ATS – An Autonomous Traffic Simulator

Bryant Pong, Derek Schultz, Matt Hancock
Rensselaer Polytechnic Institute
CSCI-4320 – Parallel Programming

***Abstract*: In today's world, autonomous cars are becoming more of a reality with each passing day. The prime example of such a vehicle is of Google's autonomous car. Supporters of autonomous vehicles envision a future in which every day driving hassles, such as traffic jams and car accidents, become a thing of the past. They claim that since autonomous vehicles will have the ability to communicate with all other surrounding vehicles, one's autonomous car will be able to "predict" the movements, actions, and responses of its neighbors to keep traffic moving safely and efficiently. While our team is supportive of the development and advancement of these autonomous vehicles, we are curious to see if such claims are true as the number of vehicles on a road increases. We created the Autonomous Traffic Simulator (*ATS*), a discrete-event simulation built upon the open source ROSS framework[1]. Our simulation models a simple, yet common everyday occurrence: traffic lights at intersections. We will use this simulation to model how much time the ideal autonomous vehicle will take to traverse from a starting point to a destination point when having to obey traffic lights at each intersection.**

***Keywords* –autonomous, vehicle, simulator, traffic, lights, ROSS**

I – Introduction

Autonomous cars are not a recent invention; one only needs to look at science fiction movies and novels to see countless examples of such vehicles. Perhaps the most famous autonomous vehicle is Google's Self-Driving Car (**Figure 1)**. Google uses a high-performance computer in conjunction with a variety of sensors, most notably a LIDAR (Light Detection and Range) sensor mounted on the roof.

Individual Contributions:

Bryant: Designed and implemented ATS Traffic Light Event Handlers, wrote up this paper
Derek: Implemented Autonomous Vehicle Inter-Vehicle Event Handlers
Matt: Designed and implemented ATS Traffic Light Event Handlers, wrote up this paper

*****Derek will be submitting our group's code*****

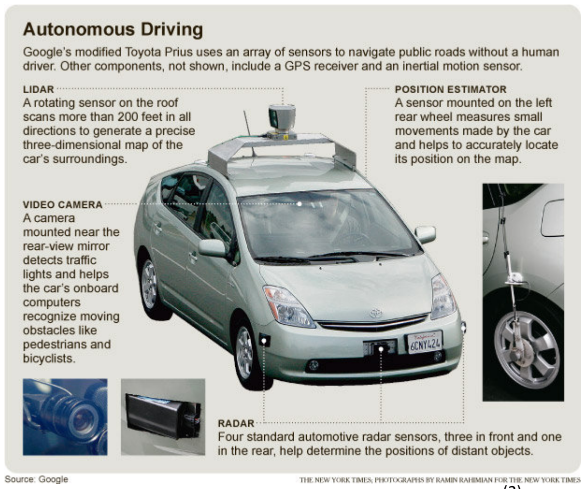
Autonomous Driving
Google's modified Toyota Prius uses an array of sensors to navigate public roads without a human driver. Other components, not shown, include a GPS receiver and an inertial motion sensor.
LIDAR
A rotating sensor on the roof scans more than 200 feet in all directions to generate a precise three-dimensional map of the car's surroundings.
POSITION ESTIMATOR
A sensor mounted on the left rear wheel measures small movements made by the car and helps to accurately locate its position on the map.
VIDEO CAMERA
A camera mounted near the rear-view mirror detects traffic lights and helps the car's onboard computers recognize moving obstacles like pedestrians and bicyclists.
RADAR
Four standard automotive radar sensors, three in front and one in the rear, help determine the positions of distant objects.
Source: Google
THE NEW YORK TIMES; PHOTOGRAPHS BY RAMIN RAHIMIAN FOR THE NEW YORK TIMES

Figure 1 – Google's Autonomous Vehicle[2]

One of the goals of autonomous vehicles is to reduce daily commuting time to and from work. Proponents of autonomous vehicles claim that these vehicles will achieve this goal because the cars are able to communicate with each other and thus synchronize their movements to improve traffic conditions and ensure a smooth flow of traffic. However, our team was interested in the case where everyone is driving an autonomous car that is capable of communicating with all other cars. We decided to write up a simulator to see how much time the average autonomous vehicle takes to traverse from one point to another as the number of vehicles increases on a fixed world size and under the condition that in each intersection on the world, there exists a traffic light.

