An Efficient Vehicle Communication Network Topology with an Extensible Framework

Hazem M. Hajj*, Wassim El-Hajj**, Karim Y. Kabalan*, Mohamad M. El Dana*, Marwan Dakroub*, and Faysal Fawaz*

* The authors are with the ECE, American University of Beirut, Lebanon.
** The author is with the College of Information Technology, UAE University, UAE

Abstract

This paper proposes an extensible solution that gives support to recent growth in wireless communication and allows developers outside of the automotive industry to develop applications leveraging vehicle to vehicle communication. The suggested software architecture, which would integrate with the automotive open source architecture (AUTOSAR) standard, has primary focus on supporting applications for safer vehicles and also provides the foundation for developing personal and entertainment applications that can be operated within vehicles. The proposed architecture finds its applications in collision avoidance, autonomous navigation, and direct vehicle to vehicle communication for negotiating pass-through at intersections. The communication network manager is a critical part of the framework, and it includes a new Vehicle Ad-hoc Network (VANET) topology for communication with other vehicles. The proposed topology, that can have at most one intermediate hop to communicate any two vehicles, assumes conformance with IEEE 802.11p and the dedicated short range communication (DSRC) standard.

Keywords: *vehicle software, VANET, vehicle communication, software architecture, network topology.*

1. Introduction

The automotive industry continues its effort in making vehicles safer in response of people's concern [1] with incorporated technological innovations while maintaining competitive prices. Lately, there has been high interest in leveraging v2v technology for proposing new vehicle applications such as for identifying parking spots [2], traffic jams [3], driver assistance [4], and fleet management [5]. Focus on vehicle software dated back to the 1980s with initial focus on software engineering practices for automotive software [6-9]. Then focus shifted to advanced models and applications [10-11] where Schaefer proposed technologies for autonomous spacecraft with focus on uninhabited air vehicles. More recent efforts on vehicle software have shifted towards standardizing development under the AUTOSAR [12] open source framework standard for global cooperation in automotive software development with the goal of full proliferation to all new development through 2011.

On the vehicle communication and networking front, there has been an increased interest in this area which has led to the development of the 802.11p standard [13] for dedicated short range communication. There have also been several proposals for communication rules and networks such as the scheduling process suggested for blind intersection [14], the modeling of vehicles as a mobile ad-hoc network of clusters

[15], or the use of dedicated infrastructure to aid the v2v communication [16]. It was observed that vehicle networks have several challenges including delays [17] and losses associated with unpredictable mobility [18]. In brief, the choice of an optimal vehicle network continues to be a challenge, with some potential leverage of existing mobile ad-hoc network (MANET) topologies.

In this paper, a new extensible solution is proposed that comprehends new technologies such as v2v communication and GPS, and can allow developers outside of the automotive industry to develop applications for users' comfort, entertainment, luxury or safety using their vehicles. The extensible software framework requires elements to support communication and networking with other vehicles and fixed infrastructure. It should support ease of development to allow entertainment applications and connection with the world-wide-web. The architecture can integrate with AUTOSAR framework, consistent with the use of the 802.11p DSRC standard, include a proposal for an extensible VANET topology that provides fast connection to a cluster of vehicles and allows extensible connections to other vehicles or expansion of the cluster. The proposed extensible software architecture will be critical to quick implementation, providing a competitive advantage for car manufacturers. The proposed efficient communication topology, that does not depend on fixed infrastructure, will make it easy to implement in all countries, both developed and under-developed. The combination of both extensible software architecture and a simple network topology that could be implemented anywhere and enriched with new applications could make the area of intelligent vehicles the new technological revolution with impact to society comparable to the internet introduction.

2. Intelligent vehicle software Solution

This section presents a block based, callable, and grey-box [19-21] framework is proposed where some external interface components that could be readily available as black-box elements based on existing standards such as IEEE 802.11p or AUTOSAR. The suggested framework, shown in Figure 1, consists of 5 framework modules: The Intra-Vehicle Communication Module (ICM), the External Communication Module (ECM), the Analytics and Intelligence Module (AIM), the Virtual Bus (VB), and the Databases (DB). The ICM module handles communication within the framework and vehicle, the ECM cluster is responsible for all communication external to the vehicle, the AIM cluster is responsible for all analytics and intelligence required by the framework and the applications, the Priority Decision Maker for Vehicle Action reads all instructions sent by the applications from the Virtual Bus and decides which action or actions must be executed, the VB component, which consists of the Virtual Bus, acts as a bus where all communication between the framework blocks and the applications occurs.

