RENDITS

MANUAL

Design and layout inspired by the work of Edward Tufte, and typeset using the tufte-latex class.

Contents

Introduction

This document explains how to get started working with the Rendits V2X router. We first give a brief overview where we explain the central concepts, and how the router fits into a larger system. Next, we will explain how to get started compiling the router software in, and how to run a first test of the router. We then explain how to configure the router.

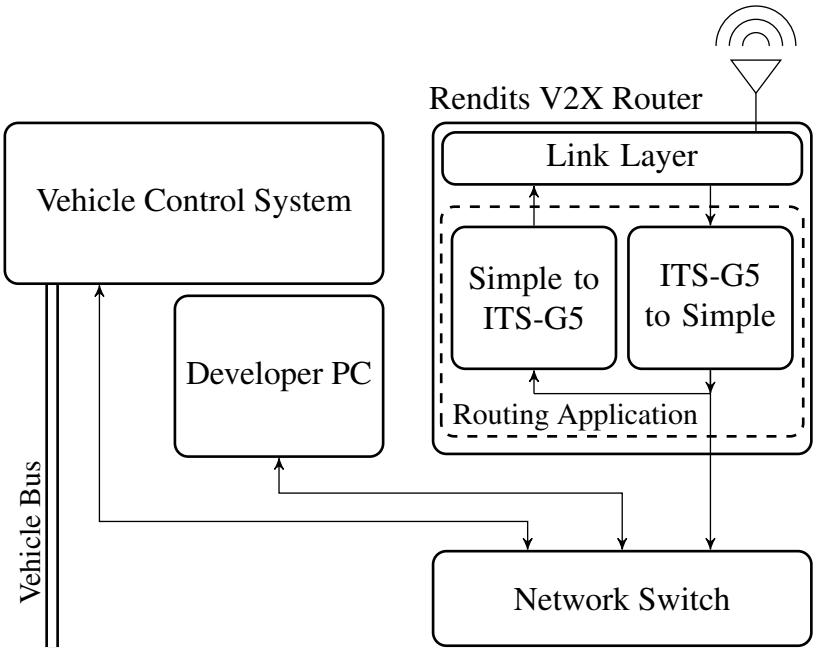

Figure 1: System overview

The Rendits vehicle-to-anything (V2X) router is an embedded Linux computer developed to make adding V2X communication to any project easy. Specifically, it is a collection of open source software components that facilitates wireless communication compliant with the ITS-G5 standard issued by the European Telecommunications Standards Institute (ETSI) [1]. The router is meant to empower its users by making it easy to modify and extend its functionality. For example, it is easy to use a modified version of ITS-G5 and to support custom messages.

The router is developed with the system architecture shown in Fig. 1 in mind. In particular, we assume that there is a local network connecting an arbitrary number of devices, with each device

[1] ETSI. ITS-G5 Standards Documents. http://www.etsi.org/technologies-clusters/technologies/automotive-intelligent-transport, April 2017

running part of an intelligent transportation system (ITS) application. We refer to the device connected to the vehicle as the "vehicle control system", and the computer that the developer is working from as the "developer PC". Note that the vehicle control system is not needed to get started using the router and that it is also possible to run arbitrary applications on the router itself. The router is added to the network using a wired network connection and communicates with other devices on the network via UDP messages. We refer to these messages as local messages and to the wireless messages sent/received by the router as wireless messages.

A key feature of the Rendits router is its simplified message set that allows devices to send and receive ITS-G5 messages without having to deal with its complexities. The simple message set specification is given in the Simple Message Set Specification chapter. The application responsible for converting incoming ITS-G5 wireless messages to their corresponding simple message and forwarding these to a device on the network is referred to as the "routing application". This application also listens for incoming local messages that it converts to ITS-G5 messages and transmits. The link layer is the interface between the routing application and the physical layer, i.e., the network card and antennae. The routing application and link layer communicate via UDP messages. It is thus possible to run the routing application on another device than the router, though that is not covered here.

The router hardware platform is the PC Engines apu2c4 x86 single board computer [2]. Hardware documentation can be found at [3]. The network card included is based on the AR9280 chipset [4].

