Vehicular Address Configuration

Maria Fazio*, Shirshanka Das†, Claudio E. Palazzi†‡, Mario Gerla†

*Dipartimento di Matematica

Università di Messina, 98166, Messina, Italy

e-mail:mfazio@ingegneria.unime.it

† Computer Science Department

University of California, Los Angeles, CA 90095

e-mail:{shanky|cpalazzi|gerla}@cs.ucla.edu

‡ Dipartimento di Scienze dell'Informazione

Università di Bologna, 40126, Bologna, Italy

Abstract

Vehicular networks are characterized by unique properties and challenging problems. The high mobility and density of nodes impede the direct utilization of traditional techniques and protocols in this context. In particular, automatic IP address configuration in vehicular ad-hoc networks is a challenging and as yet unexplored issue. The importance of this problem cannot be overemphasized since any application that involves communication between two or more nodes requires the presence of unique identifiers to deliver the data to the right destination. Due to the very high mobility of vehicles, traditional mechanisms to automatically associate an IP address to a wireless node fail to perform well. To solve this problem, we propose a novel scheme that exploits the topology of vehicular ad-hoc networks and an enhanced DHCP service with dynamically elected leaders to guarantee reliable and fast address configuration. We provide extensive simulation results that demonstrate the efficacy of our scheme.

Keywords: Ad-hoc networks, VANET, address configuration.

I. Introduction

In the field of *inter-vehicular communication* (IVC) the emerging trend is to equip vehicles with communication capabilities in order to directly exchange information within a short communication range (i.e., few hundreds of meters). Indeed, a plethora of appealing services emerges that involves vehicles located in proximity of each other, such as navigation safety, online games, and data/file sharing.

A possible vehicular scenario is represented by Internet Gateways (IGs), or hotspots, located at intermittent intervals along the roadway providing high-bandwidth connectivity within a limited range.

Since their fast mobility, cars are in the range of each IG for a very limited amount of time. Depending solely on the IGs for connectivity will hence lead to an intermittent and unsatisfactory Internet experience. Ad-hoc networks have been proposed to extend the communication range of vehicles. Hybrid scenarios arise when we consider cooperative networking applications where the vehicular ad-hoc network (VANET) complements the statically connected Internet [7]. Across all scenarios fast and efficient network configuration of the cars is needed to minimize the time spent in control overhead.

In this context, a very important topic, yet never investigated in VANETs, is represented by the address configuration. To allow effective communication in a network, each node needs to be identified through a unique and automatically assigned address. Aiming at integrating VANETs within the Internet and providing passengers with any kind of Internet applications, the IP address represents the natural identifier in the system.

Existing VANET literature bypasses the issue of node configuration by assuming that nodes are configured *a priori*. However, this issue cannot be skipped so easily since neither address autoconfiguration protocols for traditional fixed networks nor solutions proposed for classic ad-hoc networks can be directly applied to VANETs [8][15][14][12][16][3]. VANETs have unique characteristics that require a specific analysis of the problem [11]: very high mobility, theoretically infinite extension, absence of a centralized control, and intermittent connectivity through the sparse infrastructure.

In this paper we highlight properties featuring the address configuration problem in VANETs. We then propose and evaluate through experiments a novel automatic IP address configuration protocol named Vehicular Address Configuration (VAC), which is specifically designed for this scenario. In particular, VAC exploits the topology of a VANET and a distributed DHCP protocol run by dynamically elected leader-vehicles to quickly provide unique identifiers and minimize the frequency of IP address re-configurations due to mobility.

This paper is organized as follows. In Section II we review solutions for address configuration in classic ad-hoc networks. We describe prominent issues in addressing configuration with reference to VANETs in Section III. In Section IV we analyze VANET properties to provide guidelines for address configuration protocol design. Section V describes our solution for automatic IP address configuration in VANETs. Simulation evaluation of the protocol performance is presented in Section VI. Finally, Section VII concludes this paper.

II. Related Work

In spite of significant research effort in the general area of auto-configuration, an efficient solution for IP address configuration in VANETs is still missing. Researchers have addressed this issue with reference to generic ad-hoc networks, without considering VANET topology and mobility model. We claim that the unique characteristics of a VANET should be taken into account when designing the address configuration protocol. At the same time, previous techniques for ad-hoc networks represent a fundamental background and should be used as a starting point. Therefore, in this section, we present address configuration approaches in ad-hoc networks categorizing them into three groups: decentralized, best-effort, and Leader-based.

In the decentralized approach the auto-configuration problem is solved in a distributed fashion. A node that needs an address makes a request to the network and receives the configuration parameters through its interaction with other nodes. Even though this is the simplest solution in ad-hoc networks, it could generate large amounts of signaling traffic in VANETs with a high density of nodes.

An example of a decentralized approach is represented by MANETConf [8]. With this approach, a node (*Requester*) entering the network asks a configured node (*Initiator*) to negotiate for an address on its behalf. The Initiator sends a query for a specific address throughout the network and then waits for a positive acknowledgment from all known nodes. Each node in the network maintains a list of addresses which are currently in use or that are going to be assigned to new nodes. This allows nodes to be aware of the available addresses at any time. However, distributing the information to all the nodes in the network implies a high volume of traffic in large networks, as well as complexity for keeping the information updated and consistent.