In addition to the above mentioned framework components, the framework contains 5 databases. The Communication Metadata Configuration database contains information on the data formats required. The Measurement and Computational Database stores required information for the network topology, data from the Communication Data Manager, the AUTOSAR/Vehicle Data Management block, and from the Virtual Bus (data from the Analytics/Artificial Intelligence block and from applications). The Control and Data Configuration Database contains data on the framework's requirements for the data format and the vehicle's/AUTOSAR's requirements for the instruction format. The configuration database contains the configuration of the Analytics/Artificial Intelligence block, and the Knowledge Database stores the knowledge from that block.

New applications built on our framework require plug-in additions for custom logic. These are illustrated on the top of the framework in Figure 1. These contain logic specific to the application but can be easily extended to support special purpose vehicle applications for safety and driver assistance. Examples include software for negotiating pass-through at street intersection through direct vehicle to vehicle communication, collision avoidance driver-aid system, and GPS-based navigation assistance. For the case of vehicles negotiating pass-through at an intersection, the application would use the V2V/I2V/VANET Client block for transmission, and would rely on the V2V/I2V/VANET Server block for listening. The instructions to stop and go will be passed to the Priority Decision Maker for Vehicle Action which will decide on the action the vehicle will execute and forward the instruction for execution. For collision avoidance, data about sensor readings and driver actions are sent from the AUTOSAR/Vehicle Electronic Control Units to the Virtual Bus. The collision avoidance application then forwards the sensor readings to the Analytics/Artificial Intelligence block which will return a collision risk level to the application. When the application receives this assessment (the risk level) it will forward the corresponding required action to the Priority Decision Maker for Vehicle Action, which will take care of executing the command. The GPS navigation application asks the user for the destination and preferences (fastest route/avoid highways…), and then using the maps stored in it and the current vehicle position obtained from the GPS Module, it computes optimal routes leveraging the AIM analytics, and sends instructions through the Priority Decision Maker for Vehicle Action to ensure that the vehicle successfully follows the calculated route to its destination.

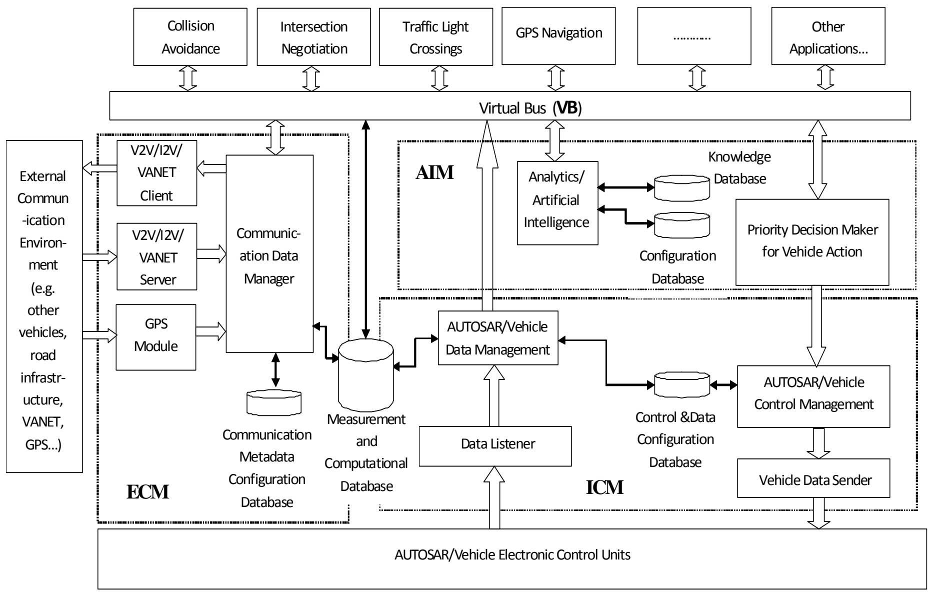
Collision Avoidance
Intersection Negotiation
Traffic Light Crossings
GPS Navigation
....
Other Applications...
Virtual Bus (VB)
External Commun-ication Environ-ment (e.g. other vehicles, road infrastr-ucture, VANET, GPS...)
V2V/I2V/VANET Client
V2V/I2V/VANET Server
GPS Module
Communic-ation Data Manager
Communication Metadata Configuration Database
Measurement and Computational Database
ECM
AIM
Analytics/Artificial Intelligence
Knowledge Database
Configuration Database
Priority Decision Maker for Vehicle Action
AUTOSAR/Vehicle Data Management
Data Listener
Control &Data Configuration Database
AUTOSAR/Vehicle Control Management
Vehicle Data Sender
ICM
AUTOSAR/Vehicle Electronic Control Units

