\*Brandon Thai Nguyen, Daniel Smith, Tyler Goffinet

05 December 2016

CPE400.1001 – Project Report Draft 1

Distributed Transport Network

(Intro – Motivation) <HOOK> With innovations in technology rapidly evolving every day, autonomous vehicles will eventually become the prominent drivers on the road; it is becoming important that vehicles are able to communicate statistical data with other vehicles. <INTRO/PROBLEM> This statistical data can include anything from speed to location and destination for applications such as analyzing traffic congestions, calculating safest routes to a destination, and determining the most optimized placement of cars on each lane of any given street all potentially maximizing flow of traffic and safety. Communication also should not be limited to cellular and satellite communication since circumstances exists where those communication methods are limited so it is wise that vehicles also have the ability to transmit data within a short range through a network of multiple vehicles in range via a peer-to-peer mobile protocol. <THESIS> (BODY 2) Our team has created a peer-to-peer protocol to effectively transfer packets of information to route packets within a network vehicles. (BODY 1)In order to test the effectiveness of this protocol we created a of a program to simulate such a network. (BODY 3) Our protocol yielded promising results

(Body – Simulator description) <WORLD DES> To start, our team needed to build a simulator program before creating any peer-to-peer protocols to be able to test them rigorously. To begin, the world class of the simulator is kept relatively simple to avoid unnecessary debugging when designing /new protocols. T+87\][\he world is constructed as a grid style city with equal distances between each adjacent intersection. The world is represented as a two dimensional array of vehicle class pointers. Every array element is looked at as an intersection and the roadways between intersections are simulated via a timer representing the travel time between intersections. Having the world represented as a two dimensional array has the advantage of having array elements that are set to NULL signifying that there is no vehicle present in that intersection. Two, the simulator is kept simple by running the world off a tick based timing system. Each vehicle operation such as moving and sending packets are set to perform their tasks in an arbitrary set of ticks. The tick based timing system was implemented to keep a level of realism; vehicles are not going to move as fast as a packet in the real world. The tick system also ensures that cycling through ticks at different rates is negated and that a user can analyze a- ny simulator activities in the middle of an operation. Three, the world handles the vehicle movement and running the vehicle’s functions within the world class. This approach allows the vehicle class to act only as a data structure storing the speed based on ticks, the destination, and the next direction the vehicles wants to go to get to the destination. This also allows the world to be in control of the vehicles itself and act as the bridge between multiple vehicle objects. The simulator is kept simple by world class use a grid style city, run off a tick based system, and manage the vehicle objects itself. <AI DES> Furthermore, the simulator needed a system to handle vehicle behavior. To begin, vehicle rules are set in place for the vehicle behaviour. First, there cannot be more than one vehicle in an intersection at a given time, second, vehicles are not allowed to move through each other, and third, they are not allowed to turn around immediately. This is done strictly for simplicity as the main focus of the project is on the routing protocols. Additionally, the vehicle behavior is controlled by an AI responsible for the vehicle’s destination, the path it will take to get to that destination, and potential collisions with other vehicles. First, the AI calculates out the destination for the vehicles in very simple manner. It is done with a random number generator that outputs a random coordinate within the range of the world. Second, the AI handles the vehicle routing getting the vehicle to its calculated destination. The vehicle will tell the world class to move itself row-wise until they are in the same column as the destination. Afterwards, the vehicle will have the world class move itself column-wise until the destination is reached. Third, the AI is responsible for handling collisions. If the path is obstructed, the vehicle will attempt to go in an optimal direction perpendicular to its current direction. If the path is further obstructed, the vehicle will attempt to wiggle its way out by trying all different directions unless there is no free direction which then the vehicle will just wait. Overall, the vehicles behavior defined by a set of rules and controlled by an AI that determines where the vehicle will go. All in all, the simulator plays a big role in the testing of protocols we create since it simulates a simplistic grid city world that is populated by moving vehicles with a behaviour system.

We created two algorithms for this project. The first was designed as a basic flooding algorithm to act as a baseline protocol. The second protocol, destination search protocol, we created as in intelligent routing protocol which uses the destination of the of the vehicle to which the packet is trying to reach.

The flood algorithm has each vehicle broadcast any packets it may contain to all vehicles in its radius. Once a vehicle successfully throws it packet to at least one vehicle (determined by a response from the target vehicle) it will not throw that packet again. After 5 cycles the packet will be discarded. This algorithm works well in high density areas but is costly as each vehicle must occupy time dealing with a packet even if that vehicle can not get it any closer to the destination.

