Vehicle-to-Vehicle Communication

CSCI 4950 – Senior Project

James Garrett Gruber  
 University of North Georgia

Dahlonega, Georgia. USA.  
 [jggrub4022@ung.edu](mailto:jggrub4022@ung.edu)

**ABSTRACT**

Vehicle-to-vehicle (V2V) communication is a fairly new and still developing technology that allows for wireless communication between two or more vehicles, as well as with roadside infrastructure. There are two major methods that would be used to implement V2V communication. The first is utilizing pre-existing cellular networks, and the second, more direct method is through the use of dedicated short-range communication (DSRC). More than likely a combination of both methods will be used to fully implement the technology. V2V communication could be used to help navigation, prevent crashes, allow drivers to directly communicate with other drivers (a possible replacement for honking), and aiding law enforcement and emergency services. Problems with implementing this technology include security risks, extremely limited support among vehicles on the road today, reliability, and regulation. This paper will document the emulation of numerous scenarios that show the functionality of V2V communication, as well as my proposal of the TAG Device.

KEYWORDS

Vehicle-to-Vehicle, Communication, Smart Cars, Infrastructure, Java, Emulation, TAG Device

1. INTRODUCTION

Communication between vehicles is not a new concept. Drivers have always been able to convey information to other drivers. When the automobile was first invented, drivers had the ability to yell to other drivers or bystanders, and they could also make hand signals. As cars became more advanced, so has the methods of communication between them. Technologies like turn signals, brake lights, and the horn allowed for greater understanding between drivers, resulting in safer and more organized roadways.

In recent years, a new form of communication between cars is emerging: wireless communication. The concept of implementing cellular network technology in cars is known as Vehicle-to-Vehicle (V2V) communication. When fully implemented, cars will be able to relay large amounts of information amongst themselves, often without requiring any action from the driver [1].

2. APPLICATIONS

There are many uses for Vehicle-to-Vehicle communication. For example, instead of honking a horn at another car which could be misinterpreted in numerous ways, a driver could send a direct message to the other driver that clearly states his or her intentions. No other drivers would get this message, whereas all drivers in the vicinity would hear a horn and wonder if it was meant for them.

Other uses of V2V communication include faster brake response. A computer would be able to react must faster than a driver if the car in front of them suddenly hits their brakes. The braking car would wirelessly signal the car behind it, so the two cars would brake almost simultaneously.

Another possible use relates to emergency vehicles. As an emergency vehicle (ambulance, firetruck, police cruiser, etc.) approaches, it could wirelessly alert the vehicles in its path, giving more time for them to get out of the way/ pull over.

There is another form of V2V communication called Vehicle-to-Infrastructure (V2I) communication. This refers to the transmission of information between vehicles and stationary roadside devices [1]. An example of this being used is a wireless toll booth. As a vehicle enters a toll road, it passes a device. The car sends its identification information to the device, including the driver’s information, and charges the driver automatically. This allows the flow of traffic to not be interrupted.

Another use is automatically notifying a vehicle of the current speed limit. The roadside devices would be placed where speed limit signs today are, as well as at any intersection or place where a vehicle enters the road. The device would signal to the car, which would display the limit in a way the driver can see it. It would automatically update when the speed limit changes and can also alert the driver when the limit is about to change.

Traffic light intersections could also benefit from the implementation of V2I communication, more specifically with self-driven smart cars. The lights would automatically notify the smart cars when they turn green. Additionally, vehicles can be notified about upcoming red lights much sooner.

3. IMPLEMENTATION

3.1 Problems

The major problems with implementing this technology include security risks, extremely limited support among vehicles on the road today, reliability, and regulation.

As with all wireless communication, security is a very important topic in V2V. When V2V technology is finally implemented, data transferred between cars must be secured in real time to prevent tampering [2]. Cryptographic algorithms, encryption techniques, and GPS-based localization algorithms [3] are a few proposed solutions to this issue. I do not address this problem in my experiments, but it would not be difficult to add security measures to them were the measures fully planned out.