[2] https://www.pcengines.ch/apu2c4.htm
[3] https://www.pcengines.ch/pdf/apu2.pdf
[4] https://www.pcengines.ch/wle200nx.htm

Getting Started

In this section, we explain how to setup the development environment, compile the necessary software, and make a first test of the router. After this section you will be able to

1. compile the Geonetworking library.
2. compile the Rendits routing application and test suite.
3. monitor incoming and outgoing traffic on the router via `ssh`.
4. send ITS-G5 messages from your PC using the Rendits test suite.

Steps 1. and 2. can be carried out without connecting to the router. For steps 3. and 4., you will need the included document giving the router settings. The explanations are given for the terminal. Each step can also be completed using an IDE, such as Eclipse, but this is not explained here. We assume that the router settings are the factory defaults (see the Configuration chapter).

Build Environment

First, we need to install the Java Development Kit (JDK) and Java Runtime Environment (JRE) [5]. We also need the Apache Maven software project management tool [6]. Maven is used for compiling and installing the necessary software. Maven also automatically downloads and manages dependencies. This guide is written with Java version 8 and Maven version 3 in mind.

After installing the JDK, JRE, and Maven, you may need to add their respective `bin` subfolders to the system path [7]. You also need to set the `JAVA_HOME` environment variable to the folder that the JDK is installed to [8]. Any open terminals need to be closed and re-opened for the changes to take effect.

Enter the commands below in a terminal to verify that Java and Maven are installed and working correctly. Make sure that the Java version printed by `mvn` matches that of the JDK. Command not found errors indicate that there is a problem with the system path.

```
java -version # JRE version
javac -version # JDK version
mvn -version # Maven version
```

[5] Java installation instructions: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

[6] Maven installation instructions: `https://maven.apache.org/install.html`

[7] Instructions for adding folders to the system path: `https://www.java.com/en/download/help/path.xml`

[8] The procedure is similar to that of adding folders to the system path. Create a new `JAVA_HOME` variable, the value of which is the folder that the JDK is installed to.

Geonetworking Library

The routing application is built on top of the Geonetworking library [9] by Alexey Voronov. This library contains the features needed to support ITS-G5. To compile and install it,

[9] Alexey Voronov, Jan De Jongh, Dennis Heuven, and Albin Severinson. Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM / ASN.1 PER / BTP / GeoNetworking, June 2016. URL https://doi.org/10.5281/zenodo.55650

1. Either make a copy of the `geonetworking` folder on the included USB stick, or download the latest version from https://github.com/rendits/geonetworking.
2. Open a terminal window in the resulting folder and enter

```
mvn clean install
```

Maven will automatically download the required dependencies, compile the library, test, and install it. You may need to enter the command several times before it succeeds in case some dependency fails to download. After the command has finished you should see the output below. Any other result may indicate a problem with Java or Maven.

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Parent for GeoNetworking ........................... SUCCESS [  0.503 s]
[INFO] GeoNetworking ...................................... SUCCESS [  5.496 s]
[INFO] Datatypes for ASN.1 ................................ SUCCESS [  0.290 s]
[INFO] UPER encoder for ASN.1 ............................. SUCCESS [  2.036 s]
[INFO] CAM and DENM message spec .......................... SUCCESS [  2.646 s]
[INFO] Upper Tester for GeoNetworking ..................... SUCCESS [  2.363 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Routing Application and Test Suite

The routing application is responsible for converting and forwarding messages. This application is already installed and running on the router. The test suite is an application used to verify that the router works properly, and for measuring message round trip time. These applications are part of the same project and are installed together. After installing, we will run the test suite against an instance of the routing application running on your PC. Next, we will run the test suite against the actual router. Finally, we will use two routers to test wireless transmission and measure the round trip time. To compile and install the routing application and test suite,

1. Either make a copy of the `rendits-router` folder on the included USB stick or, download the latest version from https://github.com/rendits/router.
2. Open a terminal window in the resulting folder and enter

```
mvn clean install
```

Note that you need to have installed the Geonetworking library in the previous step. You should see

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Rendits router suite ............................... SUCCESS [  0.376 s]
[INFO] router ............................................. SUCCESS [  5.399 s]
[INFO] testsuite .......................................... SUCCESS [  0.557 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