The destination search algorithm uses the destination of the vehicle of the packet it is trying to reach as a method for routing the packet. For this algorithm each vehicle must maintain a list of where other vehicles are traveling to and from. It would be far too costly to have each vehicle send out its current location at each cycle so instead we have each vehicle output its next destination and starting location. This information is shared by transmitting an update packet with the vehicle’s new destination and from what position the vehicle is starting from. Whenever a vehicle reaches its destination it recalculates a new destination and shares that information with the rest of the world. This packet transmission is sent to all vehicles in the system via the flood algorithm. Upon receiving an update packet that vehicle will updates its list of vehicle destination and starting locations. This list will allow vehicles to find a smarter way to locate the packet target.

For a packet being transfered from Vehicle A to Vehicle B the following steps will followed. Vehicle A will first check its list of surrounding vehicles to see if Vehicle B is within range, if that is the case, Vehicle A will throw the packet to Vehicle B and get rid of that packet. If Vehicle B is not in range, then Vehicle A will look at its array of vehicle locations to see if it has any information on Vehicle B. If Vehicle B is in this list then Vehicle A will take Vehicle B's destination and compare it to Vehicle A's current location to determine which direction would be ideal to attempt to throw the packet for it to intersect with Vehicle B. Once Vehicle A has an ideal direction it cycles through all vehicles in its immediate vicinity and assigns each one a score of how ideal it would be to throw to that vehicle. The score is based off its current direction and its position in relation to the ideal direction to throw calculated earlier. Vehicle A also gives itself a score to determine if the best choice is to not throw the packet at all. After all vehicles are considered the packet is thrown to the vehicle that received the highest score. And the process repeats with the new vehicle. The results of testing this algorithm were ultimately inconclusive yet still showed promise.

For testing purposes we passed a series of packets through a world of size 10 X 10 populated with 20 vehicles. This represents a medium sized city with moderate congestion. In this scenario our flood algorithm had a one hundred percent success rate with the average number of ticks to achieve packet transfer equaling 4.3. The Destination Search algorithm had only a 50% success rate, however the average number of ticks to achieve packet transfer was 2.6. The low success rate of the destination search is due partially to limitations with our simulator and partially due to the weights used to determine the best vehicle choice. As the simulator runs longer the vehicles have a tendency to become gridlocked and not be able to move, this is a problem for our algorithm as if the vehicle holding the packet becomes trapped by vehicles whom the algorithm determines are no worth throwing to , the packet becomes trapped with the vehicle. This is a problem both with our simulator and with our algorithm.

(Body – Results) Lastly, one of the two protocols that our team designed yields good results whereas the other has a lower success rate. <Alg result 1> First, the protocol that has good results is the flooding algorithm. The flooding algorithm is able to transmit a packet from the source to the destination very reliably; there was not a case where the packet sent was lost. The high success rate can be attributed by knowing the starting and ending location of the vehicle and being able to use that data to approximate where the packet needs to go to. The only caveat of the flooding algorithm is that it is a lot more complex than the destination search algorithm as each vehicle requires to have a consistently up-to-date list of vehicle destinations which is retrieved from other vehicles when they go to a new destination via flooding. Even though the flooding algorithm is heavy, it has virtually no failure rate in our simulation and it can be argued that the routing is very efficient since the packet is routed directly towards the general location of receiving vehicle. <Alg result 2> Second, the destination search algorithm, even though a promising one, has a low success rate. There are many circumstances where the packet will float around the network of local vehicles for a long period of time without finding its intended recipient. The algorithm had high potential but perhaps due to limited time, was not never developed to an acceptable state. All and all, even though the destination search algorithm protocol was shown unsuccessful, the flooding algorithm protocol proved to have a very high success rate and very efficient packet routing.

(Conclusion) Our team was successfully able to design and implement an algorithm tested in our own simulator environment solving the problem of close range peer-to-peer communications between multiple autonomous vehicles. Hopefully future autonomous car makers can have a broad set of communication tools to allow vehicles to transmit vital information from one vehicle to another to opening up doors for various types of applications like safety, analytics, and more. Even though our team has successfully created a custom protocol that handles peer-to-peer communication between a dynamic network of moving vehicles, we would have liked to try out other algorithms already made such as VANET, CASNET, DSR, BATMAN, AODV, and others of the similar sort. We also would have even liked to test out our flooding algorithm protocol using real hardware provided by the CSE department but could not due to time constraints. Overall, our team has learned a lot about different routing algorithms and protocols as well as having exposure to out of the box thinking creating a protocol to handle a situation that may appear as autonomous vehicles become more prominent.