The particular problem that I do address in this paper is limited support: not all cars on the road today would be able to technologically support V2V in their current states. Assuming we make it mandatory for all cars released from here on out support V2V communications, that still leaves the large majority of cars out of the picture. These vehicles would have to be included in the V2V network for the technology to work. My proposed solution for how to achieve this is the TAG device.

3.2 TAG Device

Similar to how all new vehicles manufactured in the US were required to be OBD-II specified as of 1996 [4], we could require all new vehicles to meet V2V specifications as of 2025[[1]](#footnote-2). Vehicles released before this year that do not support V2V communications must use a government approved TAG device. The TAG would plug into a car’s OBD-II maintenance port and directly access the car’s engine control module (ECM). Each device is preprogrammed to include identification information for the car it is used with. It can send and receive basic levels of data wirelessly, essentially connecting the car to the V2V network.

The TAG would also include a visual aid to represent incoming information. The bare minimum for this would be a multicolored LED, which is what is used in my scenarios. Manufacturers can of course expand on this within federal regulations, as is being done with existing aftermarket safety devices [5]. For example, a higher end TAG model could interface with the user’s smartphone and utilize a GUI.

Since they are connected to the network, TAG devices would be able to automatically receive and install firmware and security updates. In addition, TAGs can be used for real-time and automatic emissions testing because they are linked directly to the ECM, potentially eliminating the need for emission testing sites.

TAGs must be inspected yearly to check for tampering or malfunction, similar to emissions tests. TAGs that are not inspected or fail inspection are flagged in the V2V network and visible to law enforcement on the road. Finally, like the OBD-II technology they interface with, TAG devices would need to be designed to prevent data tampering through its own software [4].

4. MY APPROACH

For my project, I emulated some basic scenarios that V2V communication might involve. This was done primarily with Java. In these examples, dedicated short-range communication (DSRC) is the implementation method used, although the same results could definitely be achieved using pre-existing cellular networks or a combination of both. DSRC protocol designed specifically for V2V and V2I communication, and it can work in one direction or two [1].

What follows is a breakdown of the Java programs I wrote, as well as their outputs. I wrote three classes in all, one constructor class and two classes for demonstration.

4.1 Vehicle Constructor Class (Vehicle.java)

To properly manage all the cars in the simulation and give each one its own attributes, every car had to be an object. This was achieved using a constructor class, which I called “Vehicle.java”.

*4.1.1 Identifiers.* When a new vehicle object is initialized, the following identifiers must be declared as parameters:

**private** String Make, Model;

**private** **int** Year;

**private** String VID;

**private** **boolean** IsSmart, IsEmergency;

These are used to identify the object and are not supposed to be modified as a program executes. Despite this, I did include setter methods for each of these identifiers, as well as getter methods.

The next group of identifiers are set and modified as the program executes. These are used to describe where and how each vehicle is driving.

**private** **int** Speed, Lane;

**private** **char** Direction;

**private** **int** Position;

These identifiers also each have their own individual setter and getter methods. In addition, there is also a method to set the Speed, Lane, Direction of a car all at once:

**public** **void** setDrivingInfo(**int** speed, **int** lane, **char** dir, **boolean** print)

If the “print” boolean is set to true, a summary of the driving info will be printed.

The integer Position is the vehicle’s relative displacement (in feet) from the next mile marker[[2]](#footnote-3). Cars with lower Position values are closer to the marker and are ahead of cars with higher values. Cars with negative Position values are traveling in the opposite direction than cars with positive values, as they have already passed the marker. The Lane value works similarly to this, as the opposing lane is given a negative value. See **Figure 2A** for an example of this.

The final two identifiers are not required to be declared when object is initialized and are meant to remain constant after they are declared. They are optional because not every scenario used them. These identifiers each have a set and get method.

**private** String Console, Name;

*4.1.2 Methods.* There are several other methods that I included in “Vehicle.java” in addition to the basic set and get methods:

**public** **void** summerizeID() 🡪 Prints a formatted summary of the vehicle’s required identifiers (Make, Model, Year, VID, IsSmart, IsEmergency).

**public** **void** summerizeDriving() 🡪 Prints a formatted summary of the vehicle’s driving characteristics (Speed, Lane, Direction). This is what “setDrivingInfo()” runs if its boolean is set to true.