II – The World

In ATS, the world is referring to an *X* by *Y* grid. **Figure 2** gives an example of a 3 x 3 World.

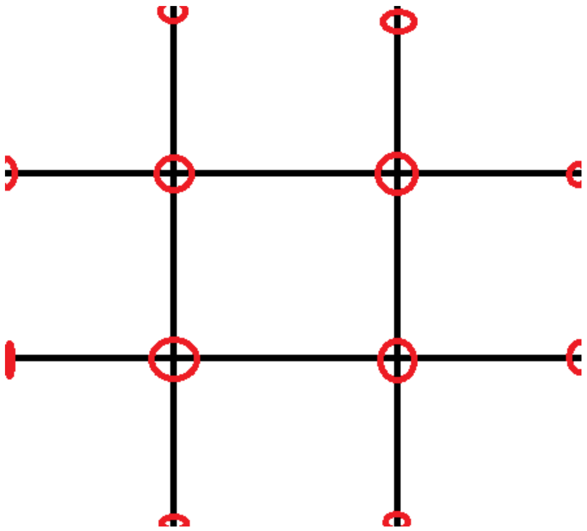
Figure 2 – A Sample 3 x 3 World

In Figure 2, each black line segment refers to a road that autonomous vehicles can travel in. Each intersecting line (indicated by a red circle) represents an intersection. At these intersections, a traffic light is placed. A red circle indicates a four-way intersection. It is important to note that the world is not finite; instead, visualize the grid is being wrapped around a sphere. In this manner, the world is infinite.

In each road, there are two-lanes; one for each direction. For each direction, there is a left-turn lane and a straight lane. An intersection is defined as a set of intersecting lines. A traffic light is placed at this intersection, and has two states: red or green. **Figure 3** shows an example of a four-way intersection with a traffic light.

In the world, there is an ending point defined that all cars are aiming to get to. The arrows in Figure 3 show the possible directions that traffic can move in. It is important to note that our traffic lights have a ten-second counter. In addition, we assume that it takes one-second for a car to cross the intersection. For our simulation, our world is a 512 x 512 square grid with 200,000 cars running. Each car is given a random starting destination and a random ending destination.

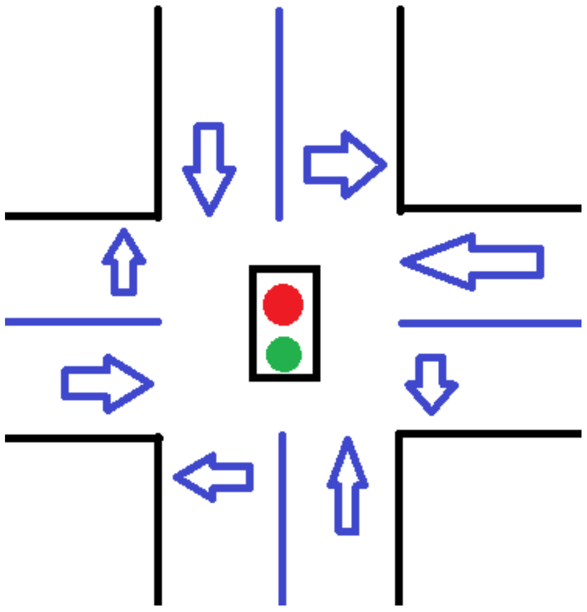
Figure 3 – A four-way intersection with a traffic light

III – Traffic Light Event Handler Algorithm

Our team drew inspiration from the "traffic" simulation provided as part of ROSS[3].

We defined two separate events that our traffic light event handler will take advantage of. These events include **light_change** and **car_arrives**. **Figure 4** shows a state table showing how the above two events interact with each other.