We will now run the test suite against an instance of the routing application running on your PC. In the next section we will run the same test against the router. To start the routing application, enter the following commands in the terminal

```
cd router
java -jar target/rendits-router.jar
```

You should see

```
12:43:24.337 [main] INFO  net.gcdc.geonetworking.GeonetStation - Initialized station with GN address
   1652898200575888977 and MAC address MacAddress[0]
12:43:24.340 [main] INFO  net.gcdc.geonetworking.GeonetStation - Starting BEACON service
12:43:24.342 [pool-1-thread-2] INFO  com.rendits.router.Router - Send thread starting...
12:43:24.342 [pool-1-thread-1] INFO  com.rendits.router.Router - Receive thread starting...
12:43:24.342 [pool-1-thread-3] INFO  com.rendits.router.Router - Send thread starting...
12:43:24.342 [pool-1-thread-4] INFO  com.rendits.router.Router - Send thread starting...
#### Rendits Vehicle Router ####
Listening on port 5000
Vehicle Control System IP is /127.0.0.1
Sending incoming CAM to port 5001
Sending incoming DENM to port 5001
Sending incoming iCLCM to port 5001
Sending incoming custom messages to port 5001
Copyright: Rendits (albin@rendits.com)
12:43:26.344 [pool-1-thread-5] INFO  com.rendits.router.Router - #CAM (Tx/Rx): 0/0 | #DENM (Tx/Rx): 0/0 | #iCLCM
    (Tx/Rx): 0/0 | #Custom (Tx/Rx): 0/0
12:43:27.345 [pool-1-thread-5] INFO  com.rendits.router.Router - #CAM (Tx/Rx): 0/0 | #DENM (Tx/Rx): 0/0 | #iCLCM
    (Tx/Rx): 0/0 | #Custom (Tx/Rx): 0/0
12:43:27.455 [pool-2-thread-1] INFO  net.gcdc.geonetworking.GeonetStation - Sending beacon
```

The printout tells you that the routing application

- listens for local messages on port `5000`.
- converts and forwards incoming ITS-G5 wireless messages to port `5001` on the IP address `127.0.0.1` (localhost).

We explain how to change these settings in the Configuration chapter. The application periodically prints the number of outgoing/incoming (TX/RX) messages. These counters will remain at zero since there are no messages. Leave the routing application running and open a new terminal in the `rendits-router` folder. In this terminal, enter

```
cd test-suite
java -jar target/rendits-test-suite.jar
```

You should see

```
13:04:10.151 [main] INFO  com.rendits.testsuite.TestSuite - Starting test suite...
13:04:10.158 [main] INFO  com.rendits.testsuite.CamService - Starting CAM service...
13:04:10.160 [main] INFO  com.rendits.testsuite.TestSuite - Starting latency test warm-up.
```

In the terminal window with the routing application running you should see the CAM TX counter increase. This means that the router application is receiving local messages from the test suite. The router is converting these to proper ITS-G5 messages. However, since there is no link layer running these messages are discarded. Note that all other counters will remain zero since the test suite only generates simple CAM messages. The test suite latency test warm-up waits until it has received a certain number of local messages. Since there are no incoming messages it will never finish. You can exit the routing application and test suite by pressing `Ctrl-c`.

PC-Router Connection

In this section we will finally connect to the router.

1. Connect your PC and the Rendits router to a network switch. Make sure to connect the router using the port marked as having a static IP address in the included documentation.

2. The router will be configured to forward incoming wireless ITS-G5 messages to an IP address and port given in the included documentation. Assign this IP address to your PC.

3. Connect to the router via `ssh` with username `root` and password `rendits`. The Linux and OS X terminal command is `ssh root@<router-ip-address>`. We recommend Putty [10] for Windows.

[10] https://www.putty.org/

4. In the `ssh` terminal, enter

```
tail -f /var/log/rendits/router.log
```

This will print the routing application log messages.

```
22:38:11.053 [pool-1-thread-5] INFO com.rendits.router.Router - #CAM (Tx/Rx): 0/0 | #DENM (Tx/Rx): 0/0 | #iCLCM
      (Tx/Rx): 0/0 | #Custom (Tx/Rx): 0/0
```

5. Next, we are going to generate local messages using the test suite running on your PC. Open the test suite properties file [11] and change the `routerAddress` property to the IP address of the router.