**public** String compassDir() 🡪 Converts the **char** Direction to a string, i.e. ‘N’ returns as “NORTH”.

**public** String yearMakeModel() 🡪 Returns a string of a vehicle’s Year, Make, and Model.

4.2 Scenario 1: Simple Direct Messaging (ScenarioOne.java)

This program demonstrates direct message communication between vehicles. There are two recipient types, smart cars and non-smart cars, each with different message/ signal sets. Smart cars can receive text messages and can also reply to the sender. Non-smart cars receive a color signal that displays through their TAG device but have no methods through which to reply.

*4.2.1 Setup.* The program first creates 3 vehicle objects, labeled as {CAR 1}, {CAR 2}, and {CAR 3}. The first two cars are smart cars, while the third is not but does have a TAG device. The summaries of these cars and their driving information is printed in the console as follows:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VEHICLES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{CAR 1} Created:

Vehicle: 2018 Tesla Model 3

VID: 111

Smart Vehicle: Yes

Emergency Vehicle: No

The 2018 Tesla Model 3

is currently heading NORTH at 30 MPH

in lane number 1.

{CAR 2} Created:

Vehicle: 2021 Honda Civic

VID: 222

Smart Vehicle: Yes

Emergency Vehicle: No

The 2021 Honda Civic

is currently heading NORTH at 26 MPH

in lane number 1.

{CAR 3} Created:

Vehicle: 2000 Oldsmobile Bravada

VID: 333

Smart Vehicle: No, TAG Device Equipped

Emergency Vehicle: No

The 2000 Oldsmobile Bravada

is currently heading NORTH at 35 MPH

in lane number 2.

The scenario itself executes as a series of menus in the vehicle’s V2V consoles. The driver selects which nearby vehicle ({CAR 2} or {CAR 3}) to send a message to. The console displays a summary of the selected vehicle.

If they choose {CAR 2}, they are prompted to select a list of predefined text messages. The selected message then displays in the console of {CAR 2}, and its driver is prompted to choose from a list of replies (or choose not to reply at all). The replies shown to the user at this step will be different depending on the message they received. This is achieved through two levels of nested switch statements.

If the driver of {CAR 1} chooses to send a signal to {CAR 3}, they are prompted to choose from a list of color signals, each one with a different meaning (such as “Thanks” or general negative sentiment). The selected signal is sent to {CAR 3} and is displayed in its console.

*4.2.2 Results.* If the driver of {CAR 1} selects {CAR 2} as the message recipient and selects “Please let me pass” as the message, and the driver of {CAR 2} replies with “Sorry, I cannot”, the program outputs as follows: (Green numbers are user input)

=========BEGIN Simulation=========

[{CAR 1} Console] Which vehicle would you like to send a message to?

[{CAR 1} Console] Nearby Cars:

1: 2021 Honda Civic

2: 2000 Oldsmobile Bravada

[{CAR 1} User] 1

[{CAR 1} Console] Recipient Vehicle Summary:

Vehicle: 2021 Honda Civic

VID: 222

Smart Vehicle: Yes

Emergency Vehicle: No

[{CAR 1} Console] Select message to send:

1: Slow Down!

2: Please let me pass.

3: Sorry!

4: Go ahead.

5: Thanks!

[{CAR 1} User] 2

[{CAR 1} Console] Successfully sent message "Please let me pass." to {CAR 2} (2021 Honda Civic).

[{CAR 2} Console] Message received from {CAR 1} (2018 Tesla Model 3).

[{CAR 2} Console] {CAR 1} says "Please let me pass."

[{CAR 2} Console] Reply?

0: Do not reply.

1: I will try.

2: Sorry, I cannot.

[{CAR 2} User] 2

[{CAR 2} Console] Successfully sent reply "Sorry, I cannot." to {CAR 1} (2018 Tesla Model 3).

[{CAR 1} Console] Reply received from {CAR 2} (2021 Honda Civic).

[{CAR 1} Console] {CAR 2} says "Sorry, I cannot."