Figure 1: Proposed framework architecture

To demonstrate the effectiveness of our proposed architecture, we implemented several components of the framework and we tested several applications using four robots to simulate vehicles. The implementation is based on multi-threaded components, publish-subscribe messaging, and a service-based architecture where components' actions are triggered based on incoming data or events. The primary driver for these choices is the need for fast communication between the components. The pseudo-code for the implemented software architecture is shown in Figure 2. The main program will run once the car is started. First (line 3), it will download the configuration of the framework and of the plugged-in applications. It then uses multi-threading to initiate the different blocks of the framework starting with the ICM (line 4), the ECM (line 5), the AIM (line 6), and then initiates the application specific plug-ins (line 7). Next, the program goes into a loop (line 8) that ends whenever the car is switched off. In this loop, the main program will read sensory data from the vehicle (Line 10) and external data from the ECM server (line 11), then publishes the collected data to all applications (lines 12 and 13). When plug-in applications act on the data, their responses are sent back through the virtual bus. From there, the data could be received by the Priority Decision Maker or by the ECM communication manager or both. These two components will pass on their actions and data to the internal vehicle controls through the ICM control manager (line 15) or to the external environment (line 16). In the next sections, we will cover the wireless (v2v and i2v) implementation, and the plug-in for vehicles negotiating pass through at intersections using v2v.

```
Pseudocode implementation for the main process of the framework
1  begin
2      \\ While car is turned on
3      Load_configuration_for_framework_and_applications;
4      Initiate_thread_for_ICM; \\ Initiates vehicle control manager and
       ICM data listener
5      Initiate_thread_for_ECM; \\Initiates Communication data
       manager, enables receiving and sending data using V2V
6      Initiate_thread_for_AIM; \\Initiates Priority decision maker for
       vehicle action
7      Initiate_plug_in_application_threads();\\all plugged-in applications
       such as collision avoidance
8      While(Car_is_on)
9      begin
10         Read_data_from_within_vehicle(); \\ If new data available, use
           ICM data listener
11         Read_data_from_external_sources; \\If new external data
           available, use ECM server
12         Publish_ICM_data(); \\Make new data available to virtual bus
           using ICM AUOTSAR/Vehicle data management
13         Publish_ECM_data(); \\Make new data available to virtual
           bus using ECM Communication data manager
14         Read_Priority_decision_maker_output(); \\receive commands
           from AIM module
15         Execute_Priority_decision_maker_output_To_vehicle(); \\use
           Vehicle control manager in ICM
16         Transmit_plug_in_specific_data_using_V2V; \\to communicate
           with v2v/i2v using ECM client
17     end
18 end
```

Figure 2: Pseudocode for the main branch of the framework.

To support vehicle to vehicle communication and infrastructure to vehicle communication, each vehicle must be able to send and receive data at the same time as shown in the ECM module. For this purpose, we designed a multi-threaded client-server application described below. The server's pseudo-code, as shown in Figure 3, will be called once the ECM module is initiated in the main program. At that point, the V2V network topology is activated.

```
Pseudocode implementation for ECM Server thread
begin
    Create buffer[ ...];
    struct server (hold server info: IP family, receive from any IP,
    port number);
    Create socket;
    bind(socket to server struct);
    buffer = recvfrom(...);
end
```

Figure 3: Pseudocode for the implementation of the ECM server.

The server process needs a dedicated socket to wait for incoming connections. To create the socket we first need a structure to save its information. The socket's information may have Address family (PF_INET for Internet family), IP address, and port number. The dedicated socket is created using the function *socket* which returns a positive integer value if successful. Once created, the socket must be bound to the address structure and if successful can receive incoming messages. We used the User Datagram Protocol (UDP) to eliminate the need for handshake between different vehicles, and achieve faster communication. Another alternative to the use of UDP would have been the Transport Control Protocol (TCP), which is characterized by its reliability. TCP has built-in functionalities that ensure the delivery of transmitted data packets. This feature however will reduce communication speed, since TCP has to perform a handshake procedure before transmitting any data which adds overhead. We tested the TCP implementation which required in addition to the UDP steps described above, to listen for incoming connection and to accept the connection before receiving any messages. In our testing, UDP turned out to be more beneficial for two reasons. First, handshake was not needed to send and receive which means the protocol was faster. Second, UDP supports message broadcasts, needed to support many applications.