6. In a separate terminal window, run the test suite as you did previously. You should see the `CAM TX` counter increase in the `ssh` terminal window. This means that the router is transmitting wireless ITS-G5 messages. Leave all terminal windows open and applications running for the next section, where we will use a second router to receive these messages and forward them back to the test suite.

[11] `rendits-router/test-suite/testsuite.properties`

Round Trip Time

In the final section of the getting started guide will use two routers to measure the message round trip time. After this section you will have a fully working ITS-G5 setup.

1. Connect the second router to the same network switch and power it on.

2. The second router will automatically start receiving the wireless messages transmitted by the first router. These will be forwarded to the test suite. You should see output similar to below. Note that Java applications get faster after running for a while due to how its just-in-time compilation works [12]. Because of this the round trip time may be lower on a second run.

[12] For more details, see https://www.beyondjava.net/blog/a-close-look-at-javas-jit-dont-waste-your-time-on-local-optimizations/

```
15:55:17.701 [main] INFO  com.rendits.testsuite.TestSuite - Starting latency test...
15:55:17.713 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 0.0% completed...
15:55:18.715 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 10.0% completed...
15:55:19.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 20.0% completed...
15:55:20.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 30.0% completed...
15:55:21.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 40.0% completed...
15:55:22.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 50.0% completed...
15:55:23.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 60.0% completed...
15:55:24.710 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 70.0% completed...
15:55:25.712 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 80.0% completed...
15:55:26.711 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - 90.0% completed...
15:55:27.701 [pool-1-thread-1] INFO  com.rendits.testsuite.TestSuite - Delay test completed. Received: 1000
     messages. Average delay: 4.736813ms.
```

You now have a fully working ITS-G5 setup. Congratulations!

Configuration

In this chapter, we explain how to configure the router. To change any settings you first need to connect to the router over `ssh` or serial console. The default username is `root` and the default password `rendits`. We recommend using Putty on Windows. The terminal command on Linux and OS X is

```
ssh root@<router-ip>
```

You can also connect using a serial console. See [13] for details.

[13] https://www.pcengines.ch/pdf/apu2.pdf

Routing Application Settings

After connecting to the router, open the configuration file `/opt/rendits/router.properties` by typing

```
nano /opt/rendits/router.properties
```

Make any changes you need, save the file with `Ctrl-o`, and close it with `Ctrl-x`. The contents of this file are similar to

```
## Rendits vehicle router properties
## Please make a backup before modifying

# Country code used by the router
countryCode=46

# IP or hostname of the vehicle control system. Incoming wireless messages are
# forwarded to this address.
vehicleAddress=192.168.1.10

# This is the port that the application listens for local messages from the
# vehicle control system on.
portRcvFromVehicle=5000

# Ports that incoming CAM/DENM/iCLCM/CUSTOM wireless messages are forwarded to.
# These can be the same or different ports.
portSendCam=5001
portSendDenm=5001
portSendIclcm=5001
portSendCustom=5001
\clearpage

# Port to receive packets from the link layer on. Only change this if you are
# using a different link layer than the default.
localPortForUdpLinkLayer=4000

```

```
# Remote port and address of the link layer. Only change this if you are using a
# different link layer than the default.
remoteAddressForUdpLinkLayer=127.0.0.1:4001

# MAC address to be used by the Geonetworking stack
macAddress=00:00:00:00:00:00

# Number of receive threads. You can experiment with this parameter to find the
# optimal setting for your application.
sendThreads=3
```

The routing application needs to be restarted for the changes to take affect.

```
/etc/init.d/rendits-router restart
```

Wired Network Settings

In this section, we explain how to change the wired network settings. These are all configured by editing the its corresponding file and restarting the router. The file names are

`/etc/network/interfaces.d/<interface>.conf`.

For example, to change the `eth0` settings you need to edit `/etc/network/interfaces.d/eth0.conf`.