In [6] the authors present a decentralized mechanism based on the buddy data structure. Every node has a disjoint set of IP addresses that can be assigned to new nodes. The system does not need to employ a Duplicate Address Detection (DAD) procedure to discover and manage late address duplications; instead, it uses a proactive approach where each node periodically broadcasts its IP address table. Obviously, in large and dynamic networks this generates high overhead.

Best-effort solutions do not ensure that every address is unique in the network: their main goal is just that of guaranteeing the correct routing of packets. If users with duplicated addresses do not communicate with each other, there is no need to waste time and resources in solving the duplication. However, problems occur if nodes with duplicate addresses start to communicate. In this case, transmissions have to be delayed until the conflict is solved and this leads to poor performance especially for real-time

applications.

Another best-effort approach is presented in [16] where nodes assign addresses to themselves by using a probabilistic algorithm in order to minimize the address duplication probability. Then, information from ongoing routing protocol traffic is used to check if duplications occur.

Leader-based approaches make use of a hierarchical structure to perform the address configuration procedure. In [14], Leaders maintain a list of all addresses in use in the network, which is identified by a network identifier (NID). A node wishing to join the network relies on an *agent* (a node already in the network) to send a request to the Leader. The reply from the Leader passes through the agent and arrives to the node. In the network there is only one Leader that proactively and periodically floods a beacon throughout the whole network. The merging of two networks can be detected through the presence of more than one Leader. A node receiving a beacon from a foreign Leader informs his current Leader of the presence of the foreign one. Then, the two Leaders communicate each other their lists of currently used addresses in order to discover duplicate addresses.

Communication between the Leaders can be complicated by node mobility, which can break the communication path, and by the fact that such path cannot be guaranteed to be free of duplicate addresses. Specific mechanisms have hence to be introduced. In [12] an elected Address Authority (AA) maintains the state information of the network. To perform the address assignment, nodes send requests for a candidate address to the AA and register this address if no rejection is received. The AA periodically broadcasts Network Identifier Advertisement (NIA) messages over the network so that partitions and merges can be detected and handled. However, since a VANET can be extremely wide and populated, even this scheme could fail or be too slow when employed within this type of network.

III. Problem Statement: Dynamic Addressing in Vehicular Scenario

A VANET differs from usual ad-hoc networks in its vehicular environment, node distribution and movement. Even the applications run are often specifically devised to be more useful in this setting. The technical considerations in designing a VANET should reflect these features. In our work we have considered the VANET environment depicted in Figure 1.

Vehicles are equipped with the Dedicated Short-Range Communication (DSRC) and have about the same computational and transmission capabilities. They move along a freeway and are able to communicate either directly or by using intermediate vehicles as relaying nodes. A specific ad-hoc routing protocol is used to support multi-hop packet delivery among nodes in the network. The most common routing protocols (e.g., AODV [10][9] or DSR [5]) make use of IP addresses to identify nodes and to perform

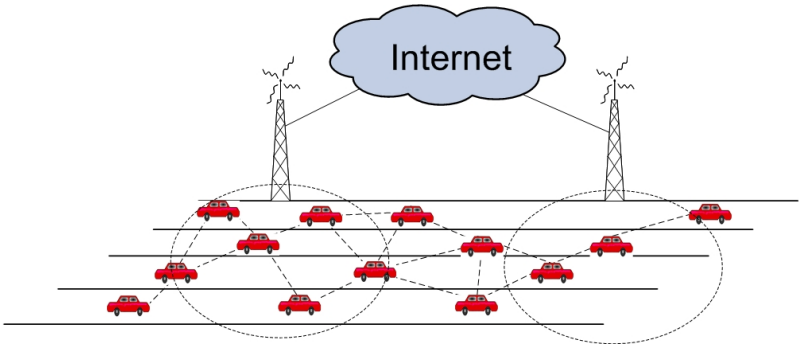


Fig. 1. Vehicular scenario

correct packet routing.

IGs can be spread along the freeway in order to provide Internet connectivity to vehicles; however, due to their mobility, vehicles pass fixed IGs in few seconds. Consequently, the IGs provide a cheap, but also intermittent and time-constrained, access to the Internet for transiting vehicles.

From the point of view of the address configuration issue, IGs cannot work as DHCP servers to assign dynamic IP addresses to vehicles. In infrastructure mode, whenever a car X passes through the communication range of an IG, it can utilize the dynamic IP address IP_X provided by that IG. The same address IP_X could be assigned again as soon as X goes out of the range of the IG. This implies that only vehicles within the coverage of an IG have unique identifiers. On the other hand, nodes that work in ad-hoc mode cannot make use of such identifiers, because the VANET expands outside the coverage of single IGs. A possible solution could be represented by a modified multi-hop DHCP that simultaneously involves several IGs and extends the DHCP service to n-hop faraway nodes. However, this approach raises other issues like IG discovery, IG information synchronization, coverage width (IGs can be more than 10 miles apart), and delays. Moreover, frequent handoffs among IGs due to high and fast mobility lead to costly IP address changes. For all these reasons, the address management system cannot reside on the fixed infrastructure along the roadside. To support ad-hoc IVC, not only a specific protocol for address management should be designed, but it should also be in the form of a distributed scheme for mobile nodes.

To perform both ad-hoc communication among vehicles and infrastructure-based communication through IGs, orthogonal channels can be used by communication devices endowed with two interfaces, one for the ad-hoc and one for the infrastructure mode. A two radio interfaces solution exploiting orthogonal

channels to minimize interferences is both practical and feasible. Indeed, multiple commercial vendors are coming out with multi-band chipsets that allow communication on two or more channels [2].

Applications in ad-hoc networks are expected to be based on service discovery and data sharing mechanisms. Users take advantage from such services and data at their convenience and usually this involves communications among users few hops away. These assumptions remain true when we focus on VANETs. For instance, navigation safety applications generate alert messages which are exchanged among cars close to each others. Indeed, most of the current routing protocols for ad-hoc networks do not support routing between faraway users; for instance, AODV limits broadcast for the Route Discovery procedure to 35 hops. Only geographic routing protocols [4] would be able to work with very distant nodes but such protocols are not suitable in VANETs because nodes have high speeds and very variable positions.

Summarizing, an IP address configuration protocol must assign unique identifiers to nodes that join the network (e.g., vehicles that enter a freeway). Since the spread of a VANET is theoretically infinite, nodes located very far from each other can utilize the same identifier. However, there is the possibility that these nodes eventually communicate. This could happen, for instance, in the case where two vehicles that are using safety ensuring application get closer to each other. In this situation, at least one of them has to change address in order to avoid mistakes in packet delivery. Therefore, the configuration protocol's work does not finish after the initial assignment of addresses to all nodes in the network. Rather, the two main tasks that a configuration protocol has to perform in ad-hoc networks are: 1) the initial address configuration and 2) the Duplicate Address Detection (DAD) procedure.

IV. System Model

As a result of each node being a vehicle, certain traditional concerns with mobile nodes in ad-hoc networks, such as power constraints, are no longer of primary importance. On the other hand, other issues become crucial. For instance, the proper functioning of navigation safety applications requires very low latency.

The design of a protocol for auto-configuration of IP addresses in VANETs is closely bound to the system properties. Therefore, we are first going to highlight VANETs' features with reference to the development of the configuration protocol.

DSRC is a short to medium range wireless communication technology that operates in the $5.9GHz$ range and provides $1000m$ as maximum radio coverage. Under these assumptions, the width of freeways becomes negligible with respect to the communication range. We can hence assume that the topology

of the network is linear when focusing our attention on vehicles that flow along one-way street. Since ad-hoc communication usually involves close nodes that are interested in sharing the same information or in accessing similar services, we reduce our case study to a group of nodes that move following a track. These nodes share the same group movement until they arrive at a switch point. A switch point is the intersection point of streets and freeways at which groups can split into smaller ones or merge into bigger ones; moreover, each group can lose or gain elements. Elements belonging to the same group have an internal mobility with respect to each other. Considering that vehicles on a freeway have speeds between $50 - 70mph$ ($22.35 - 31.29m/s$), the relative velocity among the vast majority of them can be assumed in the range $3 - 20mph$ ($1.34 - 8.94m/s$).

The environment in which nodes move is organized in road segments that cross each other, merge, or split into two or more branches. Even if we can establish starting and ending points for each segment, it is impossible to settle boundaries for the whole system. As already mentioned, VANETs have a potentially infinite extension.

Focusing on the density of nodes in the network, if we consider r as the communication range of nodes, v as the mean speed of nodes, and Δt_i as the inter-arrival time of vehicles in the VANET, then the mean number of neighbors nb for each node in a linear topology can be expressed by: $nb = \frac{2r}{v\Delta t_i}$. Since the assumed speed range of $50 - 70mph$ for vehicles on a freeway, we can set $v = 60mph$ ($26.82m/s$). The inter-arrival time of vehicles Δt_i varies with the traffic on the freeway, and it is bound to largely variable conditions such as geographic location and rush hours. Nonetheless, we can use the same assumptions stated in [7] and consider a Poisson distribution with mean inter-arrival time in the range of $0.5 - 4s$. According to these values, the mean number of neighbors nb for each node can be estimated in $10 - 100$, which is consistent with parameters used for simulation analysis in [11].

Summarizing, the main properties of the scenario we are considering are:

- linear topology of the network;
- theoretically infinite extension of the network;
- low relative speed, ($3 - 20mph$);
- high density of nodes ($\Delta t_i = 0.5 - 4s \Rightarrow nb = 10 - 100$).

V. Vehicular Address Configuration Protocol

VAC represents the first protocol for IP address configuration in VANETs. We designed VAC aiming at guaranteeing a reliable address configuration service for both the initial configuration of nodes that join the network and for the later verification of duplicates through the DAD task. VAC is fully compatible with

whichever application or service that involves the communication among vehicles located in proximity of each others and minimizes the configuration time and the generated signaling overload.

VAC is a Leader-based protocol. The Leader-based protocols for ad-hoc networks discussed in Section II are not suitable for VANETs for several reasons. The most important one is that ad-hoc solutions have been designed for networks with limited extension where every Leader needs to be aware of the presence of all other Leaders in order to synchronize information. Moreover, each Leader is responsible for a sub-network identified by a NetID (Network Identifier). The NetID is usually randomly selected and represents a limit to the size of the system. If there are too many NetID, the assumption that random NetIDs are always different could be invalidated. VANETs are networks with theoretically infinite size, thus, synchronizing information among all the Leaders and determining unique NetID along the network becomes impossible.

VAC organizes Leaders in a connected chain so as to have every node in the communication range of at least one Leader. The basic idea of Leader chain comes from the linear topology of a VANET and it is shown in Figure 2, where bigger nodes are Leaders whereas smaller ones are normal nodes that rely on Leaders for configuring their IP addresses. The hierarchical organization of the network allows limiting the signal overhead for the address management tasks. Only Leaders communicate each others and maintain updated information on configured addresses in the network. Normal nodes ask Leaders for a valid IP address whenever they need to be configured. The communication that involves normal nodes is based on local broadcast. Each normal node communicates only with Leaders in its transmission range, thus sensibly reducing flooding and multi-hop interactions among nodes in the network. Moreover normal nodes snoop Leaders' packets in order to catch information they need to maintain their IP addresses.

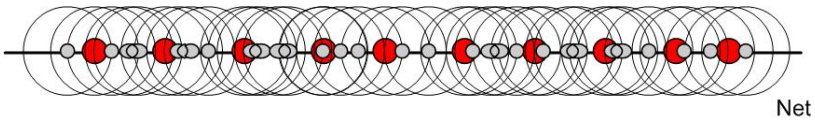


Fig. 2. Leader and normal nodes: network organization

Leaders act as servers of a distributed DHCP protocol. Each of them manages a different subset of possible addresses ($addrSet$) to serve requests for address assignment coming from normal nodes located within their communication range. The configuration of nodes is very fast because it does not suffer from delay in information discovery or in delivering of configuration parameters along the network.

As explained in Section III, the ad hoc protocols and vehicular applications act in a limited area.

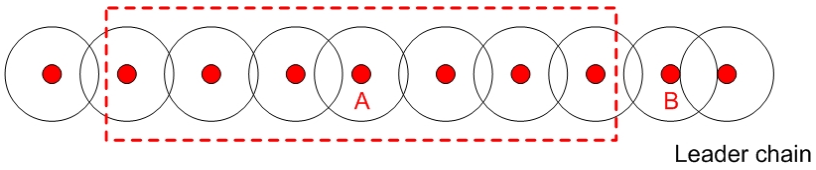


Fig. 3. Leader chain

Likewise VAC guarantees unique IP addresses within a delimited area around each Leader, called *SCOPE*. To elaborate, the *SCOPE* of Leader A is the set of Leaders whose distance from A is less or equal to $scope$ hops. In Figure 3 we depict the *SCOPE* for Leader A considering $scope = 3$. Let consider the normal node Y that received the *IPy* address from A. *IPy* will be unique as long as Y moves within the *SCOPE* of A. If Y goes out of the *SCOPE* of A, in order to still ensure address uniqueness, Y has to ask for another address to the new Leader (for instance, to Leader B in Figure 3). Considering that the relative speed between nodes is low, changes in the address configuration due to having left the own Leader's $SCOPE$ are not frequent.

This represents an important contribution of our work since, in any case, these changes would be much more frequent if nodes (i.e., travelling cars) had to rely on fixed IGs to obtain their IP addresses. To this aim, Figure 4 shows the duration of an IP address from when it is assigned to a node to when the node needs to be reconfigured. Three cases are analytically compared: i) a car travelling through the coverage area of a fixed IG at 60mph, ii) a car travelling through the coverage area (i.e., the *SCOPE*) of a Leader implementing VAC with $scope = 0$, and iii) a car travelling through the coverage area of Leaders implementing VAC with $scope = 4$. For all the compared cases we considered 400m as the transmission range, while in cases ii) and iii) the leaders were located at regular intervals of 200m and various relative speeds were tested. Obviously, case i) presents a constant outcome (30s), while cases ii) and iii) are able to ensure much higher stability to the address configuration.

VAC functionalities can be grouped into two main tasks:

A. Building and maintenance of the Leader chain: how to elect Leaders in the network and how to change them when node mobility makes it necessary;

B. Address configuration and maintenance: management of addresses that can be assigned to nodes in the network.

In the next subsections we describe how VAC performs tasks *A.* and *B.* with more details.

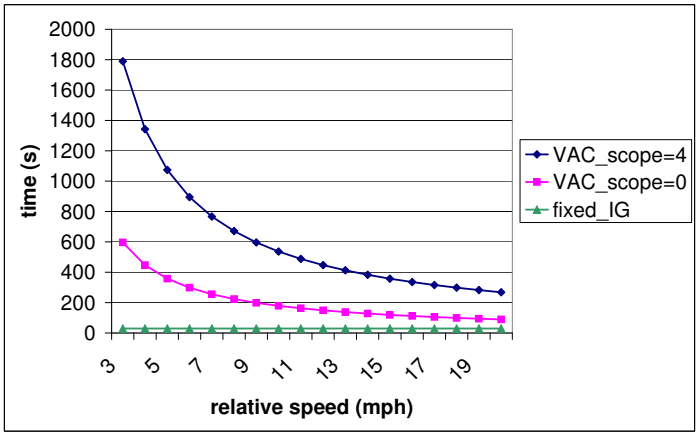


Fig. 4. Address validity time.

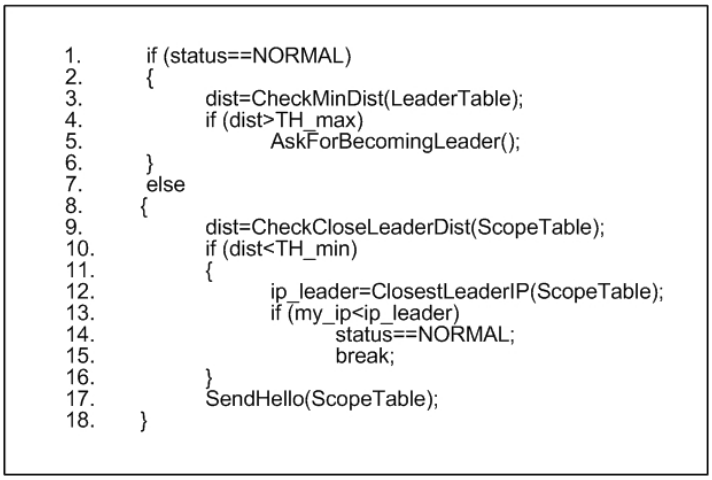
A. Building and Maintenance of the Leader Chain

The metric used for building the Leader chain is the distance among nodes. In VANETs, vehicles are equipped with on-board communication systems that support ad hoc connectivity. Such equipments could be endowed with GPS and that would represent a useful resource to support Leader chain building. Otherwise, the distance between two nodes can be easily estimated through the radio signal strength as vehicles are arranged along a line.

If the distance between two Leaders L1 and L2 exceeds a threshold *TH_max*, a node in the middle has to become a new Leader. We are sure that there is at least a node that can become Leader for the scenario of freeways is characterized by high density of nodes. On the contrary, if the distance falls under a threshold *TH_min* one between L1 and L2 will become Normal.

This geometric construction of the chain has been specifically designed for the freeways scenario that we are considering, but it can be extended for the construction of a Leader grid in other scenarios where vehicles move along several directions. Indeed, it is solely based on the relative distance among Leaders and is independent from the location of nodes in the environment.

The pseudocode in Figure 5 describes the operations that are performed by nodes to build and to maintain the Leader chain. We have distinguished the behavior of normal nodes (lines 1- 6) from the behavior of Leaders (lines 7- 18). Periodically, a normal node estimates the distance between the Leader



```
1.    if (status==NORMAL)
2.    {
3.        dist=CheckMinDist(LeaderTable);
4.        if (dist>TH_max)
5.            AskForBecomingLeader();
6.    }
7.    else
8.    {
9.        dist=CheckCloseLeaderDist(ScopeTable);
10.       if (dist<TH_min)
11.       {
12.           ip_leader=ClosestLeaderIP(ScopeTable);
13.           if (my_ip<ip_leader)
14.               status==NORMAL;
15.               break;
16.       }
17.       SendHello(ScopeTable);
18.   }
```

Fig. 5. Pseudocode for Leader chain building and maintainance

in front and the one behind (line 3) itself. If such distance is higher than the TH_max threshold (line 4), the node asks to become a new Leader (line 5). Such request will be managed from the Leaders in the area in order to select just one among all the normal nodes that detect the condition.

Periodically, a Leader estimates the distance between itself and the closest among the other Leaders in the *SCOPE* (line 9). If such distance is lower than the TH_min threshold (line 10), the Leader compares its IP address with the address of the close Leader (line 13), so that the Leader with the higher address will become normal (line 14). Then the Leader sends a *Hello* packet to update the information on Leaders in its *SCOPE* (line 17).

B. Address Configuration and Maintenance

This task is composed by three main components:

1) synchronization of address information among Leaders;

2) modified DHCP protocol to assign addresses to nodes that make a request;

3) DAD procedure to verify whether an address in the *SCOPE* ceases to be unique due to node mobility.

These components operates as follows.

1) Synchronization of address information: to carry out the address configuration of nodes, Leaders have to know which addresses are available to be assigned. For this purpose, the address space is divided in subsets ($addrSet$) and each Leader distributes addresses taking them from one of these

subsets. Obviously, Leaders within the same *SCOPE* have to manage different $addrSets$. The size of the $addrSet$ depends on the width of the *SCOPE*. A big *SCOPE* (that corresponds to a high value for $scope$) implies small $addrSets$. However, $addrSets$ cannot be excessively small, as Leaders have to be able to guarantee addresses to all the nodes in the VANET. The $addrSet$ dimension can be calculated as $SpaceADDR/(2scope + 1)$, where the $SpaceADDR$ is the addressing space for VANETs. If nb is the mean number of neighbors for each node, it is necessary that:

$$SpaceADDR > (2scope + 1)nb \tag{1}$$

These settings meet the hypothesis of local communication among users in the VANET. As shown in Section IV, in the worst case nb is equal to about 100. Under these conditions $(2scope + 1)nb < 5000$. If we use for VANETs IPv4 addresses of class B (e.g. 192.168.x.x), there are 65.535 possible addresses and the disequation (1) is largely satisfied.

The design of VAC is based on a proactive approach to synchronize information on the $addrSet$ among Leaders in a *SCOPE*. Periodically, each Leader sends in broadcast an *Hello* packet holding the subset of addresses it can assign and the subsets of all the Leaders in its *SCOPE*. Whenever a Leader receives an *Hello* packet, it updates its view of the Leader in its *SCOPE*. For example, Figure 6 depicts a Leader chain portion. The dotted rectangle circumscribes the *SCOPE* of the Leader A, meaning that Leaders

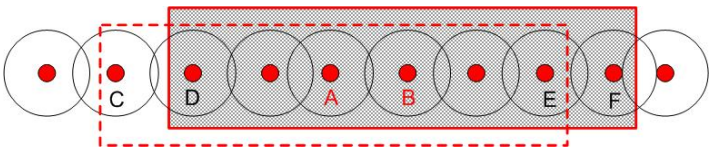


Fig. 6. Information synchronization among Leaders

between C and E manage different subsets of address $addrSets$. In the same way the filled rectangle circumscribes the *SCOPE* of the Leader B. A sends an *Hello* packet, containing information on itself and on the other Leaders in its *SCOPE*; amongst the others, the packet holds information on Leaders C and D. When B receives this *Hello* packet, it uploads the information on D because D belongs to its *SCOPE* but discards information on C. In fact, B is only responsible to guarantee that Leaders between D and F have different subset of addresses. This approach allows to deal with the problem of a network with infinite width and assures the assignment of unique identifiers in a wide area of the network.

2) Modified DHCP protocol: the configuration of nodes entering the VANET is performed through a modified DHCP protocol. The node X that is not configured yet checks if there are any Leaders. It

means that X gathers the Hello packets sent from the close Leaders for a time period *T_start*. Then, it estimates who is the nearest Leader and sends it a request for the address.

3) DAD procedure: the IP address received from a Leader is unique only within the *SCOPE* of the Leader. If a node X is configured with the IP_X address and it moves away from its Leader, it could get close to a different node that is using the same IP_X address in another location in the network. For this reason, nodes have to perform a DAD procedure.

Listening to *Hello* packets sent by Leaders, each Normal node is able to know when its Leader is too far and a new address is needed. In fact, *Hello* packets sent by a Leader contain the list of other Leaders in its *SCOPE*. Let be X a Normal node configured by the Leader A and suppose that, at a given time, X does not hear anymore packets from A, but it receives packets from the Leader B that is n hop far from A. If $n > scope$, B is not in the *SCOPE* of A and vice versa. In the *Hello* packets of the Leader B there is no information on A. Consequently, X assumes that it needs a new address. Such DAD procedure does not introduce additional signaling traffic, but it is effective to determine when the node has to configure a new address.

```
1.  if (status==NORMAL)
2.  {
3.      switch (event)
4.      {
5.          case TurnOn:  // the node is turned on
6.              sendRequest();
7.          case recvAddress: // the node receives the configuration parameters
8.          {
9.              my_ip=GetAddress(AddressPkt->ip);
10.             my_leader=GetLeader(AddressPkt->src);
11.         }
12.         case recvHello:
13.         {
14.             outside=CheckLeaderScope(my_leader, HelloPkt);
15.             if (outside)
16.             {
17.                 sendRequest();
18.             }
19.         }
20.     }
21. else
22. {
23.     switch (event)
24.     {
25.         case recvHello:
26.             UpdateScope(ScopeTable);
27.         case recvRequest:
28.             ProvideAddress(addrSet);
29.     }
30. }
```

Fig. 7. Pseudocode for the address configuration and maintenance task

In Figure 7 we provide the pseudocode for the address configuration of nodes. We have distinguished the behavior of normal nodes (lines 1- 14) from the behavior of Leaders (lines 15- 24). At the beginning the node is switched on (line 5) and it needs a valid IP address. For this purpose it sends a request to the closest Leader (line 6). As soon as the Leader processes the request, the node receives the configuration parameters (line 7). At this point, the node can configure its address (line 9) and store the address of its engaged Leader (line 10). From now on, each time a normal node receives a *Hello* packet (line 12), it verifies the presence of this Leader in such packet (line 14). If this condition is not verified (line 15) then the normal node went out from the *SCOPE* of its Leader and needs a new address configuration to avoid duplications. A new request for an IP address is hence sent to a closest leader (line 17).

Focusing on the Leader part, whenever a Leader receives a *Hello* packet (line 25), it updates information on its *SCOPE* (line 26). Instead, if the Leader receives a request for an address from a normal node (line 27), it picks up valid address from its $addrSet$ and assigns it to the requesting node (line 28).

VI. Simulation Results

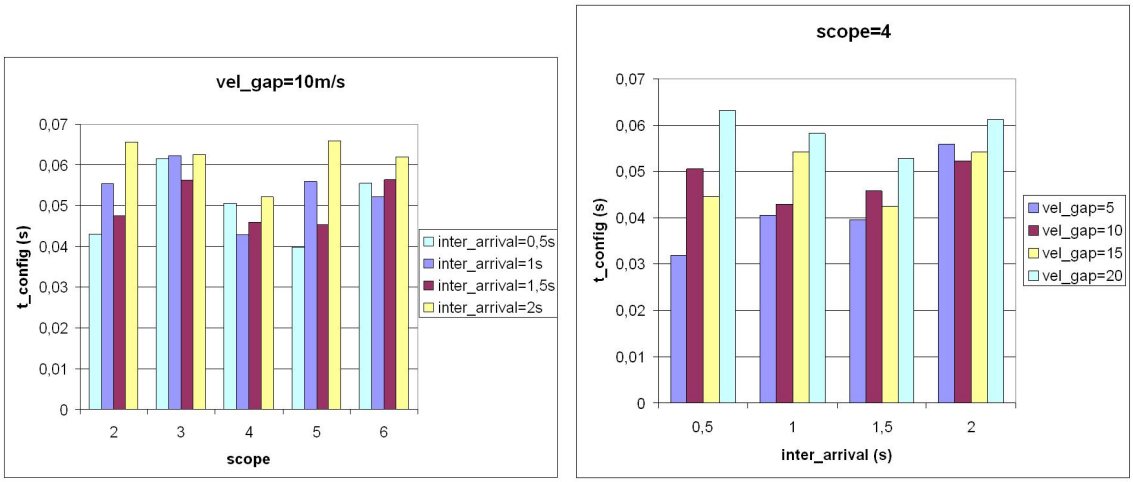
In order to test the protocol performance, VAC has been implemented as a module for version 3.7 of the Qualnet simulator [13]. Scenarios for simulations were created as follows: each scenario was configured with 50 mobile nodes that moved over a $15000m$x$20m$ terrain. Nodes joined the network at the point $(0.0; 10.0)$ with a certain $inter\_arrival$ time in the range $0.5 - 4s$, and moved along the Cartesian x terrain dimension with a random speed in the range $26 \pm (vel\_gap/2)\, m/s$, where vel_gap is the gap between the minimum and the maximum speed of nodes in the scenario. In our simulations we used several values for vel_gap in order to test the behavior of the protocol with reference to different degrees of node mobility. The simulation time was $250s$ in order to leave enough time to all the nodes for joining the network and to have the whole group shifting along the terrain of reference. The node radio range was set to $400m$.

We have evaluated the VAC performance with reference to three parameters:

- *SCOPE* size: it settles the width of the area in which unique addresses are guaranteed. We have used the following set of values for the variable $scope$: $2, 3, 4, 5, 6$.
- vel_gap: this parameter controls the relative mobility among nodes. High vel_gap values imply higher probability that normal nodes go out of the Leader's *SCOPE* and higher instability in the Leader chain configuration. For the vel_gap parameter we have used values $5, 10, 15, 20m/s$.
- $inter\_arrival$ time: this parameter allows changing the node density in the network. We have set the $inter\_arrival$ time to $0.5, 1, 1.5, 2s$.

A. Configuration Time

The main goal of VAC is to perform a reliable address configuration service with low configuration time in order to support also real-time application. ITU-T recommends a maximum delay of 250ms for a quality audio session [1]: higher delays are translated into a bad experience for users. Since during the address configuration procedure data cannot be delivered, it is important that the configuration time remains low. To this aim, Figure 8 shows that nodes are able to configure valid addresses in less than 70ms. This represents a very good result for it proves that VAC is suitable even for real-time applications. Moreover it is the lowest configuration time in comparison with others configuration protocols for ad-hoc networks (about 10 times less) [8][15][14][12][16][3]. This result is achieved through a direct interaction



(a) Configuration time per $scope$ for several values of $inter\_arrival$ time and with constant vel_gap.

(b) Configuration time per $inter\_arrival$ time for several values of vel_gap and with constant $scope$.

Fig. 8. Configuration time in seconds

Normal node-Leader that eliminates delays for leader discovery, flooding requests over the network, and multi-hop interactions among nodes. Moreover, the configuration time results independent from node mobility, SCOPE size and node density, as emerges from Figure 8.

B. Address Configuration Stability

Another important property of VAC is the stability in the node configuration obtained by limiting the number of changes in the network setting. In particular, the variable num_config represents the average

number of address assignments per node (including also the very first one). When a node becomes Leader, it changes its previously assigned address into a new one in order to be consistent with its $addrSet$. Therefore, each Leader obtains an address at least twice (i.e., num_config is at least 2 for Leaders), whereas the minimum number of configurations for a normal node is just 1. We show in Figure 9-11 that the num_config value is low; in fact, each node changes address two-three times on average. In essence, VAC protocol does not cause flickering in the address configuration. Now we discuss in more detail how the num_config value changes with the protocol parameters: *SCOPE* size, vel_gap and $inter\_arrival$.

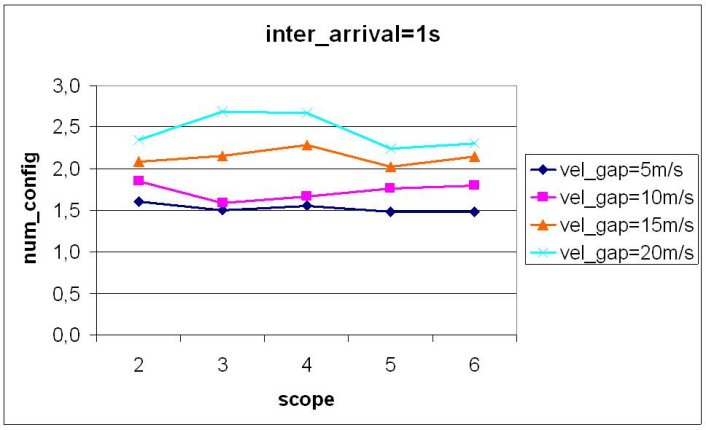


Fig. 9. Number of address configurations of a node per $scope$ for several values of vel_gap and with constant $inter\_arrival$ time.

As shown in Figure 9, the number of configurations remains constant even if the size of the *SCOPE* changes. This happens because such size does not modify the Leader chain organization or the interaction between Normal nodes and Leader ones. On the contrary, the $inter\_arrival$ variable is correlated with the node density in the network.

In the simulations, the network topology is linear and we considered 50 nodes. Increasing the $inter\_arrival$ period corresponds to spreading the nodes on a longer segment of terrain and, consequently, making a longer Leader chain as the chain building metric is the distance among Leaders. This is confirmed by Figure 10, which draws the total number of *Hello* packets sent with different simulative configurations. Since each Leader sends a *Hello* packet every $800ms$ and the thresholds for the Leader chain building were unchanged during the simulations, the only reason that causes an increase in the *Hello* traffic is a higher number of Leaders in the chain and hence a longer chain.