Finally, the plug-in needed to implement an application to illustrate the efficiency of the framework is implemented. We use the scenario of vehicles arriving at an intersection and negotiating directly for safe passage. The intersection is detected using i2v communication by the ECM module shown in Figure 4. The pseudo-code initiates a thread to detect the presence of an intersection (line 4), and once detected the application ensures the safe crossing of vehicles. An intersection process will run to negotiate (line 5) pass through using v2v communication. The car will then generate a decision (line 6) to cross using AIM intelligence. The vehicle will finally communicate (line 7) its decisions to other vehicles and will make sure there are no conflicts with the decisions of the other vehicles. If a conflict exists, the plug-in will repeat the communication. Otherwise, it proceeds with publishing (line 8) the data to the virtual bus, and from there to move the car through ICM controls. This example shows how little logic is implemented to allow a new application allowing safe vehicle passage and leveraging v2v and i2v technologies. Lines 4-6 are the only logic additions to the framework for the plug-in.

```
Pseudocode implementation for Intersection plug_in thread
begin
    While(1)
    begin
        if_intersection_detetcted; \\ detecting i2v signal through ECM
        Communicate_with_other_vehicles; \\ through ECM client
        Make_decision_based_on_Rules_or_AI;\\ use AIM if needed
        Communicate_decision_to_other_vehicles; \\ Ensures that there
        is no conflicts in decision, will repeat communication if needed
        Publish_decision_to_virtual_bus; \\ Which in turn gets passed
        as a decision to the vehicle by main() program
    end
end
```

Figure 4: Pseudocode for the implementation of an application plug-in for vehicles negotiating at an intersection.

3. Efficient vehicle communication network topology

In the proposed communication network topology managed by the ECM, each moving vehicle equipped with the extensible framework would form exactly an instance of the long envisioned mobile ad hoc networks [22]. Through the ECM, each vehicle constitutes a local communication area around itself, enabling it to exchange vital signs with the neighboring vehicles. Such inter-vehicle communication should be characterized with efficiency, resiliency, and fast response. In what follows, a high tolerant VANET topology is proposed that guarantees the existence of a maximum path of one intermediate hop between any pair of vehicles. The proposed topology is based on an idea used to obtain mutual exclusion of events in distributed environments [23].

3.1 Network design

The VANET will be specified by a graph *G = (V, E)*, where *V* is the set of vehicles (nodes) and *E* is the set of links between vehicles. In our proposed VANET design, each vehicle $v_i \in V, 1 \le i \le N$ in the network is assigned a communication set (C) satisfying the 4 constraints listed below. Each vehicle is then connected directly to the vehicles in its communication set establishing the desired topology.

1. $\forall i : 1 \le i \le N \rightarrow v_i \in C_i$
2. $\forall v_i, \forall v_j : i \ne j, 1 \le i.j \le N \rightarrow C_i \cap C_j \ne \phi$
3. $\forall i : 1 \le i \le N \rightarrow |C_i| = K$
4. Any vehicle v_i is contained in *K* number of C_js, $1 \le i, j \le N$.

Where N is the total number of vehicles in the network. Ideally, we want *only one* common vehicle between any two communication sets since this vehicle will be used as an intermediate hop when making the routing decisions. Otherwise, the complexity of the routing increases because a choice has to be made on which intermediate hop to use for routing. To achieve the best case scenario, it can be shown [23] that the following equation should be satisfied: $N = K(K-1)+1$. Once the above constraints are satisfied, the VANET topology is constructed by connecting each vehicle with all vehicles in its communication set. Figure 5 shows an example of our design where seven vehicles coexist on the road i.e. *N=7* and *K=3*.

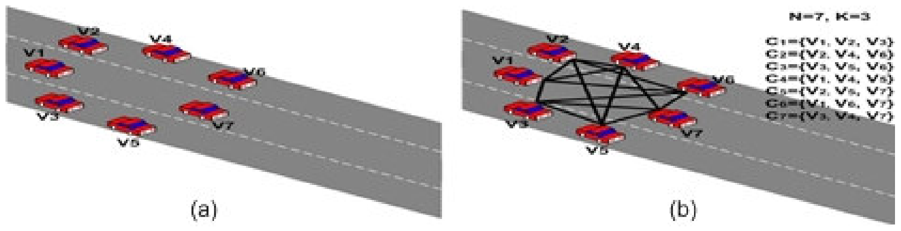

Figure 5: VANET topology when 7 vehicles exist

Constraint 1 states that each vehicle should belong to its own communication set i.e. vehicle v_2 is part of communication set C_2. Constraint 2 states that there is at least one common vehicle between the communication sets of any two vehicles. For example, in figure 5, communication sets C_2 and C_7 have vehicle v_4 in common. Constraint 3 makes sure that the size of each communication set is equal to K (in Figure 5, each communication set is of size 3). In the ideal case, this means that all vehicles have the same degree. This condition implies that all vehicles have an equal number of tasks and thus the capability of achieving a similar throughput. Constraints 3 and 4 are desirable features of the design, which can be relaxed with no major impact on the overall network topology characteristics.