Note that the horizontal space in these files is a single tab. `eth0` is configured for DHCP by default and its configuration file has contents

```
auto eth0
iface eth0 inet dhcp
```

To change the `eth1` or `eth2` settings, edit `eth1.conf` or `eth2.conf`, respectively. `eth2` is unused by default. `eth1` is configured with a static IP address by default and its configuration file has contents

```
auto eth1
iface eth1 inet static
    address 192.168.0.10
    netmask 255.255.255.0
    broadcast 192.168.0.255
```

Wireless Network Settings

In this section, we explain how to change the wireless network settings. The procedure is the same as for the wired settings. There is a single wireless interface and its configuration file is `/etc/network/interfaces.d/wlan0.conf`. We first explain how to change the settings manually. In a terminal, enter

```
# set the regulatory domain
iw reg set SE
# bring down the wireless interface
ip link set wlan0 down
# set device in OCB (outside the context of a basic service set) mode
iw dev wlan0 set type ocb
# bring up the wireless interface
ip link set wlan0 up
# set the frequency to 5900MHZ and the channel width to 10MHZ
iw dev wlan0 ocb join 5900 10MHZ
```

To make the changes persist after reboot, edit `/etc/network/interfaces.d/wlan0.conf`. Address, netmask, and broadcast have to be set even though these are not used to avoid problems with the driver. Note that the horizontal indentation is a single tab. For example,

```
# OCB Entry
auto wlan0
iface wlan0 inet static
    address 192.168.10.10
    netmask 255.255.255.0
    broadcast 192.168.10.255
    pre-up iw reg set SE
    pre-up iw dev wlan0 set type ocb
    post-up iw dev wlan0 ocb join 5900 10MHZ

```

Routing Application on a Separate Computer

It is possible to run the routing application a separate computer than the router hardware. This could,for example, be useful during development since the application does not need to be moved to the router after compiling. The routing application communicates with the link later via UDP messages (see Fig. 1). Changing the link layer settings is sufficient to have it forward incoming messages to any computer on the network. In an `ssh` terminal, enter

```
nano /opt/udp2eth/run.sh
```

The default contents of this file is

```
device=wlan0
server=0.0.0.0:4001
client=127.0.0.1:4000
/opt/udp2eth/udp2eth --device=$device --server=$server --client=$client
```

Change the client IP address and port to that of the computer you want incoming wireless messages to be forwarded to. Note that these are raw messages, i.e., they are not converted to simple messages before forwarding. Restart the link layer for the changes to take effect.

```
/etc/init.d/udp2eth restart
```

Logging

The routing application uses LOGBack classic for logging. To change how logging is performed, you need to edit `rendits-router/router/src/main/resources/logback.xml` [14]. For example,

[14] File format is explained at https://logback.qos.ch/manual/appenders.html

```
<root level="info"> <!-- set to DEBUG to also log debug messages -->
  <appender-ref ref="FILE" /> <!--change FILE to STDOUT to log to standard output -->
</root>
```

After making any changes, recompile the application, and move the resulting `.jar` file to the router.

Simple Message Set Specification

In this section, we specify the simple message set. The simple messages correspond to the proper DENM, CAM, and iCLCM messages, respectively. DENM and CAM are specified by ETSI [15]. iCLCM is specified in [16]. The values of all data elements should be provided as in the ETSI specification unless stated otherwise. The only data elements that do not follow the ETSI specification is the DetectionTime and ReferenceTime elements part of the DENM message.

Containers and data elements marked with (Opt.) are optional and their presence is indicated using bit masks. For example, if the ContainerMask in a simple DENM has a value given in binary by `1010 0000` (= 160 in base ten), the Situation Container and Alacarte Container, but not the Location Container is present in the message. Note that the message is of fixed length and that the bytes used for storing the Location Container are still part of the message, but that the value of those bytes is undefined in this case. Bit masks are used in the same manner for data elements within containers.

Data elements part of the ETSI specification but not included in the simple message set are marked as consisting of 0 bytes.

In addition to the CAM, DENM, and iCLCM messages, there is a custom message type. These messages is any message whose first byte has value 10 in base ten. These messages are forwarded by the routing application without any processing, i.e., the sender/receiver has complete responsibility for generating and parsing these messages. For example, a custom message could consist of a byte with value 10 followed by 5 64-bit floats for a total message size of 41 byte.

