**: Mar 19 - Mar 25

- Improve framework where necessary
- **Milestone**: Finalise simulator, which by this point should be mostly complete
- Record results from testing the throughput and efficiency
- **Review Meeting** with supervisor to check whether simulator is sufficient

**Week 9**: Mar 26 - Apr 1

- Develop a web-based interface which extends the framework
- Collect data from sensors to display on the interface

**Week 10**: Apr 2 - Apr 8

- Compare created solution to existing vehicle systems and locate potential security vulnerabilities and state where it has been improved
- Investigate attack surface of possible existing in-vehicle communications solutions and devise attack methods
- Investigate the potential usefulness of implementing the car simulator with the Cardiff University's "Formula Student" car

**Week 11**: Apr 9 - Apr 15

- Use the notes and information taken from the design, development and testing to write the report
- Whilst writing the report, reflect back on the development process and find where improvements can be made and adjust if necessary

**Week 12**: Apr 16 - Apr 22

- Continue writing the report evaluating the product as progressing
- **Review Meeting** with supervisor to go over the overall report and simulator

**Week 13**: Apr 23 - Apr 29

- Continue tweaking and improving the report
- Based on the supervisor feedback, adjust the report or the simulator if needed

**Week 14**: Apr 30 - May 6

- **Milestone**: Adjust and add finishing touches and make a complete review of the report

**Week 15**: May 7 - May 11

- **Deliverable**: Submit the final report and program source code

# References


[1] *'Automobilia LA' - Brian Krzanich [ONLINE]*
*Available at: https://automobilityla.com/speaker/brian-krzanich/*
[*Accessed 22 Jan 18*].

[2] *'CAN bus' - Wikipedia, the Free Encyclopedia [ONLINE]*
*Available at: https://en.wikipedia.org/wiki/CAN_bus*
[*Accessed 23 Jan 18*].

[3] *'Electronic control unit' - Wikipedia, the Free Encyclopedia [ONLINE]*
*Available at:*
*https://en.wikipedia.org/wiki/Electronic_control_unit*
[*Accessed 23 Jan 18*].

[4] *'The Smart and Connected Vehicle and the Internet of Things ' - Flavio Bonomi [ONLINE]*
*Available at:*
*https://tf.nist.gov/seminars/WSTS/PDFs/1-0_Cisco_FBonomi_ConnectedVehicles.pdf*
[*Accessed 21 Jan 18*].

[5] *'Automotive Grade Linux' - The Linux Foundation - Automotive Grade Linux [ONLINE]*
*Available at: https://www.automotivelinux.org/about*
[*Accessed 24 Jan 18*].