As will be discussed later, a VANET topology satisfying the above 4 constraints has very desirable network features in the routing and storage efficiency (Figure 5). Our goal then becomes to create a VANET topology satisfying the above constraints; this translates to finding an appropriate communication set C_i for each vehicle i such that constraints 1 trough 4 are satisfied. We show in the next section that routing between vehicles becomes very efficient since any two vehicles will either communicate directly with each other or a third vehicle will exist to connect them. Also very little storage space will be needed to manage the VANET routing. In addition, we describe the algorithm that finds the communication sets in a particular VANET.

3.2 Communication set construction

Using the theory in [23], it can be shown that the communication sets exist only when $K+1$ is the power of a prime number. So, it is not possible to generate communication sets when $K+1$ is not a power of a prime number. Communication sets for such values of K can be generated using another systematic technique. The topology in figure 5 has $N = 7$ and satisfies the first condition since $(K+1) = (3+1) = 2^2$ is power of a prime. The algorithm in Figure 6 is used to calculate the communication sets under this condition, and the illustration is shown in Figure 7. Figure 7 presents graphically the 4 major steps (modules) needed to generate the Communication Sets. It considers a network of seven nodes ($N=7$, $K=3$). Figure 7-(a) reflects the execution of lines 2 though 8 of the algorithm presented in figure 6. This module only initializes a $(K-1)\times(K-1)$ matrix with the appropriate values. It also generates the initial communication set (line 8). The time complexity of this module is $O(K^2)$. Figure 7-(b) and figure 7-(c) reflect the execution of lines 9 through 14 and 15 through 20, respectively. They are used to generate all the communication sets originating from the matrix rows and columns. Each of these modules has also a time complexity of $O(K^2)$. Figure 7-(d) reflects the execution of lines 21 though 31. This module considers all the communication sets generated from the matrix diagonals. An offset is initialed to zero and is incremented by 1 each time a new set is generated. The time complexity of this module is $O(K^3)$. Consequently the complexity of the whole algorithm is $O(K^3) \approx O(N^{\frac{3}{2}})$. The resulting topology is shown in figure 5(b).

```
   Data: $N$, $K$
   Result: Communication Sets
 1 begin
 2     Create a $(K-1) \times (K-1)$ Matrix $M$
       /* Matrix Initialization                        */
 3     $counter = 0$
 4     for $i$=0 → $(K-2)$ do
 5         for $j$=0 → $(K-2)$ do
 6             $M[i][j] = (K+1) + counter$
 7             $counter = counter + 1$
 8     $S = \{1, 2, \cdots, K\}$ forms the first set
       /* 1 combined with all Matrix rows              */
 9     for $r$=0 → $(K-2)$ do
10         $S = \phi$
11         for $c$=0 → $(K-2)$ do
12             $S = S \cup M[r][c]$
13         Prepend 1 to $S$
14         Let S be one of the sets
       /* 2 combined with all Matrix columns           */
15     for $c$=0 → $(K-2)$ do
16         $S = \phi$
17         for $r$=0 → $(K-2)$ do
18             $S = S \cup M[r][c]$
19         Prepend 2 to $S$
20         Let S be one of the sets
       /* 3 → $K$ combined with all Matrix diagonals.
          The variable $jump$ acts as an offset.       */
21     for $jump = 1 \rightarrow (K-2)$ do
22         for $i$=0 → $(K-2)$ do
23             $S = \phi$
24             $r = 0$
25             $c = i$
26             for $j$=0 → $(K-2)$ do
27                 $S = S \cup M[r][c]$
28                 $r = r + 1$
29                 $c = (c + jump)\ mod\ (K-1)$
30             Prepend $(jump + 2)$ to $S$
31             Let S be one of the sets
       /* At the end of this algorithm, $K(K-1)+1$
          sets are generated.                          */
32 end
```

Figure 6: Generating the communication sets when *K+1* is power of a prime number

