Coordination Schemes for Autonomous Vehicle Networks

Jack Ziegler
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
ziegl210@morris.umn.edu

ABSTRACT

In this paper, we provide an overview of the potential benefits of an Intelligent Transportation System, a network of communicating and cooperating autonomous vehicles. We also discuss the pitfalls and limitations that such a system faces. We then look at the results of several research attempts to address these issues through multi-agent communication and coordination, which can be used to improve the safety and efficiency of individual vehicles and the network as a whole.

Keywords

Autonomous Vehicles, vehicular ad-hoc networks (VANET), vehicle-to-vehicle (V2V) communications, Intelligent Transportation System (ITS)

1. INTRODUCTION

Over the last few years, autonomous vehicles have quickly moved from a feature of fiction and television to an imminent reality. In order to safely and effectively navigate the modern transit system, an autonomous vehicle must be able to accurately and efficiently analyze and react to its surroundings. While the sensors available to current autonomous vehicles are sophisticated, the technology does have limitations that must be recognized. Research efforts have been made to improve the efficiency and safety of these devices through communication and coordination schemes [7, 6, 1, 4]. In this paper, we will discuss how an Intelligent Transportation System (ITS) can allow networks of vehicles to communicate with one another, and improve the effectiveness of our system. We will begin by providing relevant background details on autonomous vehicles and the ITS in the next section. Then, we will investigate three coordination approaches. In Section 3, we will investigate a communications system called CarSpeak, which allows vehicles to share sensor information. Then in Section 4, we discuss how a remote cloud can be used to offload data and computation from the vehicle. Finally, in Section 5 we explore traffic aware routing and how real-time information can improve navigation.

2. BACKGROUND

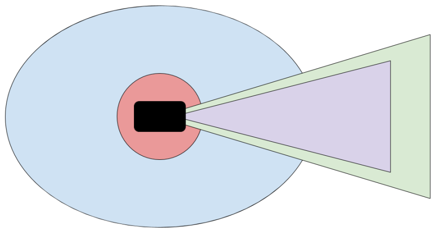

Figure 1: Sensors overlap at varying ranges. Here is a simplified graphic for Tesla vehicles. The long-range sensors are radar and narrow cameras.

2.1 Autonomous Vehicles

Autonomous vehicles are simply vehicles which do not require human operation. There are various competing designs, but at a high level all autonomous vehicles consist of a set of sensors and a computer for control. The sensors found on a vehicle vary based on manufacturer, but usually include some combination of cameras, sonar devices, radar, and LIDAR. In existing vehicles, auto manufacturer Tesla includes sonar sensors for distances under 10 meters, and a combination of radar and cameras for distances between 50-250 meters [10].

However, outside this range a vehicle is unable to gain useful information about the environment. This can lead to less effective responses to events such as accidents or lane closures, where advanced notice can be used to mitigate traffic interruptions. In addition to range concerns, the on-board sensors generate large amounts of data for processing and so the on-board computer must be quite powerful, and capable of tasks such as vision processing. These limitations make a coordination strategy appealing.

2.2 VANETs and Communication

For local communications, A *VANET*, or Vehicular Ad-Hoc Network, is a dynamic network of connected vehicles. These networks are often supported by *road-side units,* which are static objects that can be used for data storage, computation, or network management. A VANET allow vehicle-to-vehicle and vehicle-to-roadside communications for the purpose of sharing information between devices without requiring a central network authority. VANETs communicate through *Dedicated Short Range Communications* (DSRC).

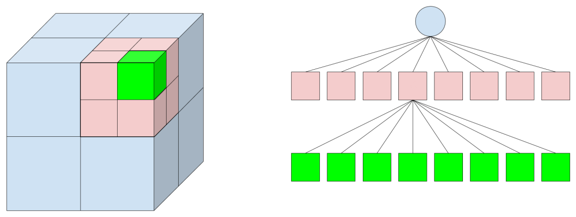

Figure 2: A visualized octree. Here, the largest cube (blue) is the parent for 8 child nodes (one colored red) which each contain their own children (the green cube.) A partially-expanded tree is also shown.

The specific technology for the wireless communications used for DSRC is largely unimportant and interchangeable, but is often described as using a variant of the IEEE 802.11 wireless standard, which is the standard on which WiFi technology is built [7].

For non-local (further than 1km or the maximum VANET range) communications or where a VANET is undesirable, it is possible that vehicles could access a remote network over existing cellular networks [1]. In such a system, vehicles would communicate with a remote server, and vehicle-to-vehicle communications are indirect. A few uses for remote communications schemes are discussed in Section 4.

2.3 Considerations

We believe there are a few points that need to be considered in the creation of an *Intelligent Transportation System* (ITS). First, vehicles must be able to act independently, as there will inevitably be periods when not all vehicles will be autonomous or when the vehicle is in a remote location and unable to communicate with other vehicles. Second, the system should be able to perform in multiple environments such as rural highways and busy urban roadways. This means the system must scale well, but should not *require* communication. Third, vehicles are often expected to have lifetimes far greater than other digital devices, as the high cost prohibits frequent replacement. This combined with the number of manufacturers means an ITS will need to be able to support a wide range of hardware and software platforms.

3. CARSPEAK

The first system we investigate is called CarSpeak, and is the product of a research project at MIT by Kumar, et al. [7]. CarSpeak is VANET communication system, implemented by the MIT group as an adaptation of the IEEE 802.11 specification for wireless transmissions. The core concept is that by improving the method in which data is transmitted between VANET members, CarSpeak allows a member vehicle to efficiently gain access to sensor data obtained by other cars in the vicinity of the vehicle as well as data from static supporting infrastructure. The vehicle can then use this additional data to alter its route and/or react to hazards outside of the vehicle's own sensor range, improving safety and responsiveness [7].

3.1 Octree Representation of 3D Space

In order to share their sensor information with other clients on the network, all vehicles using the system must have an agreed-upon naming scheme for their local region of 3-dimensional space. In the CarSpeak system, this is done using a globally defined set of octrees [7]. An octree is a data structure that uses a tree with a branching factor of 8 to reference a 3-D space. A given node N in the tree corresponds to a cube in 3-D space. The children then recursively represent 8 cubes which together form N. This process repeats until the desired level of detail is obtained; in CarSpeak, the smallest level of detail is a cubic meter. An octree node can also describe itself as *complete*, meaning that all sub-nodes contain identical values. This allows for more efficient data representation, detail is only as fine-grained as is needed. For example, a building occupies a large amount of space, but the details of the interior are not important for the vehicle. This combined with the shallow branching factor allows the CarSpeak system to significantly reduce the amount of sensor information being transmitted on the network, a necessity when the data generated by a sensor is large [7].

Suppose we are traveling on a highway. The cube from Figure 2 represents the downslope of a bridge ahead, where the left and right halves describe two lanes. We want to make sure the road is clear, so we make a request for the data from across the bridge. A car further up the bridge responds with the octree we see in Figure 2, and our vehicle determines that the blue cube (the left lane) is clear, but there appears to be an object in the road on the right. Our vehicle uses this information to preemptively change lanes without disturbing traffic.

3.2 Content-Centric MAC

CarSpeak's communication system is a modification to the 802.11 specification [7]. This technology is sufficient for the experimental work of the research group, though we note that 802.11 may not be an ideal candidate for a real-world implementation due to range limitations. However, the CarSpeak MAC could theoretically be implemented using an alternative communications protocol [2, 7].

CarSpeak differs the 802.11 standard through the use of a custom Medium Access Control (MAC) protocol, which allows for vehicle-to-vehicle communications without a central server or authority. The MAC also allows for improved data efficiency. An important requirement for CarSpeak is the ability for it to scale appropriately into situations where many vehicles are searching for data. In the 802.11 standard, connected devices compete with *each other* for network resources, and bandwidth is given proportionally based on the number of members on the network. For an autonomous vehicle system, this is an undesirable property since there are inevitably vehicles which provide redundant or unhelpful information. To improve upon this issue, the CarSpeak MAC makes the 3-D regions from the octrees the first-class citizen on the network, meaning that regions compete for network resources, rather than senders. Instead of all vehicles broadcasting all their data, vehicles send out requests for the data about the regions they care about, and the MAC assigns network resources and a designated sender to provide the information. The MAC monitors the number of requests for a particular region, and adjusts the transmission schedule based on demand [7]. So regions of "importance" to many vehicles are updated more frequently.

In addition to transmitting region data, the CarSpeak MAC can be used to transmit other types of information on the same medium, by assigning the data to a "region" which doesn't exist. This makes the technology extensible, and allows vehicles to share additional data with the resource balancing and scaling [7]. Vehicles might, for example, share localization information [5] or a vehicle's intent to change lanes in a congested environment. However, vehicles would need to know ahead of time that the fake regions exist.

3.3 Experimental Results

The MIT group implemented CarSpeak in two testing environments. The first was an indoor course, using small robots with laptops and infrared sensors placed in a densely populated environment. The second was an outdoor environment with an automated golf cart supported by robots strategically placed to provide information about obscured regions. For the tests, the researchers compared results under 3 communication schemes. First, a standard 802.11 implementation was used, where the raw 3-D point data was transmitted from all robots. Second, a *hybrid* 802.11 using the Octree region system, but without the content-centric network changes. And finally, a full implementation of the CarSpeak features [7]. For all experiments, the core programming for the robots was done using the Robot Operating System [9] and an existing path-finding solution outside the scope of this paper [7].

In the indoor testbed, experiments were performed to evaluate region contention, requests, resolution, path planning, and scaling of the three implementations. During evaluation, the robots were randomly placed throughout the testbed, and navigated to other random locations. Using the CarSpeak method, transmission rates for regions were more consistent than the other two communication methods, which is the region-contention at work. This means regions that are important will more reliably be available. Overall, the hybrid performed better than 802.11, but only "slightly" [7]. For region requests and scaling of robots, an expected increase in region data transmission was verified by experimental data. Similarly, they showed that the CarSpeak system was able to provide high levels of region detail without where the hybrid and standard 802.11 systems were overloaded with information. Finally, with increased numbers of transmitters (over 6) the path planning times were more than halved for CarSpeak.

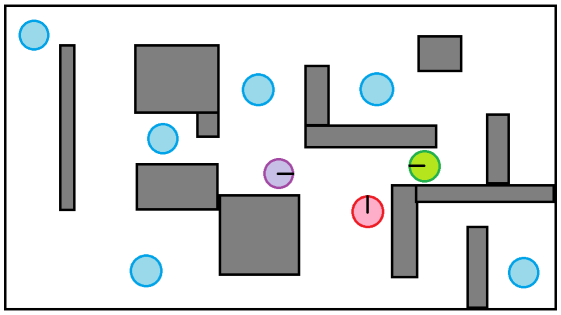

Figure 3: A hypothetical layout for the indoor testbed. Robots are placed throughout, and move to random destinations. Here, the purple robot can "help" the red and green robots avoid a collision [7].

The CarSpeak experiments also showed a significant increase in safety. On the outdoor course, the automated golf cart was given the task of avoiding a pedestrian entering the roadway from an obstructed location. The golf cart is supported by sensors placed near the entry point to provide remote data. When utilizing the remote sensors and the CarSpeak system, the golf cart was able to stop over 4 times faster than robot operating with only its own sensors [7].

3.4 Discussion

The CarSpeak experiments show the system is good at what it does, though the scale of the experiments is limited. Though the paper proposes that it is intended for low-speed urban environments where the vehicle has plenty of time to react/plan, if the wireless medium allowed enough range and throughput there is little reason the system should not perform adequately in rural or highway environments as well. However, we observe that the robot speeds in the experiments were slow (less than 5 miles/hour for the golf cart) and reaction times were not significantly faster than an average human's reaction speed, which may indicate that improvements would be required to ensure that the system is viable in higher speed environments.

4. OFFLOADING TO THE CLOUD

The next coordination scheme we will discuss is the use of computational and data offloading to remote cloud servers in order to allow for complex and platform-agnostic tasks. We will discuss the efforts of Kumar, Gollakota, and Katabi[1] [6] in using the cloud to store and communicate data, Ashok, et al. [1] in performing computation on the cloud, and Dressler, et al. [4] in using parked vehicles as a cloud.

4.1 Arguments for off-board computation

Modern vehicles contain various computers to assist in their function, controlling a range of components from the brakes to the radio. However, these *on-board units* (OBUs) are frequently limited in storage capacity or computational power, which will be necessary for an effective autonomous vehicle to be able to analyze the large data-streams generated by the sensors and from communication with other vehicles. Because the hardware available to a vehicle remains the same for its long (for computers, certainly) lifespan, it is advantageous to ensure that vehicles are adequately future-proofed [1]. We submit that this is especially relevant in the period of partial integration for autonomous vehicles where technologies are likely to rapidly change through research or government regulation. If we allow vehicles to offload computation to the more easily upgraded cloud, we can mitigate these limits to longevity.

Another argument for the use of cloud-based computing is the potential for persistent information. That is, the possibility that non-current data may be of use to a vehicle. A paper by Golestan, et al. [5] mentions the possibility that a vehicle could be able to improve localization by referencing its position relative to a known landmark. While it is possible that a car could store a private history of known landmarks, the flexibility provided by a shared knowledge base that multiple agents contribute to is likely more beneficial because it allows vehicles that are infrequent visitors

[1]This has multiple shared authors with the CarSpeak paper [7], and is the product of similar research.

to the area to benefit as well.

4.2 Offloading positional data

We will first take a look at moving data into the cloud. In [6], the MIT research group investigates using a cloud-based system called Carcel. Carcel was developed prior to the CarSpeak system, and the two share a few similarities. Primarily, Carcel utilizes the same Octree region naming scheme as CarSpeak. The researchers also used similar indoor and outdoor testing environments. Instead of CarSpeak's custom MAC, vehicles using Carcel randomly send region packets to the cloud server. The system assumes that through this randomness, regions are not likely to overlap and the server can create an approximately complete picture for a local region without requesting specific areas.

When tested, the Carcel system was able to decrease the robot response times by roughly 80% compared to the tests where vehicles operated independently. However, the response times for the robot are larger than those of the CarSpeak experiments [7, 6], which we attribute to the latency of communicating with the remote cloud server. Latency is an obvious drawback of offloading to the cloud, but does not entirely disqualify it from use. A cloud-based approach still allows access to more permanent data, as well as information about the world outside of the range of a VANET. While we hesitate to directly compare the performance of two separate experiments as we have done above, the experimental procedures for the CarSpeak and Carcel tests were identical [7, 6] other than the communication method.

4.3 Remote Application Services

Now we move on to the idea of offloading computation. The Carnegie Mellon group [1] proposes that the OBU in a vehicle will eventually cause a bottleneck in computation. The group draws a parallel to similar computational limits of mobile phones, laptops, and tablets, and emphasizes that the lifespan for those devices allows upgrades every few years. Slower mobile devices are quickly aged out as applications expand to consume the resource pool available in newer devices. Because vehicles have a much longer lifespan, it is important that the cloud services be designed in a way that allows them to be flexible in supporting a variety of platforms.

4.3.1 Task Offloading

The implementation of the remote execution server is relatively straightforward. The computation tasks performed by the vehicle are grouped into independent modules. The modules are then added to the server, and an API is created with which the vehicle can call the services as remote procedures. The system manages offloading using a software controller, which controls which (if any) tasks are performed on the server. The controller uses the abilities of the OBU and considers current task load and task priorities in order to determine which tasks are offloaded to the cloud [1]. If the OBU is not busy, there may not be an advantage to offloading. Or, newer software could be more intensive, and might be more likely to be run remotely.

To demonstrate their cloud offloading system, the authors created a prototype. In place of an vehicle's OBU, they substituted an Android smartphone, and an existing (non-local) remote server was used for the cloud. For the applications, the authors implemented a set of Java libraries for two vision processing applications, one to recognize user hand gestures and another for traffic signal recognition. They then brought the phone into a car and drove around while using their device, to determine the effects of the driving on latency and performance. While there was an increase in response time while moving, the times for the cloud computation were about 3x faster than the local execution. These times for the cloud execution included the network latency, and all responses were made within 250ms. While this may be a bottleneck considering that latency times cannot often be significantly improved, the authors note “At 60 mph a car displaces only about 5m in 250ms. Such response times seem practically reasonable for maintaining the interactivity requirement of these applications” [1]. Alternatively, we suggest that the cloud could be used to conduct less time-sensitive tasks.

4.4 Offloading onto Parked Cars

Dressler, et al. [4] examined the possibility of utilizing the large number of parked cars commonly found in urban and congested environments as a source of additional storage. While not exactly a cloud, the offloading aspect is present. The researchers explore offloading of data, but computation is performed by the vehicles during storage and communication, implying that computation could also be performed by the network. The parked vehicles could serve as an alternative to an expensive roadside support infrastructure that is likely to be needed in these urban areas. The researchers created a system that networks these parked vehicles, then used a simulation tool to evaulate the performance of their network.

4.4.1 Virtual Cords

To create the virtual cloud, the parked vehicles form *virtual cords*. Cords are a way for vehicles to pool their resources, which can then be used by nearby vehicles on the road. Members of a cord appear as a single unit for storage and computation to non-members. In order to establish a cord, nodes (parked vehicles) use *virtual coordinates* to obtain a “location” for themselves with respect to other parked vehicles. A vehicle's virtual coordinates are not based on geographical location, but rather describes the relation to neighbors that the vehicle has. A node establishes a connection to nearby vehicles by listening and populating a *neigh-*

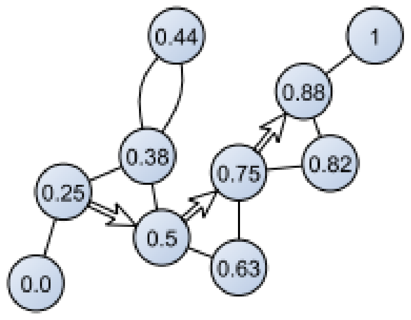

Figure 4: A visualization of a virtual cord [4]. Nodes are numbered based on an *order* within the cord, used in coordinating storage locations.

borhood table by receiving *hello* messages which are broadcasted from any nearby vehicles that are already members of a cord. The node will then attempt to join a cord, inserting itself into the cord based on its virtual coordinates. Using Figure 4 as an example, we can pretend that node 0.82 is not yet a member, and has no number. It's neighborhood table consists of 0.75 and 0.88. When it requests to join, it inserts as 0.82 This insertion works similarly to how a node would be added to the middle of a linked list.However, the virtual cord is actually a directed graph, and there is an internal path from node 0 to node 1. The graph structure allows vehicles that can only see one other cord member to join the network, such as node 0.44. This also means that the join process is lightweight, and does not require authorization from the entire cord. The virtual cord protocol also allows for cords to acknowledge the existence of other cords, allowing for communication through designated *gateway nodes* without combining the cords [4].

4.4.2 The Vehicular Cloud

The virtual cord allows for both a decentralized network similar to that of a VANET, and off-board data storage. Non-parked vehicles can interact with the nearby vehicular cloud using simple *publish* and *lookup* commands, which are handled by the vehicles in the virtual cord. Data is either routed to a node which can store it, at which point the cord *acknowledges* the submission, or the request is dropped.

The data stored in the vehicles is abstract, and addressed by an identifier hash. A moving vehicle can store any data, and is not limited by the software in the parked cars [4]. Unfortunately, the authors do not give specific examples for the kinds of data stored in the vehicular cloud. However, we speculate that this could include information about potential parking locations or data regarding nearby (but out of sight or VANET range) traffic conditions or road closings.

4.4.3 Simulation Results

Due to the scale required to adequately evaluate the author's protocol, a simulation was used to investigate the performance of the vehicular cloud. The first area the researchers investigated was whether or not the system would create useful cords, rather than dozens of small/singleton cords, which would each provide very limited capacity and thus be ineffective. Because the vehicles are parked, a vehicle is able to take its time when joining a cord, which allows for a vehicle to consider multiple cords, rather than joining the first available. Through their experiments, they found that the virtual cords protocol was able to connect 90% of the vehicles in the simulation to a cord within 2 minutes, which was deemed acceptable. They also found that the system scales with larger number of nodes, with cords increasing in average size (and thus effective capacity) rather than only the number of cords increasing. It was also demonstrated that the storage of data is viable in the system. Specifically, they found that data insertions and retrievals were completed within 1 second.

5. VANET SUPPORTED ROUTING

Darwish and Abu Bakar [3] have proposed a Lightweight Intersection-based Traffic Aware Routing (LITAR) protocol for use in urban environments. Traffic Aware Routing (TAR) protocols can take advantage of the real-time traffic information available through a VANET when making routing decisions. However, these protocols have limitations imposed by the network capabilities, or design decisions in existing systems. The protocol proposed by the authors aims to decrease network overhead and improve routing efficiency. We will first discuss types of TAR protocols, and then will move on to the LITAR proposal and simulation.

We would like to note that the Darwish and Bakar paper does not specifically discuss autonomous vehicles, though it does state that the protocol is intended for VANET communications [3]. However, we do not believe that there is a meaningful difference to be found in routing techniques used by human drivers or an autonomous vehicle. Frequently, human drivers already rely on computer-generated routes, and simply follow the instructions given by a GPS or mobile device.

5.1 Traffic Aware Routing Protocols

5.1.1 Full Path TAR

Full path TAR is a routing method that constructs a full navigation path from origin to destination. This full path is not dynamic, and once set the route is rarely modified. Additionally, most existing full path protocols do not adequately consider the current vehicle density for a stretch of road when making routing decisions, opting instead to evaluate the speed and throughput of vehicles. Because of this, they are sensitive to emergent traffic congestion found in high-density areas. The resulting lack of flexibility makes these protocols a poor choice for VANET routing, as they fail to take advantage of the information a real-time VANET can provide even if they can use real-time information in creating the initial route.

5.1.2 Intersection-Based TAR

Compared to full path TAR protocols, an intersection-based routing approach is more adaptable to real-time road environments. In these protocols, the route is calculated at each intersection based on the current traffic conditions, and only the information regarding the next step in a vehicle's route is shared with other vehicles. This makes it more suitable for use in a VANET environment. Unfortunately, there is a significant network overhead created by these protocols, as road metrics must be frequently collected and shared by a central *leader* vehicle (which will be discussed shortly) to ensure that the local information remains up to date.

5.2 LITAR Protocol

The LITAR protocol is (as the name implies) an adaptation of the intersection-based TAR format. The LITAR protocol uses two concurrent processes to manage routing. The first is an *RTNSM* (Real-time Traffic and Network Status Measurement) process, which uses the information provided by the VANET to determine the fitness of adjacent roads. The second is a packet routing process, which controls how data travels through the network. Data is transmitted across multiple hops, meaning a path is created from source to destination using cars in between to "forward" information as necessary.

In order to make informed decisions about routing, vehicles need to know what the traffic conditions are in the roads stemming of from the next intersection. However, due to range and bandwidth limitations, it is undesirable to have all vehicles share their location with all other vehicles. Instead, traffic metrics are aggregated for a stretch of road, and then transmitted to the vehicles in adjacent stretches. To create a complete picture of a given stretch, the LITAR protocol selects a *collector vehicle* near the geographic center. The collector is sent "collection packets" from other vehicles on the stretch, and creates a status packet containing the aggregate information describing the state of traffic in its section. This status packet is then forwarded to vehicles in adjacent stretches, which can use the information to reevaluate their path before arriving at the intersection.

However, this does not alleviated the network congestion concerns found in intersection based approaches. In order to combat this and improve the scaling of the system, the LITAR protocol implements a validity period algorithm. Collection packets are given a validity period based on the total number of vehicles on the road, the speed and direction of travel, and the volatility of traffic. In stable traffic environments, the collector vehicle receives fewer updates, and the data is considered to be valid for longer periods of time. In areas with lots of congestion or inconsistent numbers of vehicles, updates are calculated more frequently.

6. CONCLUSIONS

We believe that all the coordination schemes we have discussed here have their pros and cons. It is likely that an Intelligent Transport System will not be limited to a single coordination scheme, and we believe there is little reason that it should be. If the goals of coordination are to improve safety and efficiency, then it seems reasonable to continue improvements as long as is possible.

In Section 3, we saw how the use of a VANET to share region information between vehicles and roadside units can improve the safety of an autonomous vehicle. We also observed that the system can be used to improve vehicular routing in an urban environment. However, we found that there were limitations to the VANET in terms of range, which limit it in the macro scale.

In Section 5, we again see the use of a VANET for increasing efficiency. By analyzing the density of vehicles on the road, an autonomous vehicle can adapt its route during transit, which can allow for faster travel times. If combined with a system like CarSpeak, we propose that these benefits could be provided even if not all vehicles are autonomous, as sensors could allow a vehicle to estimate density for its stretch of road. Because the VANET itself is simply a communications platform, multiple systems can be created to run cooperatively, and there is not necessarily a need to select a system as the "winner."

The results of the Carnegie Mellon group [1] show that the use of a cloud system to offload computing is viable, even if the exact use cases may need to be carefully considered. The issue of latency is not insignificant, but for tasks that require large amounts of computational effort, the benefits may outweigh the potential issues. In Carcel [6] and the work of Dressler, et al. on the vehicular cloud, we see that a cloud-based communication scheme allow for vehicles to share data over longer distances than a VANET such as CarSpeak provides, but adds the requirement of a dedicated network and suffers from higher latency.

While not covered in this paper, we find it important to note that most of the coordination schemes we discuss are vulnerable to various attacks. We suggest [8] as additional reading on the topic for any interested parties.

Acknowledgements

I would like to extend many thanks to my advisor Nic McPhee, instructor Elena Machkasova, and external reviewer Scott Steffes for their time and helpful feedback.

7. REFERENCES

[1] A. Ashok, P. Steenkiste, and F. Bai. Enabling vehicular applications using cloud services through adaptive computation offloading. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, MCS '15, pages 1–7, New York, NY, USA, 2015. ACM.

[2] S. Aust, R. V. Prasad, and I. Niemegeers. Outdoor long-range WLANs: A lesson for ieee 802.11ah. *IEEE Communications Surveys & Tutorials*, July 2016.

[3] T. Darwish and K. Abu Bakar. Lightweight intersection-based traffic aware routing in urban vehicular networks. *Comput. Commun.*, 87(C):60–75, Aug. 2016.

[4] F. Dressler, P. Handle, and C. Sommer. Towards a vehicular cloud - using parked vehicles as a temporary network and storage infrastructure. In *Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities*, WiMobCity '14, pages 11–18, New York, NY, USA, 2014. ACM.

[5] K. Golestan, F. Sattar, F. Karray, M. Kamel, and S. Seifzadeh. Localization in Vehicular Ad Hoc Networks Using Data Fusion and V2V Communication. *Comput. Commun.*, 71(C):61–72, Nov. 2015.

[6] S. Kumar, S. Gollakota, and D. Katabi. A cloud-assisted design for autonomous driving. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 41–46, New York, NY, USA, 2012. ACM.

[7] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus. Carspeak: A content-centric network for autonomous driving. *SIGCOMM Comput. Commun. Rev.*, 42(4):259–270, Aug. 2012.

[8] A. Lima, F. Rocha, M. Völp, and P. Esteves-Veríssimo. Towards safe and secure autonomous and cooperative vehicle ecosystems. In *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, CPS-SPC '16, pages 59–70, New York, NY, USA, 2016. ACM.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[10] Tesla. Tesla - autopilot. https://www.tesla.com/autopilot/.