=========END Simulation=========

If the driver of {CAR 1} selects {CAR 3} as the message recipient and selects “Green Signal (Positive Sentiment)” as the message, the program outputs as follows: (Green numbers are user input)

=========BEGIN Simulation=========

[{CAR 1} Console] Which vehicle would you like to send a message to?

[{CAR 1} Console] Nearby Cars:

1: 2021 Honda Civic

2: 2000 Oldsmobile Bravada

[{CAR 1} User] 2

[{CAR 1} Console] Recipient Vehicle Summary:

Vehicle: 2000 Oldsmobile Bravada

VID: 333

Smart Vehicle: No, TAG Device Equipped

Emergency Vehicle: No

[{CAR 1} Console] Select signal to send:

1: Red Signal (Negative Sentiment)

2: Green Signal (Positive Sentiment)

3: Blue Signal ("Sorry")

4: Yellow Signal ("Thanks")

[{CAR 1} User] 2

[{CAR 1} Console] Successfully sent <GREEN> signal to {CAR 3} (2000 Oldsmobile Bravada).

[{CAR 3} TAG SIGNAL READER] <GREEN>

=========END Simulation=========

4.3 Scenario 2: Intelligent Braking (ScenarioTwo.java)

This program simulates “Smart Brakes,” a system of intelligent braking that utilizes V2V communication. As a smart car applies its brakes, its computer sends a signal to the car following behind it. The following car would then automatically apply its own brakes, then signal the car behind it, and so on.

The initially braking car would only send a signal to cars that satisfy the following conditions, which are checked using nested if statements.

* The two cars are heading in the same direction
* The two cars are in the same lane
* The braking car’s position value is less than the second car’s position value. In other words, the second car is behind the first car
* The second car is in “close proximity” to the braking car. This distance is determined by dividing the second car’s speed by 10 (with no remainder) then multiplying this number by 15 (representing the average car length in feet).

These if statements are in the method **public** **boolean** isBehind(Vehicle car2), which is located in the “Vehicle.java” constructor class.

In the “ScenarioTwo.java” class, the static void method brake handles the action of the smart car applying its and sending a signal.

**public** **static** **void** brake(Vehicle car1, ArrayList<Vehicle> cars, **boolean** indent, **boolean** forward)

In regard to the parameters, car1 is the braking vehicle. cars is an ArrayList containing all vehicles besides the braking vehicle. A new ArrayList is initialized for each scenario. The boolean indent is used to determine if the indentation is reset, and forward represents whether or not the method is being run as a reaction to another car braking.

When executed, the brake method displays the V2V console of the braking vehicle and its outputs, as well as the console outputs of other cars affected by the braking.

A for loop is used to run the isBehind() method on each vehicle in the ArrayList. If a vehicle that meets the conditions is found, that vehicle (assigned to temp) will also run the brake method. This recursive method is called as *brake*(temp, cars, **false**, **true**);.

If no vehicle that meets the conditions is found in the ArrayList, the V2V console displays a message conveying this.

The first action “ScenarioTwo.java” does when executed is create 4 vehicle objects, labeled as {CAR 1}, {CAR 2}, {CAR 3}, and {CAR 4}. Each vehicle’s identification information summary is printed out in the console.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VEHICLES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{CAR 1} Created:

Vehicle: 2018 Tesla Model 3

VID: 111

Smart Vehicle: Yes

Emergency Vehicle: No

{CAR 2} Created:

Vehicle: 2021 Honda Civic

VID: 222

Smart Vehicle: Yes

Emergency Vehicle: No

{CAR 3} Created:

Vehicle: 2000 Oldsmobile Bravada

VID: 333

Smart Vehicle: Yes

Emergency Vehicle: No

{CAR 4} Created:

Vehicle: 2004 Toyota Tundra

VID: 444

Smart Vehicle: Yes

Emergency Vehicle: No

Each car is traveling at 70 mph for all scenarios. With the exception of {CAR 1} which remains constant, the position, direction, and lane of each vehicle depends on the scenario and can be seen in the included figures. Any changes to these values are done before the scenario’s brake method is executed.