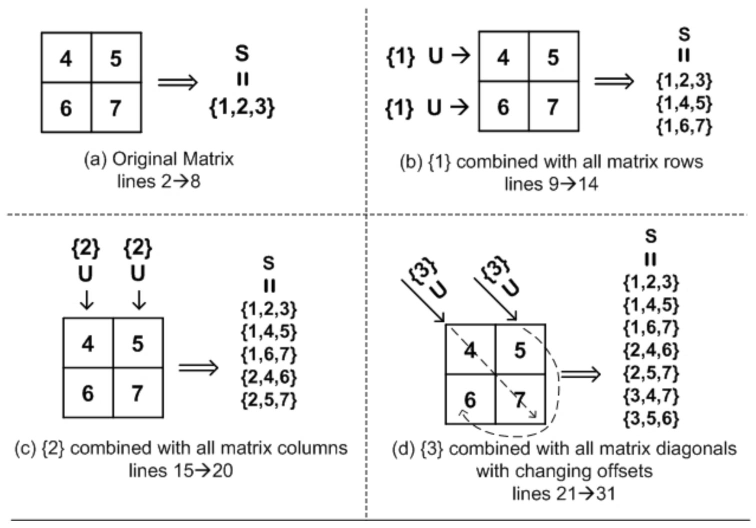
4 5 6 7
S = {1,2,3}
(a) Original Matrix lines 2→8
{1} U → 4 5
{1} U → 6 7
S = {1,2,3} {1,4,5} {1,6,7}
(b) {1} combined with all matrix rows lines 9→14
{2} U ↓ {2} U ↓
S = {1,2,3} {1,4,5} {1,6,7} {2,4,6} {2,5,7}
(c) {2} combined with all matrix columns lines 15→20
{3} U {3} U
S = {1,2,3} {1,4,5} {1,6,7} {2,4,6} {2,5,7} {3,4,7} {3,5,6}
(d) {3} combined with all matrix diagonals with changing offsets lines 21→31

Figure 7: Detailed Graphical example of the Communication Sets' generation algorithm (figure 6) when *N=7, K=3*

The algorithm in Figure 6 can be extended to calculate the communication sets when *(K + 1)* is *not* the power of a prime. This extension is discussed in details in [24]. The algorithms discussed in figure 6 when *K+1* is a power of a prime and in [24] when *K+1* is not a power of a prime are only used to generate the communication sets. Then each generated communication set has to be associated with a particular vehicle. This association is achieved using the algorithm presented in Figure 8.

```
Data: Communication Sets, $M$, $K$
Result: Vehicle Association
begin
    Step 1: Vehicle 1 is associated with the set
    costructed from $\{1, 2, \cdots, K\}$
    Step 2: Vehicle 2 is associated with the set
    costructed from vehicle 2 and the first column in
    matrix $M$
    Step 3: A communication set constructed from a
    row is associated with the $2^{nd}$ vehicle in the set.
    Step 4: A communication set constructed from a
    column is associated with the $2^{nd}$ vehicle in the set.
    Step 5: A communication set constructed from a
    diagonal $jump$ is associated with the $(jump + 2)$
    vehicle in the set.
end
```

Figure 8: This algorithm associates each vehicle to a particular Communication Set

The algorithm takes as input the communication sets and outputs the appropriate assignments. The complexity of the algorithm is *O(N)* because the assignment stage is performed in constant time *O(1)* for each vehicle. For a network size *N=7*, the algorithm is illustrated as follows:

- Step 1: Vehicle 1 (v_1) is associated with communication set $\{v_1, v_2, v_3\}$ i.e.

$C_1=\{v_1, v_2, v_3\}$.
- Step 2: v_2 is associated with communication set $\{v_2, v_4, v_6\}$ i.e. $C_2=\{v_2, v_4, v_6\}$.
- Step 3: v_4 is associated with communication set $\{v_1, v_4, v_5\}$ and v_6 is associated with communication set $\{v_1, v_6, v_7\}$ i.e. $C_4=\{v_1, v_4, v_5\}$ and $C_6=\{v_1, v_6, v_7\}$.
- Step 4: v_5 is associated with communication set $\{v_2, v_5, v_7\}$ i.e. $C_5=.\{v_2, v_5, v_7\}$.
- Step 5: v_7 is associated with communication set $\{v_3, v_4, v_7\}$ and v_3 is associated with communication set $\{v_3, v_5, v_6\}$ i.e. $C_7=\{v_3, v_4, v_7\}$and $C_3=\{v_3, v_5, v_6\}$

The same reasoning can be followed for any value of N when calculating the communication sets and assigning them. It is important to note that, in the example just described, C_1 through C_7 satisfy the 4 constraints discussed in section 3.1.

