**Implementing Road Train with Reliable Broadcast Algorithm in Vehicular**

**Ad-Hoc Networks (VANETs)**

**Report**

**Andrew Marshall| Evan Hall | Prashanth Tanjore Saikumar**

INTRODUCTION:

The Advancement of Mobile Ad-hoc Networks has opened avenues for vehicular communication, with vehicles acting as wireless nodes to form a mobile network. This mobile network is termed as Vehicular Ad-hoc Network (VANET) which is the base concept upon which this project is worked. VANET was developed with the intent of making transportation safe and convenient in addition to other decorative, commercial features. In forethought, Cars in the future might have the standard installation of the VANET systems, given its potential and the resulting advantages. As promising and splendid as this technology may appear, it is not lacking of complications. This project illustrates these challenges and issues in the course of the implementation of a Road Train that demonstrates the VANET technology.

ISSUES:

The issues relating to VANET might include all the inherent limitations of a Mobile, infrastructure-less network. Data routing, Network congestion, range, Inefficiency in bandwidth utilization are some of the areas of concern to be addressed in this concept.

PROBLEM STATEMENT:

Now venturing into the subject of this project, the problems analyzed and faced during the “Road Train” implementation were ceaseless information updates and the ensuing overhead, Platoon formation in an over-populated train, data exchange, Redundant broadcast -------(your time to chip in)--------------

OBJECTIVE:

We aim to design and implement a reliable broadcast algorithm over a vehicular ad hoc network(VANET) that will support the implementation of a Road Train cooperative adaptive cruise control application. In pursuit of this objective, we use PCs to act as mobile nodes (a vehicle in this case) that will congregate to model a Wi-fi environment. This way, we intend to replicate a Wireless Vehicular Ad-hoc network and allow it to transfer data and control packets using a reliable broadcast protocol. This exchange of packets will enable the nodes(cars) to form a platoon and/or join the road train. The broadcast protocol finds its purpose in that it must allow for the packets from each node to be broadcast to every other node in the Road-train so as to maintain a uniform platoon configuration.

ARCHITECTURE:

Our program construct is divided into 3 threads.

* server thread
* timer thread
* main thread

The Server thread opens a UDP server in a node and listens for packets. When a new packet arrives, it is pushed on to a queue that the main thread can pop from.

The Main thread runs a main infinite loop that constantly checks the packet buffer. Whenever there is a packet in the buffer it takes it out and processes it. This includes forwarding it in accord with the designed Reliable Broadcast Algorithm, updating other nodes’ positions and using that packet to determine if we need to speed up, change lanes, slow down or join a platoon.

Timer thread waits for a small amount of time and then sets a Boolean variable to true. Our main thread will transmit a status packet to all its neighbors only if this Boolean is set to true. After a status packet has been sent, it sets the Boolean to false. This allows us to constantly process packets (in line with the main thread) while only sending a status packet at regular intervals.

DESIGN:

The following are some design details on how the program works

Simulation set-up:

When you start the program, it first checks if a Config file already exists. If the file does not exist, this program knows it is the first node to join and is therefore the truck. If the file does exist, the node is set as a car and reads in the other node information from the Config file. Once the node reads the config file, it sends out an initialization UDP packet that indicates, the other programs, of a new node that is joining. Then the joining node will wait and listen for status packets. Once it has received a status packet from every node, it can safely find a place to enter the highway.

Updating the position of every node:

Every node (car/truck) sends an “admin packet” to every other node regardless if it is in range. This packet is guaranteed to send. When a node receives one of these packets it updates its Data Structure that keeps track of every node’s position. This packet is flagged and is not forwarded through the RBA algorithm. It simply serves to update the simulator’s data structure.

Writing the output data:

Apart from the standard output text you see on the screen, the Truck node will write to an Output.txt file, the current locations and connections of every node. This way we can view the current locations of every car at different times throughout the simulation by typing “cat Output.txt” in the terminal.

Implementation Issues:

We had to address a lot of issues with the file writing. Having more than one program reading and writing to a file causes all kinds of complications. The issues were not very critical when dealing with a computer with one core. But when we tried it on a multi-core system, the file read/write operations deteriorated under severe complexities.

To resolve this problem, we declared that only the truck writes to the output file. The cars only write to the config.txt file when they are first entering the simulation. So the chance of two cars writing at the same time is impossible as long as two car programs are not started at exactly the same time.

Another assumption was made in which the cars would only read the config file when they first enter the simulation rather than having them read it at all times. The information that they extract from the config file is other node’s hostnames and port numbers.