***4.3.1 Scenario 2A.***

*4.3.1.1 Setup.* In this scenario, shown in **Figure 2A,** {CAR 1} brakes and sends a signal to {CAR 2}, which is closely following it. The signal is received and causes {CAR 2} to also brake.

{CAR 1} and {CAR 2} should be the only cars that brake in this situation. {CAR 3} will not brake because it is in a different lane, and {CAR 4} will not brake because it is in a different lane and going in a different direction. {CAR 2} receives the signal because it is in the same lane, going the same direction, and positioned within the minimum distance of 105ft ((70mph/10mph)\*15ft).

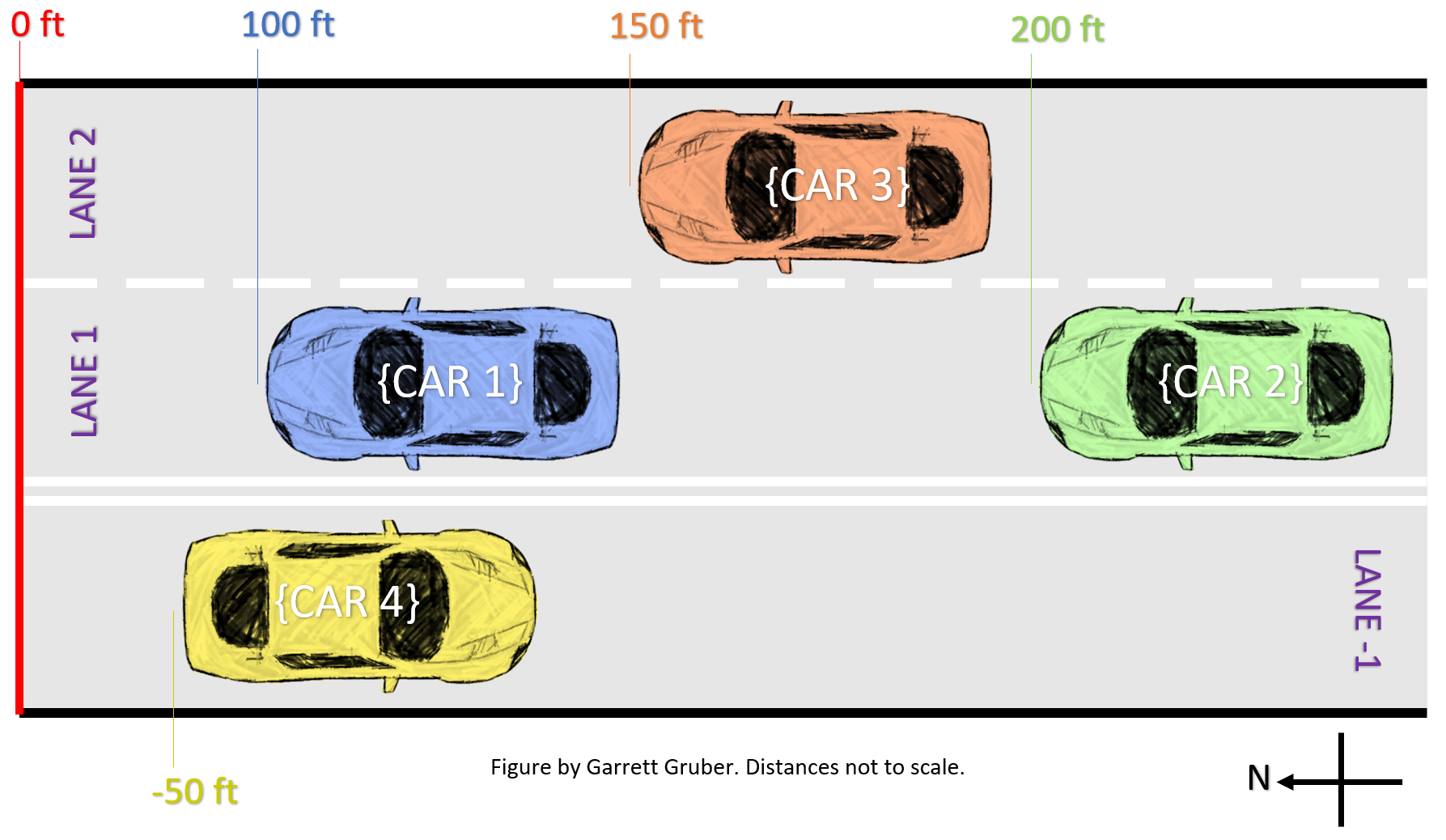


Figure 2A: A typical highway scenario with four cars

*4.3.1.2 Results.* The output for this segment is as follows. Everything works as intended.