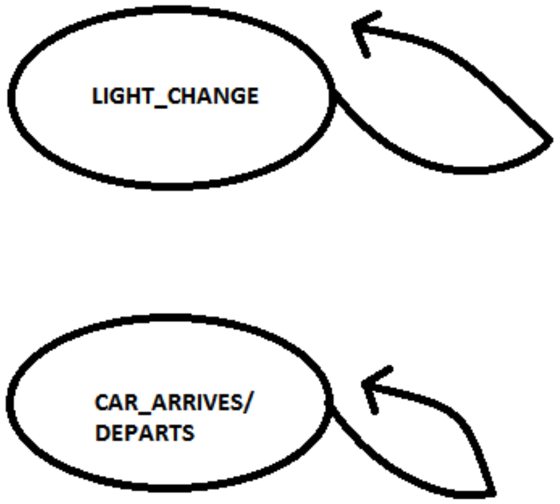

Figure 4 – State Diagram of Traffic Light Intersection Events

When a car arrives at an intersection, the intersection calculates the number of cars arriving from each direction. To get from a starting destination to an ending destination, a car will first travel either north or south in the Y-Axis until the car and its destination point both have the same Y-coordinate. Next, the car will turn left or right and then travel east or west until the car reaches its destination point. If the car has reached its destination, the event breaks and returns to main. For debugging purpose, a print statement will occur helping determine where we are in the execution. The car arrives event calculates the next intersection and direction the car must travel to. This information gets stored in the new message for the next iteration in the simulation. Finally, the CAR_ARRIVES event schedules another CAR_ ARRIVES event.

The *LIGHT_CHANGE* event, as indicated by Figure 4, is independent from the CAR_ARRIVES event. Similar to a real-world traffic light, our traffic light checks if the light timer has reached 0. If the time has expired, the light causing traffic to switch between north-south and eat-west and alternating between left-turns and straight lanes. After the light has successfully been changed, the time remaining on the traffic light is reset, and another LIGHT_CHANGE event is scheduled.

IV – Autonomous Vehicle Event Handler Algorithms

The autonomous vehicle event handler, AVEH, manages how the actual vehicles move optimally and concurrently through an intersection. We defined four events for this handler including **arrive**, **move_up**, **enter_intersection** and **exit**.

When a car arrives into an intersection, the AVEH schedules an *arrive* event. This event places the car into a queue of cars. If this car is the only car in its lane, the AVEH next schedules an *enter_intersection* event, in which the car now begins to drive into the intersection. When the car is about to leave the intersection, AVEH schedules an *exit* event. All these events are communicated between all cars in the simulation.

The above case only handles the case in which the car arriving is the first and only car in the lane. In the case when an arriving car is not the first car in line, the event *move_up* is scheduled. Move_up calculates the amount of time the car will spend waiting in line until it is at the front of the line. AVEH will continue to schedule move_up events until the car reaches the front of the line.

Figure 5 shows a state diagram of how the aforementioned four events are scheduled by the AVEH.

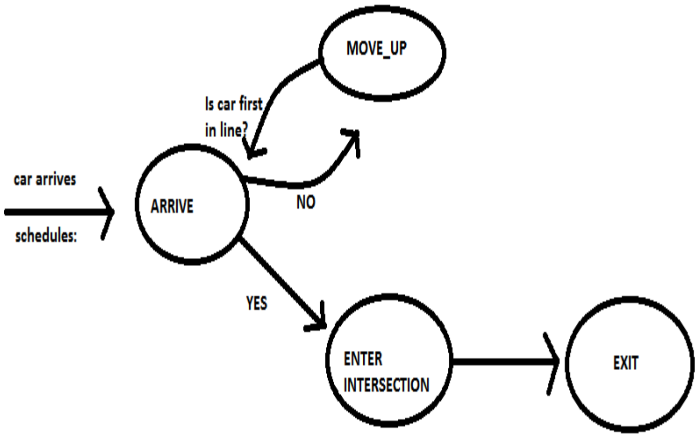

Figure 5 – State Diagram for the Autonomous Vehicle Event Handler

By default, the enter_intersection event performs a round-robin algorithm to allow cars to travel across the intersection. For instance, a car from the south lane will go, followed by a car in the west lane, then one from the north, and finally one from the east. However, to further optimize the intersection, the enter_intersection event checks to see how many cars are in the opposite directions. If no cars exist, multiple cars may cross the intersection. **Figure 6** demonstrates an example of when multiple cars may enter the intersection.