3.3 Distributed routing between vehicles

The main purpose behind designing the network according to section 3.1 constraints is to increase the routing efficiency in terms of minimizing delay and increasing resiliency. Recall that given a VANET, we first calculate the communication sets and then associate each communication set with a particular vehicle. Our topology is then constructed by connecting each vehicle with its corresponding communication set. As a consequence, if a vehicle needs to communicate with another vehicle located in its communication set, a direct connection is present between them. However, if a vehicle needs to establish a path to a destination vehicle not in its communication set, only one intermediate hop is needed to establish the path. In both cases, the maximum path size between any source and destination pair is composed of two hops. This can be illustrated through the following example. Given a VANET of size $N = 13$, the communication sets of vehicles v_1 and v_6 can be calculated (using the algorithm in figure 6) to be: $C_1=\{v_1, v_2, v_3, v_4\}$ and $C_6=\{v_2, v_6, v_9, v_{12}\}$. In this case, v_1 can directly establish communication with vehicles v_1, v_2, and v_3. However in order to establish a communication path from vehicle v_1 to vehicle v_6, vehicle v_1 needs to go through the intermediate vehicle v_2, which represents the intersection of communication sets C_1 and C_6. This design ensures a minimum routing path size between any 2 vehicles.

The only requirement for the success of this routing strategy is to make sure that each vehicle knows its communication set and the communication set of any other vehicle in the network. This can be achieved by executing the algorithms in figures 6 and 8.

3.4 Evaluation of the proposed vehicle network topology

When a vehicle is moving on a certain road, there might exist multiple vehicles in its communication range. Regardless of the number of vehicles, our proposed scheme has the following advantages:

- Resilient and fault tolerant network topology: this property came from the fact that each vehicle in the proposed topology has at least K alternative paths from itself to any other vehicle in the network (constraint 3). If any of these paths got broken for any reason, another route can be calculated to the destination vehicle.
- Efficient and fast routing protocol between vehicles since any two vehicles are

directly connected or a third vehicle exists to connect them i.e. you can communicate between any two vehicles using a maximum of 2 hops.

- Low storage space. Traditionally routing protocols store routing tables on each node in the form of an $N \times N$ matrix; these tables become very large as the network grows. In our approach, no routing tables are needed at all as routing decisions are made on demand. To improve performance, a small cache can be maintained on each node to store the already calculated communication sets.

In comparison with a fully connected VANET, the proposed VANET achieves higher savings in the required number of connections. In fact, the number of connections in a completely connected network is [$N(N-1)/2$] while the number of connections in the described topology is: $N(K-1)$. This is due to the fact that every vehicle is connected to the $(K-1)$ vehicles in its communication set. The above represents a gain ratio of $(\sqrt{N}+1)/2$. Moreover, the number of hops used in a completely connected network is $N(N-1)$ because each node can reach every other node using one hop. In the suggested topology each node can communicate with other nodes by either 1 or 2 hops. So, the total number of hops is $2N(\sqrt{N}-1)^2$. The above constitutes a loss ratio of $2(\sqrt{N}-1)/(\sqrt{N}+1)$.

In addition, the quality of a network Q is inversely proportional to the product of the node degree (d), the network diameter (D), and the number of links (L). In a completely connected network, the network quality is given by $Q=2/N(N-1)^2$ where for the suggested network, it is found to be $Q=1/8N(\sqrt{N}-1)^2$.

4. Conclusion

This paper presented an extensible and scalable software framework for quick turn-around on vehicle applications. The framework supports developing intelligent solutions with capabilities of intra-vehicle and external to vehicle wireless communication. An efficient VANET topology was also introduced that allows inter-vehicle communication with no more than one intermediate hop for end to end communication. Examples of special-purpose applications for vehicle safety were provided to demonstrate the effectiveness of the framework. The efficiency of the proposed VANET topology was supported with quantitative measures. The plug-in nature of the software makes it attractive to developers. The solutions can interface with state-of-the-art standards such as AUTOSAR vehicle software and the DSRC wireless communication standard. The proposed solutions are not just stand-alone concepts, but they are rather based on real scenarios that can be easily integrated with currently manufactured vehicles, and at low cost.