=========BEGIN Simulation=========

[{CAR 1} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 1} Console] Signaling vehicle {CAR 2} (2021 Honda Civic)

[{CAR 2} Console] Smart Brake signal alert received.

[{CAR 2} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 2} Console] No appropriate vehicle found. Signal not sent.

=========END Simulation=========

***4.3.2 Scenario 2B.***

*4.3.2.1 Setup.* This scenario, shown in **Figure 2B,** is almost identical to *Scenario 2A,* with the only difference being {CAR 2} is now 400 ft from the mile marker as opposed to the previous 200 ft. {CAR 2} is still travelling at 70 mph, so the minimum distance between the two cars is still 105 ft. The actual distance between the two cars is now 300 ft. As a result, no cars other than {CAR 1} should brake.

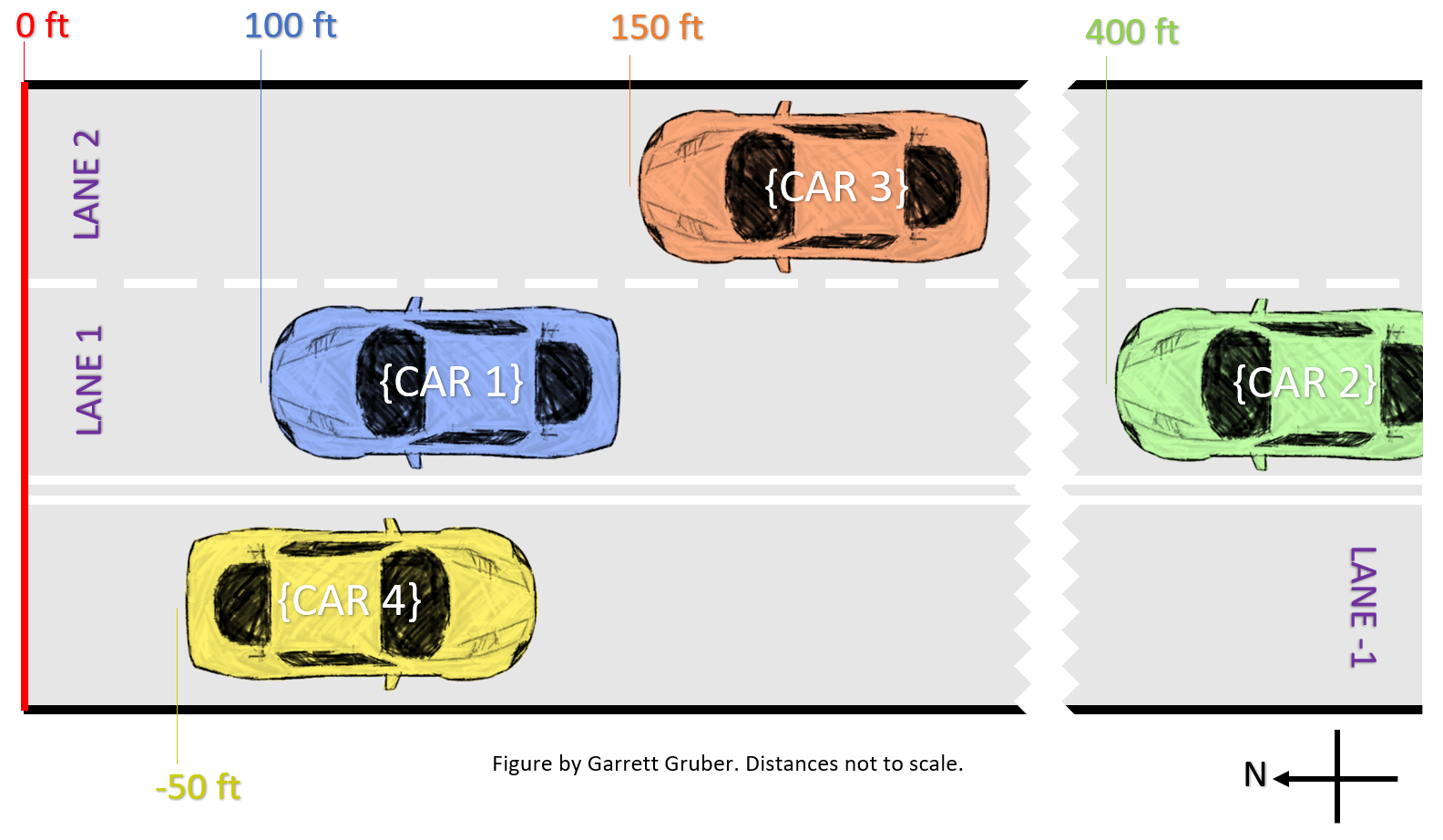


Figure 2B: {CAR 2} is now much farther away from {CAR 1}

*4.3.2.2 Results.* The output for this segment is as follows. Everything works as intended.

=========BEGIN Simulation=========

[{CAR 1} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 1} Console] No appropriate vehicle found. Signal not sent.

=========END Simulation=========

