1. Introduction

Cooperative Intelligent Transportation System (C-ITS) have the potential to shape the mobility experience by creating the applications that provide the means for safer, more efficient and confortable journeys.

Although there is a broad range of applications, a common framework may be used to support them since they share some basic needs. This means that, instead of developing each application from scratch, each one of them must be supported on a middleware that contains the shared services.

So, before starting developing your own application, you are going to design and implement one of these shared services - the Basic *service* - that is used to support cooperation among entities.

2. Service specification

The *Basic service* is used to maintain updated information the node's 1-hop neighbors. This information is stored in a *Neighbor table*, which contains one entry per valid neighbor.

In order to keep the information of the *Neighbor table* up-to-date, each node periodically disseminates a message to its 1-hop neighbor with the node's status. The status information must comprise:

- Node ID: a 16 bit integer that is an unique identifier of the node in the C-ITS system
- Position: GPS coordinates of the node
- Time: time at which the position was retrieved from the GPS system
- Message ID: unique identifier of the message generated by the node

Upon reception of a message from a node, the receiving node updates the *Neighbor table* entry of the sender node with the sender status information.

The entries of the *Neighbor table* are refreshed by soft state. This means that, whenever a message from a neighbor is received, a timer is started. If a new message from the same node is received, the timer is cancelled, the information of the table updated and the timer restarted. If no message is received, the timer expired and the entry is removed.

3. Service validation

Include in the *Basic service* design a test mode operation that allows you to debug how your system is work. If test mode is activated, both sender and receiver must print information about all relevant events.

Whenever a message is sent the sender must print its contents.
Whenever a message is received or a timeout occurs, the receiver must print the event (content of received message or timeout), plus an updated version of the *Neighbour* table.

4. Service delivery

The CA service must be submitted to the Fenix system, in a .zip file, at the end of week 6 (6 of April). Be ready to demonstrate it to the instructor and other teams during the lab class of this week (5 of April). Your service will be used to support your application. So, design, implement and test it carefully.

5. Project tips

You have to implement your service over the available Internet protocol stack. This is not the most adequate option, but is the one that we have available for now.

When deciding the Internet protocol that you are going to use, you can select either IPv6 or IPv4. For your application scenario and test-bed this option has no impact, but in a real use case only IPv6 will work properly and is compliant with ITS protocol stack. Nevertheless, you can still use IPv4 if you feel more confortable with it.

Be careful when selecting the type of transport service and protocol that you are going to use in your service. There is only one right choice!

There is code available in the Internet that might help you to develop your service. We should use it, unless another team of this course produced it. Do not forget to reference the source. Otherwise, you may get in troubles.

You will receive the raspberry PI equipment soon, but you can start the implementation and validation of your service using the team's laptop.

Concerning positioning information, there are plenty of GPS traces available in the Internet. If you don't know where to find them, have a look at Crawdad project (https://crawdad.org) and if you find some interesting trace register into the website to get it. Later, you will have access to a GPS device that you can connect to a Raspberry PI to read data from a real receiver. But, you have to do it outdoor.