References

1. UNECE (United Nations Economic Commission for Europe) Transport Division, Statistics of Road Traffic Accidents in Europe and North America, 2007 edition. http://www.unece.org/trans/main/wp6/transstatpub.html
2. S. Miura, Z. Yi, T. Kuroda, "Evaluation of parking search using sensor network," in 1st International on Wireless Pervasive Computing, 2006, pp. 6.
3. K. Ohara, Y. Nojima, H. Ishibuchi, "A study on traffic information sharing through inter-vehicle communication," in IEEE 22nd Internation Symposium on Intelligent Control, 2007, pp. 670-675.

4. K. Ch. Fuerstenberg, U. Lages, "New european approach for intersection safety- the EC-project intersafe," in Proceedings of the Intelligent Vehicle Symposium, 2005, pp. 177-180.
5. R. L. Sabounghi, "Intelligent vehicle highway system- the universal close-range road/vehicle communication system concept-the enhanced AVI and its CVO applications," in Vehicle Navigation and Information Systems Conference, 1991, pp. 957-967.
6. M. D. Grover, "A uniform object/process model for autonomous vehicle component communication" in Proceedings of the IEEE International Symposium on Intelligent Control, 1988, pp. 113-117.
7. B. Overton, I. Spalding, M. Thomas, "Issues in the validation and verification of vehicle software," in Eighth International Conference on Automotive Electronics, 1991, pp. 98-101.
8. D. Ward, "Guidelines for development of automotive software," Software Engineering Journal, vol. 11, pp. 76-81, March 1996.
9. M. L. Nelson, "A Design Pattern for Autonomous Vehicle Software Control Architectures," in Computer Software and Applications Conference, 1999, pp. 172-177.
10. P. Schaefer, R. D. Colgren, R. J. Abbott, H. Park, A. Fijany, F. Fishjer, M. L. James, S. Chien, R. Mackey, M. Zak, T. L. Johnson, S. F. Bush, "Technologies for reliable autonomous control (TRAC) of UAVs," in Proceedings of Digital Avionics System Conference, 2000, pp. 1E3/1 - 1E3/7
11. D. Earle, D. Wallis, R. Wenham, "Future architecture and design trends for automotive control systems," in The 29th Annual Conference of the IEEE, 2003, pp. 2847-2852.
12. Joergen Moessinger, "AUTOSAR: The standard for global cooperation in automotive software development", ATI 2008, Tokyo Jaan.
13. D. Jiang, L. Delgrossi, "IEEE 802.11p: towards and international standard for wireless access in vehicular," in IEEE Vehicular Technology Conference, 2008, pp.2036-2040.
14. L. Li, W. Fei-Yue, "Cooperative driving at blind crossings using intervehicle communication," IEEE Trans. Vehicular Technology, vol. 55, pp. 1712 - 1724, Nov. 2006.
15. Y. Lui, U. Ozaguner, "Effect of inter-vehicle communication on rear-end collision avoidance," in Proceedings of the IEEE Intelligent Vehicle Symposium, 2003, pp. 168-173.
16. K. Tsukamoto, H. Nakata, M. Fujii, M. Itami, K. Itoh, "A study on collision avoidance in the system that integrates IVC and RVC," in Proceeding of the Intelligent Vehicles Symposium, 2003, pp. 1-5.
17. T. Sadayuki, K. Shit, "Evaluation of incident information transmission on highways over inter-vehicle communications," in Proceedings of the Intelligent Vehicle Symposium, 2003, pp. 12-16.
18. J. J. Blum, A. Eskandarian, L. J. Hoffman, "Challenges of intervehicle ad hoc networks," IEEE Trans. Intelligent Transportation System, vol. 5, pp. 374-351, Dec. 2004.
19. ME Fayad, DC Schmidt, RE Johnson, "Building application frameworks: object-oriented foundations of framework design, - 1999", John Wiley & Sons, Inc. New York, NY, USA
20. S. Sparks, K. Benner, C. Faris, "Managing Object-Oriented Framework Reuse Computer, Los Alamitos 1996, vol. 29, number 9, pp. 52-62.
21. H. Züllighoven, R. F. Beeger, Object-oriented Construction Handbook: Developing Application-oriented Software with the Tools & Materials Approach, Illustrated, Elsevier, 2004.
22. C. Perkins, Ed., Ad hoc networking, Addison-Wesley, 2001.
23. M. Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," ACM Trans. Computer Systems, May 1985.
24. W. El-Hajj, H. Hajj, and Z. Trabelsi, "On Fault Tolerant Ad Hoc Network Design", International Wireless Communications and Mobile Computing Conference (IWCMC'09), Leipzig, Germany, 21-24 June 2009.