***4.3.3 Scenario 2C.***

*4.3.3.1 Setup.* This scenario, shown in **Figure 2C,** depicts a situation in which all cars are in the same lane and heading in the same direction. As {CAR 1} brakes, all of the cars behind it should apply their brakes simultaneously. This scenario best demonstrates the way the brake method functions recursively.

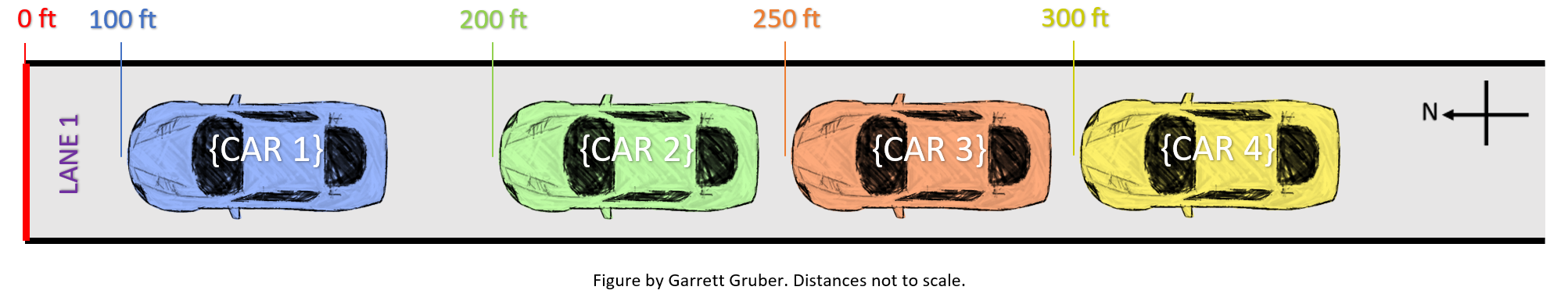


Figure 2C: All cars are in a single file line

*4.3.3.2 Results.* The output for this segment is as follows. As intended, each car automatically brakes as a response to the car in front of it.

=========Begin Simulation=========

[{CAR 1} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 1} Console] Signaling vehicle {CAR 2} (2021 Honda Civic)

[{CAR 2} Console] Smart Brake signal alert received.

[{CAR 2} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 2} Console] Signaling vehicle {CAR 3} (2000 Oldsmobile Bravada)

[{CAR 3} Console] Smart Brake signal alert received.

[{CAR 3} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 3} Console] Signaling vehicle {CAR 4} (2004 Toyota Tundra)

[{CAR 4} Console] Smart Brake signal alert received.

[{CAR 4} Console] Brakes applied. Signaling any trailing vehicle(s)...

