Simulation of Driverless Cars managed by a Software Defined Network using A* Search for Routing

Andrew Frost and Richard Millar

CPE 400

12/06/2017

Abstract:

As driverless cars come closer to being available to consumer markets, efficient methods for routing them through cities will become an important consideration. Routing a driverless car to its destination in the least amount of time is a challenging and complex problem. This work uses a Software Defined Network (SDN) that utilizes a Central Compute Node (CCN) for routing cars to their destinations. The protocol in this work is reactive and routes cars based on their request. This paper presents how A* search can be used to determine the optimal route for routing cars through a city simulation.

I. Introduction:

A Software Defined Network is dynamic at its core. This network must make decisions based on the current landscape of the network to efficiently route packets through the quickest route possible. In this paper, the use of a software defined network to route cars through a city is examined. A* search is used as the algorithm for route finding.

II. Simulation:

The simulation takes in mind only the major considerations of a city roadway system: capacity, average travel time, and the city layout. The simulation is a largely simplified representation of the considerations of routing traffic in a modern city. The simulation is designed based on two protocols: the Central Compute Node (CCN) Protocol and the Vehicle Protocol.

A. Simulation Design:

This simulation was designed around the use of the Central Compute Node (CCN) Protocol and the Vehicle Protocol running in the application layer. Any lower level networking functions, such as connections between access points, are abstracted away.

The Central Compute Node Protocol:

The CCN Protocol is the protocol used by a central compute hub responsible for directing driverless cars through the city. The CCN Protocol is reactive. Cars initiate all route requests and re-routing requests. The protocol was designed to be reactive to minimize its overhead and to optimize finding routes through a city.

The CCN Protocol stores a map of the city with estimated driving times in seconds, a table of IDs of cars that have joined the network, and a table that denotes the driverless capacity of every roadway. This information is used by the CCN's A* search to find the optimal route for each car. The A* search uses the expected cumulative travel time between two roads as its heuristic. Rather than try to predict how many cars will be on a given road in the future, the CCN assumes that the congestion of a roadway will be similar to the current congestion level while planning. In order to deliver cars with the optimal routes, the CCN has a number of messages, which are shown in Table 1.

The CCN Message Types

Message Type	Transaction
Join Network	Receive
Leave Network	Receive
Request Route	Receive
Send Route	Send
Change Road	Receive / Ack or Nack

Table 1: the CCN message types. This table shows what types of messages are used in the CCN protocol. The "/" indicates the sequence of the transaction; this separates the request and response.

As displayed in Table 1, the CCN Protocol has the following message types: the join network message, the leave network message, the request route message, the send route message, and the change road message. The join network message allows cars to join the network at the road on which they have just started. The job of this message is to notify the CCN that there is a driverless car at a certain location. The leave network message notifies the CCN that a tracked driverless car is leaving the network and will no longer be on the city's roadways. The request route message is used to make a formal request for a route to the CCN. This message includes the car's ID, the car's destination, and the car's current location. The send route message is used by the CCN to transmit route information to a vehicle. Finally, the change road message is used to request a road change by a vehicle. This message will return to the vehicle whether or not it is allowed to turn onto a roadway at the current moment in time. An Ack indicates that the car is free to turn on to the road; a Nack indicates that the car should not turn on to the road.

The Vehicle Protocol:

The Vehicle Protocol is the protocol used by each driverless car. It is used to communicate with the CCN. The protocol uses the messages of the CCN Protocol in order to get routing information. When a driverless car enters a road way it sends the join network message to the CCN. This lets the CCN know that a driverless car is going to be on the road and that it needs to be considered when routing traffic.

The driverless car sends the request route message when it wants to know where it should go. Every driverless car should send this request after joining the network. By requesting a route the car makes a request for the best way to go. Once the CCN is finished computing the optimal route, the CCN will use the send route message to distribute the route to all driverless cars that are requesting a route with the same destination and source. The CCN will only send the route to the minimum of the number of vehicles requesting the route and the capacity of any road on the route. The driverless car can proceed to go on its local known best path until the response comes in; some backtracking could be required on the part of the driverless car.

