University of Colorado
Network Systems
CSCI 5273

Milestone Report

Vehicle to Vehicle Broadcast Protocol

Approvals

	Name	Affiliation	Approved	Date
Customer	Jose Santos	CU Boulder		

Project Customers

Name: Prof. Jose Santos
Address: University of Colorado Boulder
Phone: +1 (303) 735 0102
Email: Jose.Santos@colorado.edu

Team Members

Name: Isha Burange Phone: (720) 447 1511 Email: isbu5273@colorado.edu	Name: Bhoomika Singla Phone: (720) 313 4957 Email: bhsi9770@colorado.edu
Name: Kiran Jojare Phone: (720) 645 6212 Email: kijo7257@colorado.edu	Name: Kevin Jones Phone: (503) 867 7438 Email: kejo7187@colorado.edu

1. Project Status

1.1. Past Planning:

Table below shows the weekly breakdown of what happened in the month of October including analysis and understanding the project definitions.

Week (October)	Status
Week 1 & Week 2	Read papers regarding V2V protocols for vehicle communication
Week 3	Designed the overall flow and how different interfaces communicate with each other
Week 4	Adding more details into the flow diagram(figured out how the overall flow would look like and the important messages that need to be transferred among components) and working on Networking midterm

1.2. Future plan:

Table below shows the future plans for building the basic version of software and scaling, testing the same based on data from multiple real-life scenarios.

Week (November)	Status
Week 1 & Week 2	Level 1: implementation for creating UDP connections and broadcasting data using these connections Level 2: Implementation of security constraints inside software.
Week 3 & Week 4	Simulating a test environment for executing the test scenarios and validating important test metrics like the performance and scale testing

1. Software Architecture

The project's software architecture, including functions to be implemented and data to be transmitted between functions, is shown below. As shown in the diagram, the software is divided into two sections, one of which is a while loop and the other is a software section triggered by an event.

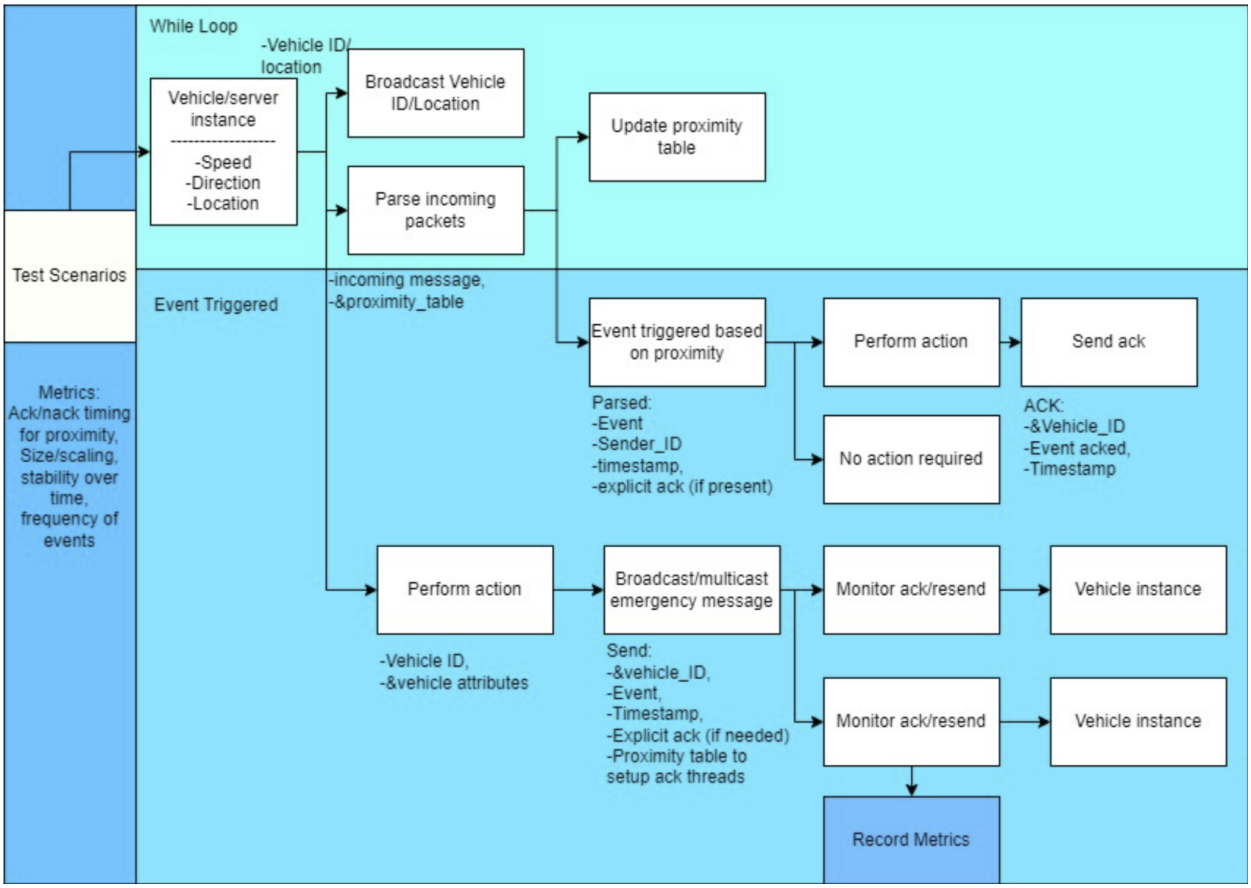

Figure 1: Software Architecture for vehicle to vehicle broadcast protocol

Function: _Load_Vehicle_Instance_

Function to load vehicle parameters into software. Most of the vehicle parameters like Vehicle Speed, Direction, Location will be loaded from user input. The inputs can be modified to different sets of values to test different scenarios.

Function: _String_Parsing_

After broadcasting the vehicle parameters in a while loop, this function loads the data received in packets from all vehicle instances.Data received from parsed packets will be stored in global data to update the proximity table.

Function: _Broadcast_Vehicle_ID/Location_
This function sends a UDP broadcast message containing vehicle ID, Location, optional: speed, and direction.

Function: _Update_Poximity_Table_
Update proximity table is called on incoming messages being defined as a vehicle broadcast. If the vehicle is within a proximity that requires an ack (say, 100m), the vehicle is added to the proximity table.

Function: _Event_Trigger_
On receiving a parsed event trigger, the action is performed (decelerate, change direction, etc), and an acknowledgement is sent back to the sending vehicle.
Note: Need to decide if the event trigger is based on the current calculated distance from the sender or a lookup in the proximity table.
The first approach seems more resilient to error, but also more hackable.

Function: _Perform_Action_
Perform Action is initiating an event. The vehicle braking etc triggers a broadcast message saying “this is who I am, and this is what I am doing.” There are then separate threads to monitor and target acks from vehicles on the proximity list, or a single thread that copies the proximity list and resends/checks off acks as they come back.

Function: _Update_Location_
For a given time/iteration, the vehicle instance updates it’s location based on velocity and direction.

Testing and metrics:
This will be tested over CLI. Ideally, we can automate bringing up vehicle instances and vary parameters/event triggers based on inputs from a test script. The vehicle instances record ack times required, and the test script may iterate time to look into outcomes and system dynamics under varying model parameters. Ideally, the switch could be modified to add in delays etc, but this is beyond the scope of this project.