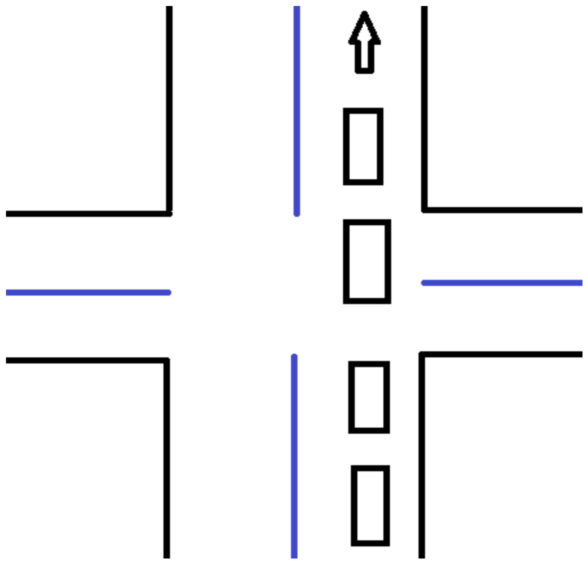

Figure 6 – Optimizing how many cars can travel across an intersection by AVEH

In Figure 6, since there are no cars arriving in either the east-west or south left-turn directions, multiple cars will schedule event_intersection events and proceed across the intersection. Another optimization is when multiple cars wish to make a right-turn from different directions. The AVEH will allow multiple cars to make these turns as long as the cars do no conflict with each other while turning. By using these optimizations, our team would like to see if autonomous vehicles truly decrease the commute time for everyone.

IV – Results of Simulation

Number Of Cores	Average Runtime per Car (in seconds)	Total Simulation Runtime (in seconds)
1	5533	1303.1202
64	5533.5625	55.9924
128	5533.4375	43.7785
256	5533.914063	34.8499
512	5533.878906	36.2796
1024	6715.641602	18.2012

Table 1 – Results from Naïve Algorithm

Our team ran a series of four separate simulations on the Blue Gene Q. We created two simulations that modeled a naïve navigation algorithm for the cars navigating through a traffic light. The other two simulations include the AVEH and its intersection optimization algorithms. We performed a strong – scaling analysis whereby our world is a grid of 512 x 512 units and cars randomly spawn in the world. The two simulations from both versions result from running each simulation in a sequential manner using ROSS's synch = 1 for a serial run and synch = 3 for a parallel run. Using synch=1 allows for a baseline for which parallelism can be compared to.

As Table 1 indicates, the car takes an average of 5533 seconds to reach the ending destination, while the entire simulation runs in 1303.1202 seconds for the naïve sequential implementation. The average runtime for the sequential run is 5533 seconds. A graph representation is shown in Figure 7. The total simulation runtime for the sequential run can be viewed in Figure 8.

However, as Table 1 indicates, in general, as the number of cores increases, the average car takes less time to reach the ending destination. The same is true for the total simulation runtime for the parallel implementation.

Even though our team only used a maximum of 1024 cores, the trend seems to indicate that as the number of cores increases, the average car will have a decreased travel time. In addition, with each increase of core, the simulation runtime will also decrease.