The driverless car sends the change road message to the CCN while it is making the approach to the intersection between its current road and the new road. The expectation is that the message is sent with ample time for the CCN to issue a response. The CCN can issue two response in this message: a confirmation that the driverless car can change roads or a negation warning the car not to turn onto that road. If the driverless car receives a response indicating that it should not turn off, then the Vehicle protocol directs that the car request a new route and pull off the road if necessary.

B. Simulation Assumptions:

The simulation makes several assumptions in order to simplify its implementation. The first assumption made is that the driverless cars can independently navigate through traffic and avoid local collisions. Neither protocol offers any directive in the local navigation of the driverless cars. The protocols are restricted to routing cars between different roadways.

The second assumption is that any packet loss and network connectivity issues would be handled by a lower level protocol, such as TCP. Since cars travel slow relative to a computer network any loss or delay of packets should be negligible to the performance of the CCN and Vehicle Protocols.

A third assumption being made is that the CCN will be massively parallelized and will have the computational resources to compute an optimal route before a car has traveled the distance of an entire road.

The fourth assumption is that the capacity of each road way is exclusive of non-driverless cars and that non-driverless cars do not occupy space on a road that is reserved for a driverless car.

A fifth assumption made by this simulation is that cars do not have unexpected failures or accidents that can entirely block off a roadway or cause massive traffic delays. Slight traffic delays are simulated by having cars only checking in at random intervals.

A sixth assumption is that the average travel times on roads are defined and written to a file of the city's roads before the simulation is run.

The seventh and final assumption is that the Vehicle Protocol notifies road change intentions well ahead of reaching an intersection and gives the CCN ample time to re-route the driverless car as necessary.

C. Simulation Implementation:

The simulation was implemented using C++11 and uses multiple threads to simulate the concurrent nature of a real network. Responses can come in at conflicting times. The use of mutexes ensures thread safety and prevents the corruption of data. The simulation has drive times in seconds. These drive times are purposely small in order to ensure that the simulation finishes in a reasonable amount of time.

The CCN runs in the main thread of the simulation. Each driverless car in the simulation runs in its own thread. Once all driverless cars in the simulator have reached their destination and left the network, the simulation ends. Figure 1 shows the state machine sequence diagram of the CCN Protocol.

CCN State Machine

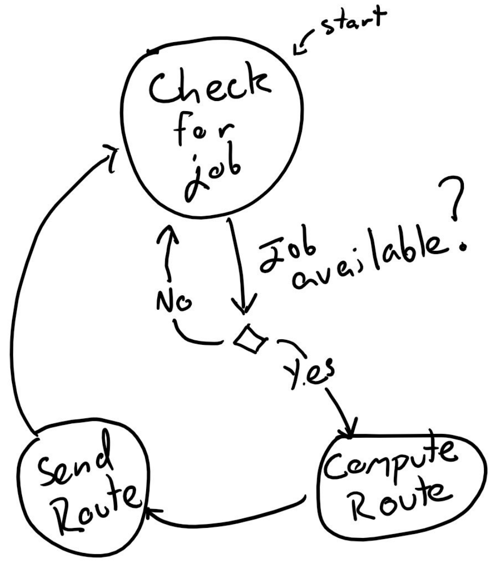

Figure 1: the state machine sequence diagram of the CCN Protocol.

The simulation was implemented using timers to track when a car has completed traversing a roadway. The simulation is kept simplistic and does not account for things such as car accidents or random delays or failures. Traffic disturbances, however, are simulated through the use of the capacity limit on roadways and effectively creates bottlenecks and error conditions: a car may be unable to turn on to a certain roadway.

Since each driverless car runs in its own thread, they also have a state machine that utilizes the messages of the CCN Protocol. Figure 2 shows the vehicle thread's state machine sequence diagram.