ALGORITHM:

Reliable Broadcast Algorithm:

Every node broadcasts its status information along with its knowledge on other existing nodes in its vicinity to every other node in the network. Prior to every transmission, a node evaluates the probability of the impending transmission, towards a particular node, to be successful. Based on this evaluation performed with the inclusion of the distance and other proximity based information, the node determines if the packet to be generated is of type ‘status’ or ‘admin’. To impart reliability there is a rebroadcast feature and this depends on the probability which is in turn based on the number of times the corresponding packet has been rebroadcasted.

Road Train Algorithm:

In order for a car to join the road train it must first place its request with the lead truck. Upon receiving this request, the truck checks to see if it is in the middle of an ongoing link establishment with another car. If this case is true, the truck cannot progress further with this requisition but decline with a packet that asks the car to try again. Otherwise the truck would respond with information about the rank that the particular car holds in the train and the link number that could be used to determine the distance and the speed that the car must maintain. Upon receiving this information the car would respond with an ALL CLEAR packet that would consummate the entire process. Now the truck might open up for join requests from other cars.

From the perspective of a non-truck node, the algorithm is as follows. When the car is in the right lane, it generates a request to join the train and sends it to the truck(Platoon Leader) as mentioned in the previous section and upon receiving the information from the truck it adjusts its speed to match up with the platoon configuration. If its request was declined, the car maintains a reasonable speed and keeps sending request packets until it is served. Also when in the same lane, if the car senses its preceding node to be less than 20 meters away and that there is a steady deceleration which is indicative of its resignation from the platoon, this car checks if there is anything within 20 meters in the left lane and if clear it moves on to the left lane else it matches up with the speed of the previous car. In another case of the car being in the left lane, the car must sense if there is any node in its closeness within 20 meters in the right lane. If the vicinity is clear, it moves on to the right lane, otherwise the car speeds up by 5 m/s while preserving the speed regulations prescribed.

Assumptions:

1. Any node that is sending us a packet is within 100m
2. The truck is always in the RIGHT LANE and in FRONT
3. Truck speed is in random range of **[20, 35]** m/s
4. Cars can start in LEFT or RIGHT LANE
5. Car starting speed is **[25, 35]** m/s
6. Connecting / disconnecting nodes is done in another layer
   1. We will never be < 100m and not connected to all nodes in that radius

**Road Rules (Not in Road Train)**

1. IF I am in the LEFT LANE, check and see if there is anything within 20 meters in the RIGHT LANE
   1. IF CLEAR, move to RIGHT LANE, resume normal speed (if different)
   2. ELSE increase speed by 5 m/s
      1. Make sure I do not go above some MAX SPEED (Starting speed + 5)
2. IF I get a packet indicating a car is less than 20 meters ahead AND is going slower than me AND IS NOT in a PLATOON…
   1. IF I am in RIGHT LANE
      1. Check if anyone is in LEFT Lane within 20m
         1. IF CLEAR, move to LEFT LANE (Case 1 takes over)
         2. ELSE MATCH the Car’s speed ahead of me
            1. NOTE: We match the speed so we stay at 20 meters behind, this should make us keep checking if the Left Lane is clear (Case 2)
   2. ELSE I am in the LEFT LANE
      1. MATCH the car’s speed ahead of me
      2. Wait until the lane is clear then case 1 should take over

**Road Train Rules - Truck**

1. [TRUCK]: I receive a REQUEST to JOIN the RoadTrain
   1. Check my variable to see if the RoadTrain is OPEN
      1. IF OPEN (No other cars are in the process of joining)
         1. Send the car the link number he is (used to calc distance)
         2. Set RoadTrain to CLOSED
         3. Wait for ALL CLEAR Packet from Car
      2. ELSE CLOSED (A car is joining the Road Train)
         1. Send the car a packet saying, “TRY AGAIN”
2. [TRUCK]: I receive an ALL CLEAR packet from a car
   1. Set RoadTrain to OPEN, the next car may now join

**Road Train Rules – Car**

1. IF I am in RIGHT LANE and one of my links is a PLATOON\_MEMBER, I send a REQUEST TO JOIN to PLATOON LEADER
   1. Wait for Response
      1. IF RoadTrain is OPEN (Received my platoon number)
         1. Increase speed until I am the correct distance away from TRUCK, then match Platoon speed
         2. Send ALL CLEAR PACKET
      2. ELSE MAINTAIN SPEED and keep sending REQUEST PACKETS until OPEN
2. IF I am in LEFT LANE and join a network with a PLATOON
   1. Slow down until I can safely move back to Right Lane
      1. Resume Speed case 1 should take over

CONCLUSION: