2008 IEEE Intelligent Vehicles Symposium
Eindhoven University of Technology
Eindhoven, The Netherlands, June 4-6, 2008

# Vehicle-to-Vehicle-to-Infrastructure (V2V2I) Intelligent Transportation System Architecture

Jeffrey Miller
Department of Computer Systems Engineering
University of Alaska, Anchorage
jmiller@uaa.alaska.edu

*Abstract* – **In this paper, I describe the vehicle-to-vehicle-to-infrastructure (V2V2I) architecture, which is a hybrid of the vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) architectures. The V2V2I architecture leverages the benefits of fast queries and responses from the V2I architecture, but with the advantage of a distributed architecture not having a single point-of-failure from the V2V architecture. In the V2V2I architecture, the transportation network is broken into zones in which a single vehicle is known as the Super Vehicle. Only Super Vehicles are able to communicate with the central infrastructure or with other Super Vehicles, and all other vehicles can only communicate with the Super Vehicle responsible for the zone in which they are currently traversing. I describe the Super Vehicle Detection (SVD) algorithm for how a vehicle can find or become a Super Vehicle of a zone and how Super Vehicles can aggregate the speed and location data from all of the vehicles within their zone to still ensure an accurate representation of the network. I perform an analysis using FreeSim to determine the trade-offs experienced based on the size and number of zones within a transportation network and describe the benefits of the V2V2I architecture over the pure V2I or V2V architectures.**

## I. INTRODUCTION

Much of the research in intelligent transportation systems (ITS) assume that vehicles will be able to communicate speed and location data to roadway infrastructure and to other vehicles. Two primary architectures have been proposed for these purposes – a vehicle-to-infrastructure (V2I) architecture and a vehicle-to-vehicle (V2V) architecture.

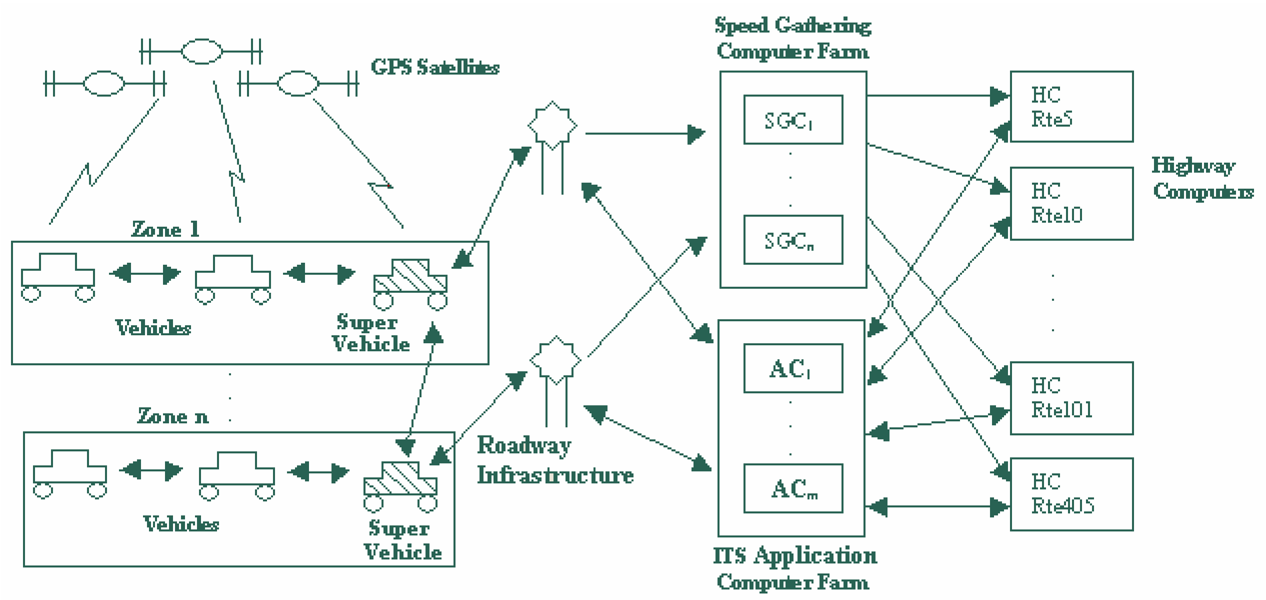
The V2I architecture allows vehicles to communicate with some roadway infrastructure to allow the speed and location of the vehicle to be transmitted to a central server. This server will maintain the speed and location of all vehicles and will aggregate this data for ITS applications, such as determining the fastest path from a vehicle's current location to its destination or identifying the location of an incident, among other applications. Taking the Los Angeles freeway system as an example, based on the California Department of Transportation's Annual Average Daily Traffic in 2003 [18], there could be potentially up to 1 million vehicles in the freeway system at any given time. With that many vehicles transmitting speed and location data simultaneously, the amount of data that is transmitted to the central server will exceed current wired or wireless bandwidth limitations. In addition, a central server has the limitation of being a single point-of-failure.

The V2V architecture, on the other hand, is quite fault-tolerant because of the highly-distributed nature of the network. As vehicles come into the network, they become nodes that communicate with other vehicles that are close in proximity to them. However, if a vehicle would like to know the fastest way to get from its current location to its desired destination, there is a substantial amount of data that must be transmitted from other vehicles. Queries must be sent to vehicles along all potential paths from the source to the destination, and speed and location data must be received by the requesting vehicle so that it can accurately determine the fastest path based on the real-time data. This is a substantial amount of data which is not currently capable of being transmitted based on current wireless bandwidth limitations.

The new architecture I am proposing is a hybrid of the V2I and V2V architectures, which is the vehicle-to-vehicle-to-infrastructure (V2V2I) architecture. In this architecture, vehicles still communicate with each other, similar to how they communicate in the V2V architecture. However, the road network is broken into zones, in which one vehicle is designated as a Super Vehicle. The size of the zones is configurable and dependent upon the bandwidth and accuracy desired by the application. The Super Vehicle will receive data from all of the other vehicles within its zone, aggregate the data, and then transmit the aggregated data to the central server. In addition, the Super Vehicle will transmit the data to the other Super Vehicles in adjacent zones. Queries for ITS applications can be sent to the central server, but if there is some sort of failure, the V2V architecture consisting of Super Vehicles can be used.

The remainder of the paper is organized as follows. In section II, I provide a description of the related work and the ITS architectures that exist. Section III gives an overall description of the V2V2I architecture, and section IV provides the Super Vehicle Detection (SVD) algorithm used by vehicles for discovering and becoming Super Vehicles and an algorithm for aggregating speed and location data to ensure an accurate representation of the network. Section V describes the benefits of the V2V2I architecture and shows the trade-offs experienced by the number of zones in the network, and the conclusion is given in section VI.

FIGURE I. V2V2I ARCHITECTURE




## II. RELATED WORK

Many papers have been written on vehicle-to-vehicle [2] and vehicle-to-infrastructure communication. As for V2V communication, secure mobile computing has been discussed in [4], and vehicle ad-hoc networks (VANETs) have been proposed in [18] and [19]. With V2I communication, much research has already been conducted utilizing the current data gathering methods of inductor loops and estimating the speeds based on the occupancy, number of vehicles, and average length of a vehicle [5]. Further, using this data to attempt routing of vehicles along fastest paths has been done in [6].

Many applications based on speed and location data from vehicles have also been proposed, such as incident identification, characterization of traffic flows [1], fastest path retrieval, and trip planning [3]. Many papers have been written on traffic prediction [7, 8], and many simulators exist that attempt to implement these and other ITS applications. A good overview of traffic simulators is presented in [9], and the work discussed in this paper utilized FreeSim [9-11] due to the fact that V2V and V2I communication are built into the framework, and FreeSim is open source, free, and easily extensible for other applications, including implementing the V2V2I architecture.

Representing a transportation network as a graph and determining fastest paths from one node to another has been discussed in [12]. Static graph algorithms, such as Dijkstra's [13], Bellman-Ford's [14, 15], and Johnson's [16] algorithms were extended to enable dynamic edge updates and constant queries by Demetrescu and Italiano in [17]. Miller and Horowitz added to the dynamic nature of the graph algorithms and customized the algorithms for ITS applications by utilizing a pre-processing step in [12].

Throughout all of these applications, it is assumed that the data is gathered either via a V2I or a V2V architecture. However, the feasibility of a hybrid of these two architectures has not yet been discussed. Combining these architectures and utilizing the benefits of both provides a new paradigm for ITS applications known as the V2V2I architecture.

## III. V2V2I ARCHITECTURE OVERVIEW

The vehicle-to-vehicle-to-infrastructure (V2V2I) architecture combines the advantages of both the vehicle-to-vehicle (V2V) and the vehicle-to-infrastructure (V2I) architectures, specifically the fault tolerant behavior of the V2V architecture and the fast queries and accuracy of the V2I architecture. A diagram of the V2V2I architecture is shown in Figure I.

In any ITS architecture, two types of applications must be supported: (1) gathering of speed and location data and (2) queries on this data. In the V2I, V2V, and V2V2I architectures, the transportation network is modeled as a graph. The weight of an edge in the graph represents the amount of time to traverse that edge based on the current speeds. Algorithms for dynamically updating the weights of the edges to enable fast query on the graph have been proposed by Demetrescu and Italiano [17] and Miller and Horowitz [12].

For gathering speeds and locations in the pure V2V architecture, all of the vehicles report their data to the other vehicles that are "close" to them, and those vehicles can choose to propagate this data to other vehicles. Vehicles that are "close" to another vehicle will receive a message sent over a wireless link from that vehicle. The query for a fastest path based on the current speeds would have to be propagated through the network to all of the vehicles along any potential fastest path, which would require a substantial amount of data. The number of vehicles that must be sent a fastest path query in a V2V network to ensure accurate routing can be infeasible at times. This problem can be ameliorated by the pure V2I architecture, in which all of the vehicles report their speed and location through some roadway infrastructure to a central server, which aggregates all of the speeds for each edge to enable responding to queries about the edges at a later time. When a vehicle wants to determine the fastest path from its current location to a desired destination, it queries the central server, which should have an accurate representation of the transportation system at all times. However, one limitation to the V2I architecture is that it contains a single point-of-failure, meaning that if the central server fails, or the link to the server fails, there is no way to retrieve any data. In addition, there is a large amount of data that must be received by the server when all of the vehicles are sending speed and location data, as well as querying for fastest paths (or other information). The V2V2I architecture attempts to reduce these limitations while leveraging the benefits of the V2V and V2I architectures.

In the V2V2I architecture, the transportation network is broken into zones, each of which consists of one or more edges. The zones are pre-configured, so each vehicle, as well as the central server, knows which edges are in which zones. Each zone has one vehicle, known as the Super Vehicle, that is responsible for communicating the speed data of the zone to the central server, as well as communicating this information to the Super Vehicles that are in adjacent zones. The size of the zone needs to be small enough such that two vehicles that are at the furthest points from each other within the zone can still communicate with each other. This is necessary in case one of those vehicles is the Super Vehicle that needs to get the data from all of the vehicles within the zone. In addition, the zones should be small enough so that the Super Vehicles of adjacent zones can communicate with each other. This will allow for the architecture to revert to a V2V structure in the event of a failure with the centralized components.

All of the vehicles within a zone will send their speed and location over a wireless link to the Super Vehicle of the zone. The Super Vehicle will aggregate this data and send the aggregated speed and location to the central server. Note that the Super Vehicle does not have to send only one speed and location to the central server. The data sent to the server, as well as the frequency of the data sent, can be configured based on the application needing the data. The aggregation algorithm used will determine how accurate the data will be at the central server when compared to the V2I architecture, which assumes every vehicle is transmitting its speed and location to the central server.

## IV. SUPER VEHICLE DETECTION ALGORITHM

Since vehicles are constantly traversing edges and zones in the network, the Super Vehicle of a zone will change frequently. The Super Vehicle Detection (SVD) algorithm can be used to find the Super Vehicle for the zone, or if one does not exist, create a Super Vehicle.

Initially when a vehicle enters a zone, it will make itself a Temporary Super Vehicle, which means it will act like a Super Vehicle and start aggregating speed data that it receives, but it will not respond to Find Super Vehicle messages. It will then send a Find Super Vehicle message for the zone to see if a Super Vehicle already exists. If it receives a response, it will remove its responsibility as a Temporary Super Vehicle. While it is waiting for a response (which may never come if a Super Vehicle does not exist), it may receive other Find Super Vehicle messages from other vehicles entering the zone. If a Temporary Super Vehicle receives a Find Super Vehicle message, it ignores it. After a random duration of time, the Temporary Super Vehicle will send a Temporary Find Super Vehicle message, which is a second try at finding a Super Vehicle. If another vehicle is a Super Vehicle or a Temporary Super Vehicle, it will respond with a Super Vehicle Response, which will make the sending vehicle remove the Temporary Super Vehicle responsibility from itself. If no vehicle responds after a short duration, the sending vehicle will make itself the Super Vehicle for the zone. When the Super Vehicle leaves the zone, it relieves itself of that responsibility, and a new vehicle that enters the zone will become the Super Vehicle by following the above algorithm.

To show that this algorithm will always find a Super Vehicle for a zone, two cases need to be considered: (1) when a vehicle enters a zone in which a Super Vehicle already exists, and (2) when a vehicle enters a zone in which a Super Vehicle does not already exist. Assume the following abbreviations – SV is a Super Vehicle, TSV is a Temporary Super Vehicle, V1 and V2 are vehicles, Z is a zone, FSV(Z) is a Find Super Vehicle message for zone Z, SVR(Z) is a Super Vehicle Response message for zone Z, and TFSV(Z) is a Temporary Find Super Vehicle message for zone Z.

For case 1, assume that V1 is a SV for zone Z, and V2 is entering the zone. Following the above algorithm, V2 will become a TSV as soon as it enters the zone Z and will send a FSV(Z) message. V2 will receive the FSV(Z) message and respond with a SVR(Z) message. Once V2 receives the SVR(Z) message, it will remove the TSV responsibility from itself.

For case 2, there are two scenarios that must be considered – (1) when a vehicle enters the zone by itself, and (2) when more than one vehicle enters the zone simultaneously. In scenario 1, V1 enters the zone Z and becomes a TSV. It will immediately send a FSV(Z) message, though it will receive no response. After a short wait, V1 will send a TFSV(Z) message, for which it will also not receive a response. After another small duration, V1 will become a SV for zone Z.

For scenario 2 in case 2, there is no SV for Z, but V1 and V2 both enter the zone at the same time. Since the vehicles are operating independently of each other, a race condition is possible and is avoided in the above algorithm. V1 and V2 both enter Z at the same time, so V1 and V2 both become TSVs. They both then send FSV(Z) messages. V1 receives the FSV(Z) from V2, and V2 receives the FSV(Z) from V1. Since V1 and V2 are both TSVs, they both drop the FSV(Z) messages. They will then wait a random amount of time before sending the next message. Assume that the amount of

TABLE I. V2V2I PERCENTAGE OF INCORRECT FASTEST PATHS BASED ON THE NUMBER OF EDGES PER ZONE

| # Edges / Zone | # Incorrect Fastest Paths | Total # of Paths | % Incorrect Fastest Paths |
|---|---|---|---|
| 1 | 1 | 30 | 3.3% |
| 2 | 4 | 30 | 13.3% |
| 3 | 12 | 30 | 40% |
| 4 | 28 | 30 | 93.3% |
| 5 | 28 | 30 | 93.3% |

time V1 waits is less than that of V2, so V1 then sends a TFSV(Z) message. V2 receives the TFSV(Z) message, and since it is a TSV for Z, it will become the SV for the zone and send the SVR(Z) back. Once V1 receives the SVR(Z), it will remove the TSV responsibility from itself.

In scenario 2 of case 2, there is a possibility that both V1 and V2 will wait for the same random amount of time before sending the TFSV(Z) message. In that case, both of the vehicles will transition from TSVs to SVs, and the zone will have two SVs until one of the vehicles leaves the zone. Although this will double the amount of data being sent to the central server from this zone during this time, it is a rare occurrence and does not affect the accuracy of the data transmitted.

## V. V2V2I ANALYSIS

Using the V2V2I architecture over the V2V and V2I architectures provides many benefits, including reducing the bandwidth requirement for the roadway infrastructure (which includes the central server), and allowing fault tolerance in the event of a hardware failure of one of the centralized components. However, reducing the bandwidth needed by the central server means that less data is being transmitted to it. In fact, instead of every vehicle transmitting its speed and location to the central server (as is the case with the V2I architecture), in the V2V2I architecture, only one vehicle per zone will transmit speed and location data. Even though the Super Vehicle in each zone can run its own aggregation algorithm and transmit as much data as it would like, I performed a feasibility analysis using FreeSim [9] assuming that only one speed was transmitted from each zone every second. The transportation network graph I used had 35 edges, with each edge connected to no more than three other edges. The transportation network used was from the northwest portion of District 7 of the California Department of Transportation, which includes the greater Los Angeles area. The nodes numbered 1-35 in Figure II represent the network used for the V2V2I analysis. I ran five tests based on the number of edges that were in each zone, from one edge per zone to five edges per zone. The edges were being constantly updated by live aggregated data gathered from the California Department of Transportation via inductor loop detectors. Every second, after all of the zones were updated by the Super Vehicles, I ran a fastest path query between two nodes. Using the pure V2I architecture as a baseline, Table I shows the percentage of paths that were incorrect based on the number of edges per zone.

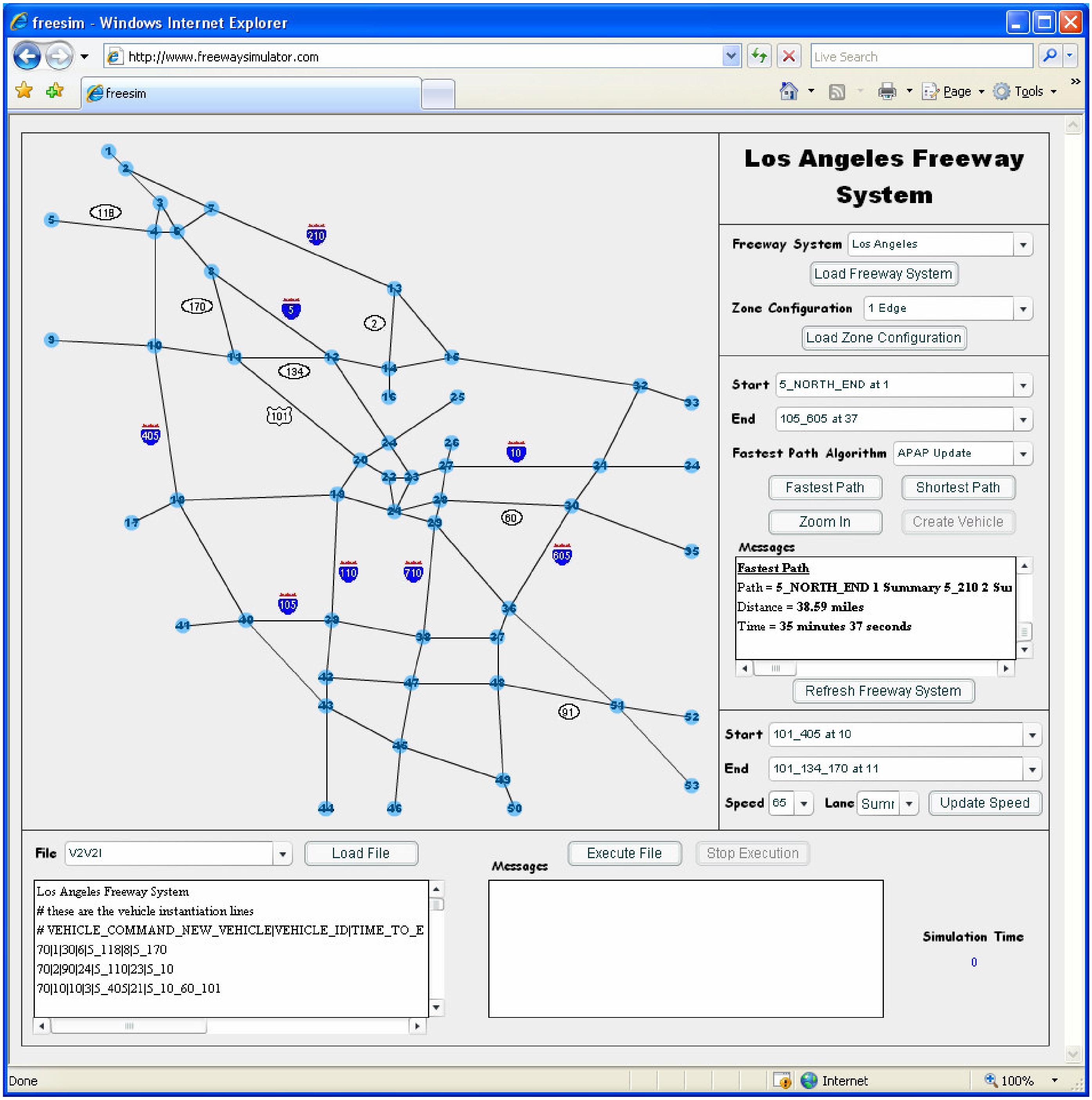
As can be seen from the table, only one fastest path was incorrect when there was one edge per zone as compared to the pure V2I architecture. With two edges per zone, only four out of 30 were incorrect, which is 13.3%. As more edges were incorporated into each zone, the percentage of incorrect paths jumped tremendously, with four or more edges per zone producing incorrect results almost 100% of the time.

In the section of the Los Angeles freeway system used for this analysis, there were an estimated 100,000 vehicles during the time period captured. Given 35 edges, there were approximately 2850 vehicles on each edge. With only one of the vehicles on each edge transmitting the speed and location to the central server each second, the amount of bandwidth saved was 1/2850. Assuming that 1 byte of data is needed to represent the speed, 8 bytes to represent the location, and 40 bytes of packet overhead, using the pure V2I architecture with every vehicle transmitting their speed and location every second would require 37.4 Mbps. Using the V2V2I architecture with one edge per zone, the required bandwidth for gathering speed and location for the central server is reduced to 13.4 Kbps.

## VI. CONCLUSION

In this paper, I presented a new ITS architecture called the vehicle-to-vehicle-to-infrastructure (V2V2I) architecture, which is a hybrid of the vehicle-to-vehicle (V2V) and the vehicle-to-infrastructure (V2I) architectures. The V2V architecture provides fault tolerance in a highly distributed environment, whereas the V2I architecture provides fast queries and accuracy given an abundance of speed and location data. The bandwidth requirement for the V2I architecture may make it unappealing, especially when bandwidth-intensive ITS applications, such as fastest path or traffic prediction algorithms, become more prevalent. Using Super Vehicles in the V2V2I architecture, the bandwidth requirement on the central server can be reduced by a factor proportional to the number of vehicles in each zone, while still retaining the accuracy of the V2I architecture. The algorithm for discovering and becoming a Super Vehicle of a zone was presented, and the entire architecture and analysis using live data gathered from the California Department of Transportation was simulated using FreeSim [9].

FIGURE II. FREESIM SCREENSHOT WHERE NODES 1-35 WERE USED FOR V2V2I ANALYSIS




## VII. REFERENCES


[1] Ni, Daiheng. "Determining Traffic-Flow Characteristics by Definition for Application in ITS." IEEE Transactions on Intelligent Transportation Systems, Volume 8, Number 2, June 2007.

[2] Blum, Jeremy, Azim Eskandarian. "A Reliable Link-Layer Protocol for Robust and Scalable Intervehicle Communications." IEEE Transactions on Intelligent Transportation Systems. Volume 8, Number 1, March 2007.

[3] Chen, Yanyan, Michael Bell, Klaus Bogenberger. "Reliable Pretrip Multipath Planning and Dynamic Adaptation for a Centralized Road Navigation System." IEEE Transactions on Intelligent Transportation Systems, Volume 8, Number 1, March 2007.

[4] Sklavos, Nicolas, Maire McLoone, Xinmiao Zhang. "MONET Special Issue on Next Generation Hardware Architectures for Secure Mobile Computing." Mobile Networks & Applications, Volume 12, Number 4, August 2007.

[5] Zhanfeng, Jia, Chao Chen, Ben Coifman, Pravin Varaiya. "The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors." *IEEE 4th International Intelligent Transportation Systems Conference*, February 12, 2001.
[6] Petty, Karl F., Peter Bickel, Jiming Jiang, Michael Ostland, John Rice, Ya'acov Ritov, Frederic Schoenberg. "Accurate Estimation of Travel Times from Single-Loop Detectors." Transportation Research, Part 1 (Policy and Practice), Volume 32A, #1, January 1998.
[7] Vlahogianni, Eleni, Matthew Karlaftis, John Golias, Nikolaos Kourbelis. "Pattern-Based Short-Term Urban Traffic Predictor." *IEEE 9th International Intelligent Transportation Systems Conference*, September 17-20, 2006.
[8] Quek, Chai, Michel Pasquier, Bernard Boon Seng Lim. "POP-TRAFFIC: A Novel Fuzzy Neural Approach to Road Traffic Analysis and Prediction." IEEE Transactions on Intelligent Transportation Systems, Volume 7, Number 2, June 2006.
[9] Miller, Jeffrey, Ellis Horowitz. "FreeSim – A V2V and V2R Freeway Traffic Simulator." *IEEE 3rd International Workshop on Vehicle-to-Vehicle Communication* in conjunction with *IEEE 3rd Intelligent Vehicle Symposium*, June 2007.
[10] Miller, Jeffrey, Ellis Horowitz. "FreeSim – A Free Real-Time Freeway Traffic Simulator." *IEEE 10th Intelligent Transportation Systems Conference*, October 2007.
[11] Miller, Jeffrey. "FreeSim – A Free Real-Time V2V and V2I Freeway Traffic Simulator." *IEEE Intelligent Transportation Systems Society Newsletter*, December 2007.
[12] Miller, Jeffrey, Ellis Horowitz. "Algorithms for Real-Time Gathering and Analysis of Continuous-Flow Traffic Data." *IEEE 9th International Intelligent Transportation Systems Conference*, September 17-20, 2006.
[13] Dijkstra, E.W. "A note on two problems in connexion with graphs." Numerische Mathematik, 1959.
[14] Bellman, Richard. "On a Routing Problem", Quarterly of Applied Mathematics. Volume 16, Issue 1, 1958.
[15] Ford Jr., Lestor R., D.R. Fulkerson. Flows in Networks. Princeton University Press, 1962.
[16] Johnson, Donald. "Efficient Algorithms for Shortest Paths in Sparse Networks." Journal of the ACM, 1977.
[17] Demetrescu, Camil, Giuseppe Italiano. "A New Approach to Dynamic All Pairs Shortest Paths." ACM Symposium on Theory of Computing, June 2003.
[18] CalTrans 2003 AADT – Average Annual Daily Traffic, 2003. http://www.dot.ca.gov/hq/traffops/saferesr/trafdata/.