Vehicle Thread State Machine

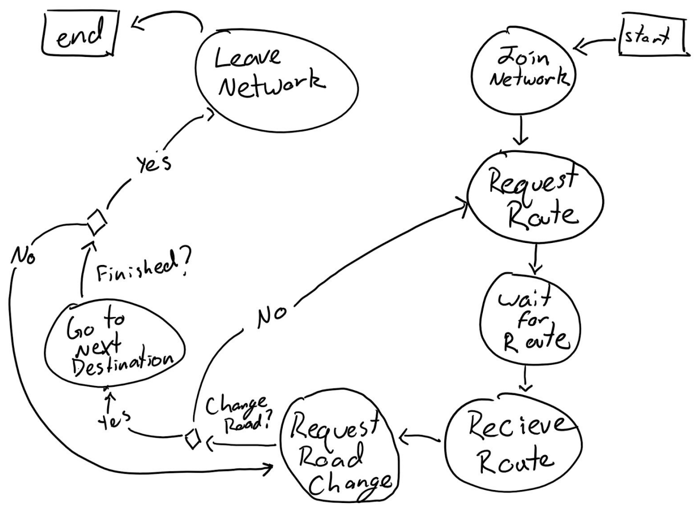

Figure 2: the sequence diagram of the vehicle thread.

The simulation uses a file, whose name is specified as command line parameter, to control how many cars are used and to design the city map. Error checking is done while reading in the contents of the file to ensure that the simulator can run on the provided input.

The implementation of the A* search algorithm used for finding optimal routes is based off of the description of the algorithm on its wikipedia page.

D. Error Handling:

It is important to discuss the errors that our simulation and protocol takes into account. Our simulation generates traffic disturbances through the use of randomly generated intervals. Each driverless car runs in their own thread with a while loop that updates the state of the car. In this while loop the driverless car's thread is put to sleep, or blocked, for a random duration. This duration causes very small delays in changing roads, departing on routes, and can actually cause delays to this or other cars depending on the capacity of the roadway they are entering or leaving.

These disturbances in traffic can cause cars to be unable to turn on to a full, or congested, street. This is considered an error and is resolved by requesting a new route from the CCN.

In terms of the actual implementation of the simulation, there is extensive error checking in the handling of command line parameters and the input file. The configuration file importer checks for errors as it processes each line of the file.

III. Novelty of the solution:

Our solution to routing driverless cars through a city is novel for a few reasons. The first reason is that our approach uses the A* search algorithm to plan the routes. Although A* search is widely used in other fields of computer science, the use of it in networking is novel. It can difficult to apply this algorithm to some problems in computer science since A* relies on the use of a heuristic. Given that a city map can be effectively measured with a heuristic, it makes sense to use A* to route traffic around a city.

A second way in which our solution is novel is the use of a centralized, reactive protocol that places the burden of responsibility on the vehicles, or clients. Although the CCN stores information about the city's roadways and the driverless cars on them, it does not actively seek out vehicles unless they have initiated a request. Road capacity levels are estimated based off of the driverless cars' road changing queries. By placing the burden on the vehicles on the roadway, management by the CCN is minimized. Minimizing the managerial responsibilities of the CCN allows for the CCN to devote all of its resources to computing the optimal routes through the city.

The third reason our protocol is novel is based on how the CCN does route caching. Routes are not cached for a set amount of time, but are instead cached only if any vehicle still needs them. As soon as all vehicles that need a certain route have received it, the route is removed from memory. This allows for the performance benefits of route caching while reducing the overhead of maintaining and updating cached routes. Let's assume the CCN calculated a route from A to B. Once this route is calculated, the route is distributed to all vehicles that have requested a route from A to B, but is not stored thereafter. This ensures that cached routes never go stale and frees up system memory for route discovery.

IV. Results and Discussion:

When the simulation is run, it performs as expected. Initially starting with a single vehicle on the network, it is obvious that the central node computes the fastest route between start and finish. However, as the number of vehicles on the network increases, the simulator continues to determine the fastest routes between these nodes.