[15] ETSI. ITS-G5 Standards Documents. http://www.etsi.org/technologies-clusters/technologies/automotive-intelligent-transport, April 2017

[16] GCDC. Proposal for extended message set. http://www.gcdc.net/images/doc/D3.2.Proposal.for.extended.message.set.for.supervised.automated.driving.pdf, June 2016

Simple DENM

Container:	Bytes:	Data Element:
Header	1	MessageID (=1)
	4	StationID
	4	GenerationDeltaTime
Container Mask	1	ContainerMask
Management	1	ManagementMask
Container	4	DetectionTime [1]
	4	ReferenceTime [1]
	4	(Opt.) Termination
	4	Latitude
	4	Longitude
	4	SemiMajorConfidence
	4	SemiMinorConfidence
	4	SemiMajorOrientation
	4	Altitude
	4	(Opt.) RelevanceDistance
	4	(Opt.) RelevanceTrafficDirection
	4	(Opt.) ValidityDuration
	4	(Opt.) TransmissionIntervall
	4	StationType
(Opt.) Situation	1	SituationMask
Container	4	InformationQuality
	4	CauseCode
	4	SubCauseCode
	4	(Opt.) LinkedCauseCode
	4	(Opt.) LinkedSubCauseCode
	0	(Opt.) EventHistory
(Opt.) Location	0	LocationMask
Container	0	(Opt.) EventSpeed
	0	(Opt.) EventPositionheading
	0	Traces
	0	(Opt.) RoadType
(Opt.) Alacarte	1	AlacarteMask
Container	4	(Opt.) LanePosition
	0	(Opt.) ImpactReducationContainer
	4	(Opt.) ExternalTemperature
	0	(Opt.) RoadWorksContainerExtended
	4	(Opt.) PositioningSolution
	0	(Opt.) StationaryVehicleContainer

Simple CAM

Container:	Bytes:	Data Element:
Header	1	MessageID (=2)
	4	StationID
	4	GenerationDeltaTime
Container Mask	1	ContainerMask
Basic Container	4	StationType
	4	Latitude
	4	Longitude
	4	SemiMajorConfidence
	4	SemiMinorConfidence
	4	SemiMajorOrientation
	4	Altitude
High-Frequency Container	4	Heading
	4	HeadingConfidence
	4	Speed
	4	SpeedConfidence
	4	VehicleLength
	4	VehicleWidth
	4	LongAcceleration
	4	LongAccelerationConfidence
	4	YawRate
	4	YawRateConfidence
(Opt.) Low-Frequency Container	4	VehicleRole

Simple iCLCM

Container:	Bytes:	Data Element:
Header	1	MessageID (=10)
	4	StationID
Container Mask	1	ContainerMask
High-Frequency Container	4	RearAxleLocation
	4	ControllerType
	4	ResponseTimeConstant
	4	ResponseTimeDelay
	4	TargetLongitudinalAcceleration
	4	TimeHeadway
	4	CruiseSpeed
(Opt.) Low-Frequency Container	1	LowFrequencyMask
	4	(Opt.) ParticipantsReady
	4	(Opt.) StartPlatoon
	4	(Opt.) EndOfScenario
MIO	4	MIO ID
	4	MIORange
	4	MIOBearing
	4	MIORangeRate
Lane	4	Lane
Pair ID	4	ForwardID
	4	BackwardID
	4	AcknowledgementFlag
Merge	4	Merge request
	4	SafeToMerge
	4	Flag
	4	FlagTail
	4	FlagHead
Intersection	4	PlatoonID
	4	DistanceTravelledInCZ
	4	Intention
	4	Counter

[1] This data element should be provided in increments of 65536 milliseconds since 2004. Milliseconds since 2004 is is defined as in the ETSI specification.

Bibliography

ETSI. ITS-G5 Standards Documents. http://www.etsi.org/technologies-clusters/technologies/automotive-intelligent-transport, April 2017.

GCDC. Proposal for extended message set. http://www.gcdc.net/images/doc/D3.2.Proposal.for.extended.message.set.for.supervised.automated.driving.pdf, June 2016.

Alexey Voronov, Jan De Jongh, Dennis Heuven, and Albin Severinson. Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM / ASN.1 PER / BTP / GeoNetworking, June 2016. URL https://doi.org/10.5281/zenodo.55650.