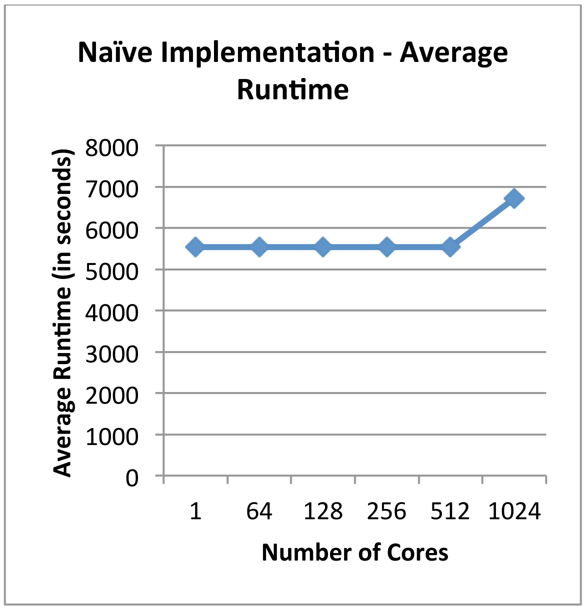

Figure 7 – Average time spent for a car traveling

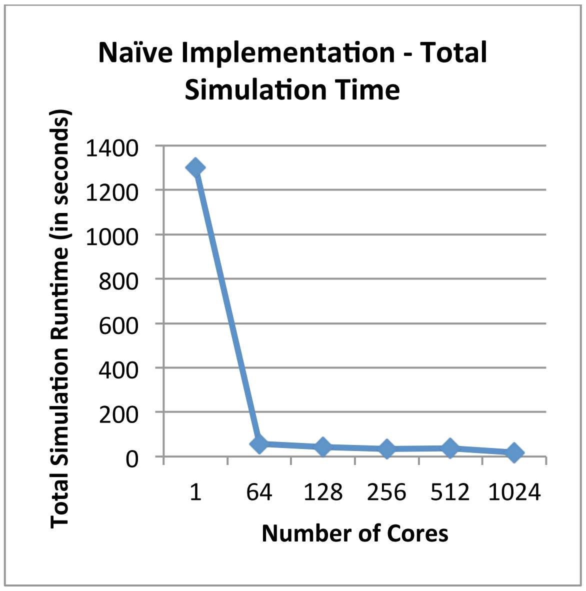

Figure 8 – Total runtime of the simulation

It is interesting that runtime for a car to reach its final destination is flat for almost all cores and 1024 cores spikes. The opposite occurs for overall simulation runtime. It continues to decrease as cores increase.

Number Of Cores	Average Runtime per Car (in seconds)	Total Simulation Runtime (in seconds)
1	4722	713.721
16	4655.0625	67.3897
32	4583.59375	34.4671
64	3241.28125	18.5808

Table 2 – Results from Autonomous cars

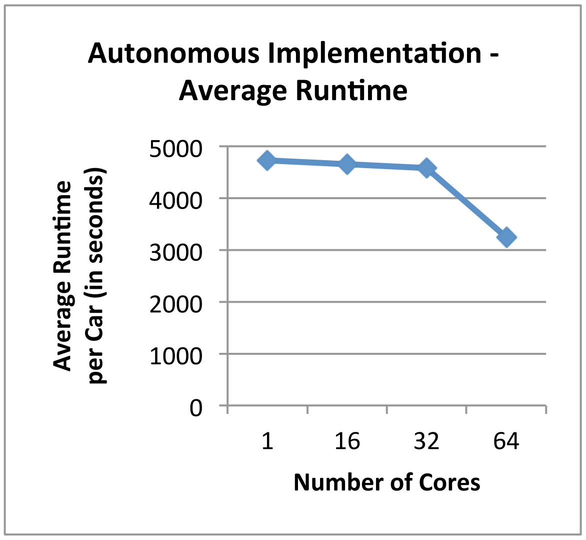

Figure 9 – Average time spent for a car traveling

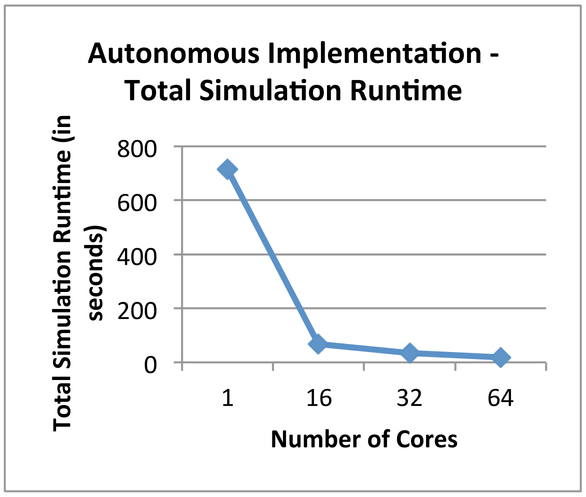

Figure 10 – Total runtime of the simulation

Figure 9 has a different trend then Figure 7. As the number of cores increase, the total time for a car to travel decrease. It especially decreases when we use 64 cores. Figure 10 describes the overall execution time. It shows how time decreases with each increase in core. Comparing between traffic light and autonomous, autonomous runs much faster. This proves our point that autonomous cars that communicate with each other is beneficially to faster commute times and less traffic.