An example run of the simulation will now be examined. For this example the time estimate of cars on roadways was scaled down in an effort to reduce the time required to run the simulation. The simulation using the example input runs in about 100 seconds. In our example, the intersection of Virginia and Kietze has a capacity of two cars. This intersection serves as a bottleneck designed to delay one of the three cars started from that intersection. This delay causes one car to request a reroute. Table 2 shows the example input car's start points and destinations.

Example Input Starts and Destinations

Car	Start	Destination
BMW	Longley-Airway	Longley-Mccarran
Volvo	Virginia-Moana	Kietzke-Neil
Subaru	Kietzke-Mccarran	Virginia-Longley
Honda	Virginia-Neil	Longley-Mccarran
Saab	Longley-Airway	Longley-Mccarran
VW	Virginia-Moana	Kietzke-Neil
Toyota	Virginia-Moana	Kietzke-Mccarran
Ford	Kietzke-Mccarran	Virginia-Longley
Hyundai	Kietzke-Mccarran	Longley-Mccarran
Mitsubishi	Virginia-Neil	Virginia-Longley

Table 2: this table shows where cars are starting and going in the example input.

The simulation was ran with everything scaled down relative to a real-world scenario. The time it takes to complete a road is scaled down, the road capacity is scaled down, and the number of cars is scaled down relative to what might actually be seen. Table 3 shows the results of running the simulation with the example input.

Resulting Times of the Simulation

Car	Expected Time (s)	Simulation Time (s)	Difference (s)
BMW	15	20	5
Volvo	55	57	2
Subaru	90	101	11
Honda	50	67	17
Saab	15	18	3
VW	55	57	2
Toyota	40	56	16
Ford	90	99	9
Hyundai	80	100	20
Mitsubishi	60	64	4

Table 3: the resulting times from running the simulation with the example input. The expected time is the smallest expected time to traverse the route without congestion or random delays.

As Table 3 shows, the cars were able to successively reach their destinations despite delays with only a small increase in travel time. Considering that the expected time does not take random delays or traffic into account, the CCN Protocol did fairly well at directing traffic in an efficient way. Figure 3 shows the graphed results of the simulation.

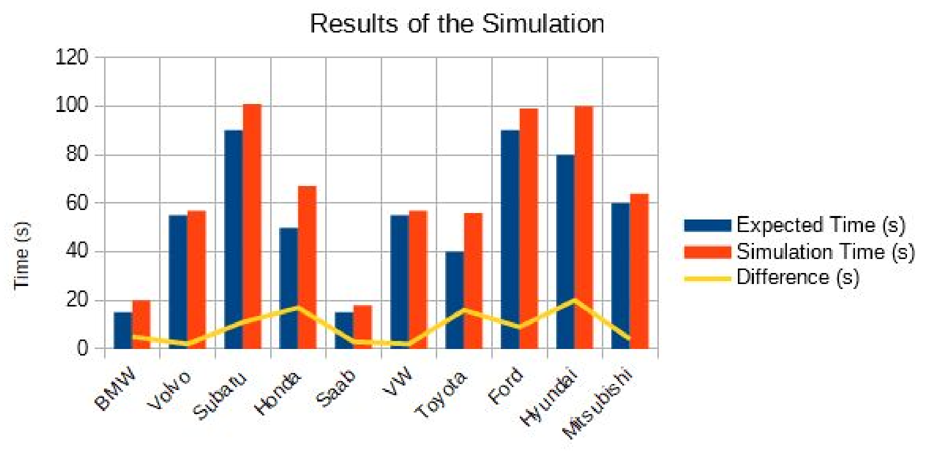

Figure 3: the graphed results of the simulation.

When taking congestion and delays into consideration, the algorithm was able to effectively route cars through a simulated city with minimal delays.

Even during periods of congestion, in which the roadway capacities are reduced significantly, the node handles the overflow and rerouting exceptionally well, with travel times remaining fairly steady, with only a small increase periodically.

The protocol works as efficiently as possible within the simulation environment, however it should be noted that this environment is not an exact representation of a real world city map. Despite this, the protocol should easily be able to adapt to any network no matter the geography or complexity.