[{CAR 4} Console] No appropriate vehicle found. Signal not sent.

=========END Simulation=========

5. ANALYSIS

Though very basic and not nearly as complex as reality, my simulations still convey the usefulness of using V2V communication. However, several factors were not taken into account in my tests that would need to be addressed before actual implementation. In the case of direct messaging, safety would be the biggest concern. Texting while driving is already a huge problem, and to avoid adding to that, this technology must be implemented differently. Perhaps it could be entirely controlled by voice, or even completely automated by the thoughts of the driver. Receiving messages would also need to be very unobtrusive so as to minimize the distraction it causes the driver. The choice to receive messages would also be optional, and drivers should have the ability to block other drivers from communicating with them in this way.

Smart braking would also need to address some problems. While my program is assuming that the braking that occurs is a sudden, hard brake meant to dramatically decrease your speed (e.g. “slamming the brakes”), in reality there are other levels of braking. Sometimes drivers apply the brakes for less than a couple seconds just to drop 5 mph. The system would need to either ignore this minor braking or send a different signal for this. The same would need to be addressed for tapping the brakes to exit cruise control. Optionality would also need to be implemented, allowing the driver to override the automatic braking. A car receiving the braking signal could have 3 options that the driver would set before hand: automatically brake, display an audio/visual cue to tell the driver to brake, or ignore the signal all together. The braking car would always send out a signal, similar to how braking lights are required. In addition to this, my tests assume that the receiving car is a smart car, which will definitely not be the case for the majority of the time. I would look to my TAG device idea to address this. The TAG would receive the signal and alert the driver.

In regard to the TAG device itself, security and regulation would be the biggest issues. Smart cars that exist today already have to deal with hacking, and the same would be true for any V2V add-on device. However, security updates would be relatively easy to implement given the technology’s integral network connection. As for regulation, proper and effective legislation would be needed to ensure widespread use.

6. CONCLUSION

Vehicle-to-Vehicle communication is a budding technology with seemingly limitless potential. I learned a lot about it in the few experiments I performed. A surprisingly large number of factors need to be considered for implementation, more than I anticipated. Even a simple messaging program required over two hundred lines of code.

If I were to do any parts of this project differently with more time, I would have rewritten the Vehicle.java class to contain most of the code from the scenarios as methods. This would have allowed all the scenarios to be run in a single class and creating new tests theoretically would be easier. I would have also made the operations (messaging, braking) more dynamic so that different cars could be used under the same conditions. For example, sending a message as {CAR 2}, or braking as {CAR 4}. Finally, I would have done more with V2I technologies and network communications.

7. REFERENCES

[1] Smith, C. (2016). *The Car Hacker's Handbook.* San Francisco, CA: No Starch Press.

[2] Nagarjuna Chidara, Chinmayi Nibhanupudi, and Hrishikesh Venkataraman. 2016. Integration of different cryptographic techniques for real-time V2V communication: poster. In Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '16). ACM, New York, NY, USA, 379-380.

[3] Heath J. LeBlanc, Firas Hassan, Edgar Gomez, and Nesreen Alsbou. 2016. Inter-vehicle communication assisted localization with resilience to false data injection attacks. In Proceedings of the First ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys '16). ACM, New York, NY, USA, 64-65.

[4] Lu Yu, Juan Deng, Richard R. Brooks, and Seok Bae Yun. 2015. Automobile ECU Design to Avoid Data Tampering. In Proceedings of the 10th Annual Cyber and Information Security Research Conference (CISR '15). ACM, New York, NY, USA, Article 10, 4 pages.

[5] Harding, J., Powell, G., Yoon, R., Fikentscher, J., Doyle, C., Sade, D., . . . Wang, J. (2014). *Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application.* Washington, DC: National Highway Traffic Safety Administration.

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00

1. This year is just an estimate for when both V2V technology is ready to be mass implemented and proper legislation and federal requirements for implementation have been passed. [↑](#footnote-ref-2)
2. The mile markers used in this situation could be considered as V2I if they actively connect with the car through the V2V network to determine its position value. [↑](#footnote-ref-3)