V – Network and Computation Performance

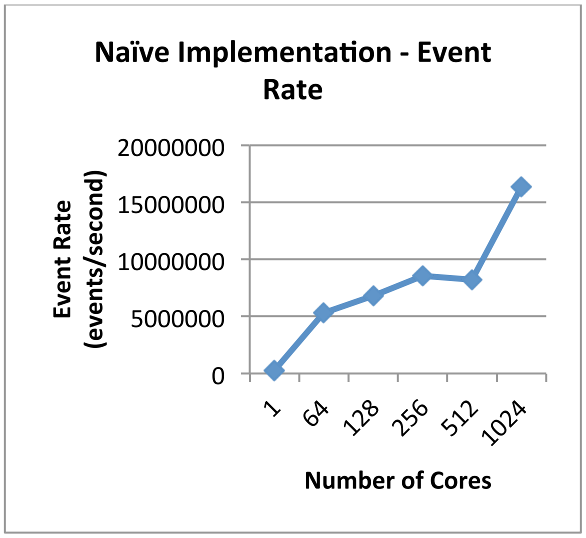

Figure 11 – Events computed per second

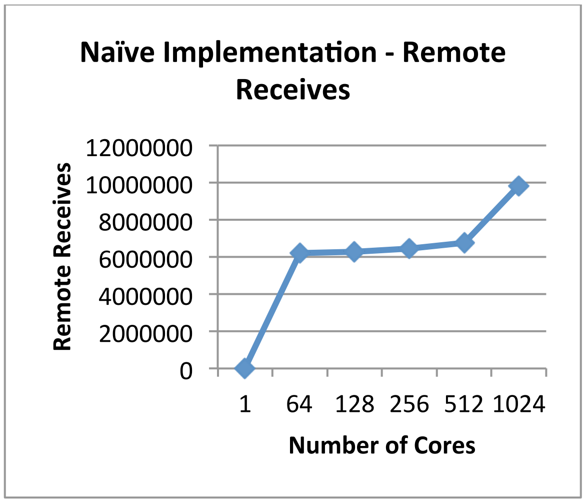

Figure 13 – Total receives that occurred

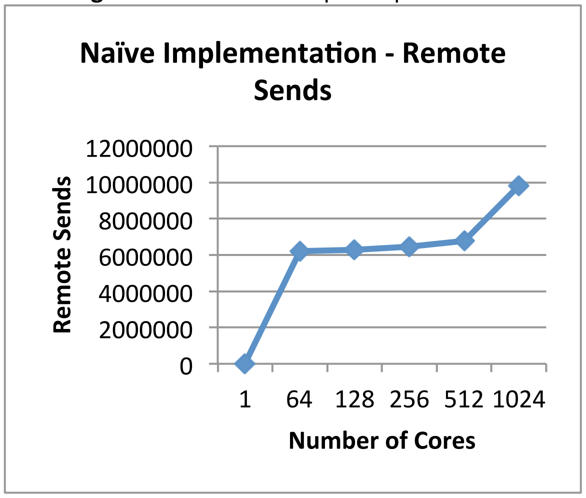

Figure 12 – Total sends that

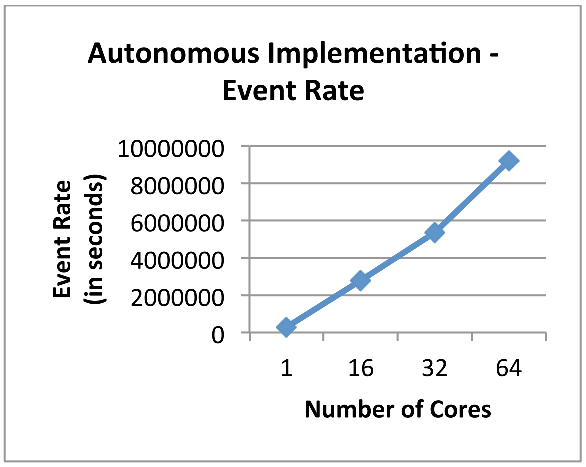

Figure 14 – Events computed per second

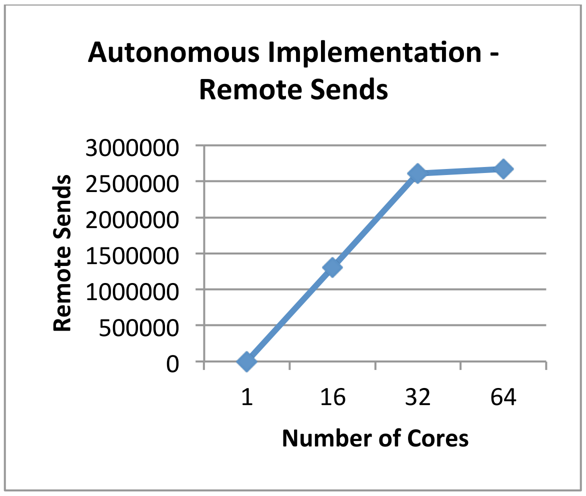

Figure 15 – Total sends that occurred

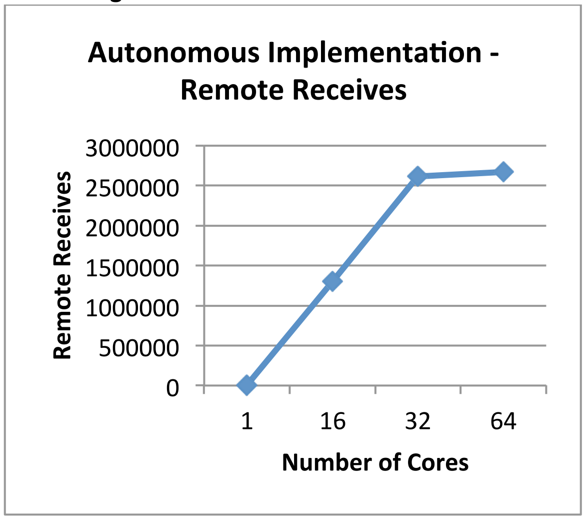

Figure 16 – Total receives that occurred

Figures 11, 12 and 13 are all related to the performance with intersections using traffic lights. The data shows that as cores increase, the overall events that get processed increase. This makes sense as with each increase in cores, each core has less to compute allowing more events to be processed. In addition, the number of sends and receives are very similar in their trends. They both increase as cores increase. Figures 14, 15 and 16 are all related to autonomous cars. The data here shows similar trends. As the cores increase, the number or processed events increase for the same reason as the traffic light scenario. The number of sends and receives again have very similar trends. Comparing between traffic light and autonomous cars, traffic lights process more events per second. Traffic lights also have more sends and receives.

VI – Future Research

The currently developed world is simplified compared to a real world setting. Further research can be done using actual map data from Google Maps. This can accurately determine travel time between lights, the maximum number of cars between intersections, the speed limit and many more factors. In addition, adding more than one lane to an intersection creates a more realistic map. Some intersections have a combination of one to three lanes with left turning lights. Lastly, the current model only has red and green lights. Adding the yellow light will have new effects on how cars move from intersection to intersection. Most drivers tend to speed up to get through a yellow light. The newer model can mimic the same behavior.

References

1. "ROSS Wiki Main Page", *Rensselaer Polytechnic Institute*. http://odin.cs.rpi.edu/ross/index.php/Main_Page, 2013.

2. "Google-self-driven-car.jpeg", *Alecdifrawi*. http://www.alecdifrawi.com/wp-content/uploads/2011/08/google-self-driven-car.jpeg, 2013.

3. "Traffic ROSS Simulation", *Rensselaer Computer Science*. http://www.cs.rpi.edu/websvn/filedetails.php?repname=rossnet&path=%2Ftrunk%2Fross%2Fmodels%2Ftraffic%2FIntersection.c, 2013.

4. "Adaptive Traffic Lights Using Car-to-Car Communication" http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4212445

5. "A car-fllowing theory for multiphase vehicular traffic flow" http://www.sciencedirect.com/science/article/pii/S0191261504000864