The num_config increases with the $inter\_arrival$ time (Figure 11). This is an obvious consequence of

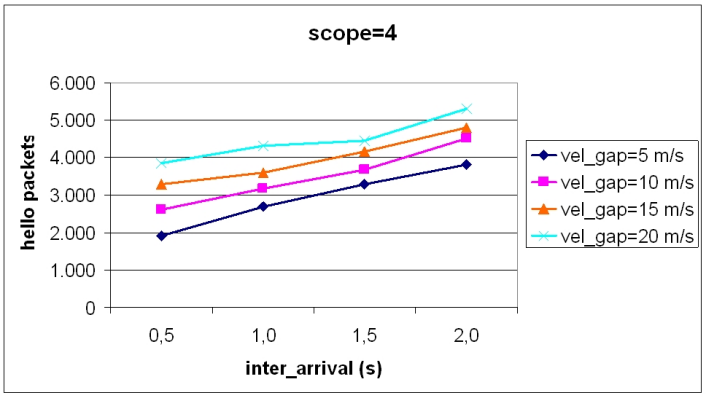


Fig. 10. Number of *Hello* packets per $inter\_arrival$ time for several values of vel_gap and constant $scope$.

the fact that higher $inter\_arrival$ times generate less dense networks with a higher ratio between Leaders and Normal nodes. Since Leaders reconfigure their addresses at least twice, the higher the percentage of Leaders, the higher the num_config value.

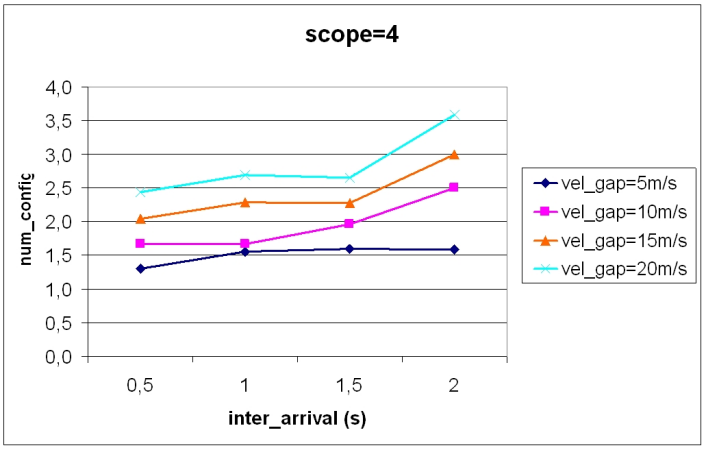


Fig. 11. Number of address configurations of a node per $inter\_arrival$ time for several values of vel_gap and with constant $scope$.

Similarly, the num_config value also increases with the vel_gap (Figure 9-11). This is due to a higher instability of the Leader chain configuration generated by higher node mobility. Indeed, with higher relative speeds among nodes, there is a higher probability that new Leaders enter the chain or current Leaders go away. Furthermore Leaders come close to each other and some of them may be forced

to switch into Normal status.

C. Protocol Overhead

In this Section we evaluate the total overhead introduced by our protocol with reference to the $inter\_arrival$ and the vel_gap parameters.

In order to asses the weight of the hierarchical organization on the whole system, we have separated the signalling traffic due to Leader chain management (i.e., insertion/elimination of Leaders in the chain and information synchronization among Leaders) from the address configuration overhead. The comparison among the two kinds of signalling traffic with respect to various $inter\_arrival$ values is presented in Figure 12.

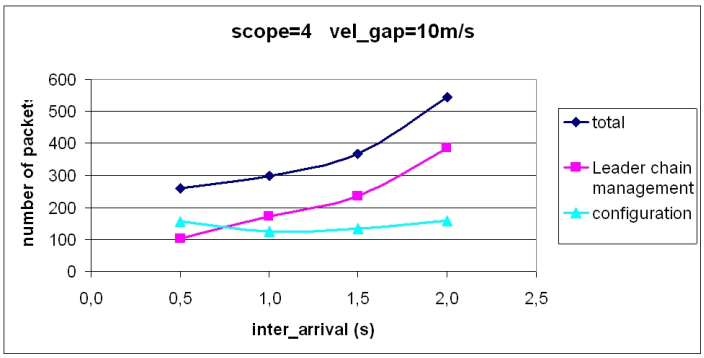


Fig. 12. Number of signalling packets per $inter\_arrival$ time for several values of vel_gap and with constant $scope$.

The traffic due to the chain management increases more rapidly than the node configuration traffic, with an exponential trend that heavily affects the total traffic, especially for high values of $inter\_arrival$ time. However, the total overhead introduced by VAC is very low if compared to the traffic of *Hello* packets (see Figure 10). To limit the *Hello* traffic, an interaction between VAC and the ad-hoc routing protocol through a cross-layer architecture could be exploited. In fact proactive routing protocols make use of periodic update messages and reactive routing protocols generally make use of beacons to be aware of their neighborhood. We can improve the VAC performance by piggybacking information for address management on the periodic update packets or beacons. The current implementation of our protocol is independent from the ad-hoc routing protocol; in our experiments we hence show performance of VAC in the worst case overhead (i.e., without piggybacking).

Similar considerations can be expressed for the total overhead per vel_gap whose outcomes are reported in Figure 13. The node mobility affects the chain management traffic more than the configuration traffic,

and the trend of the total traffic follows the Leader chain behavior. However such trend is quite linear and it can be approximated by a line with a correlation coefficient of only 14.75 thus demonstrating VAC's resilience to high node mobility.

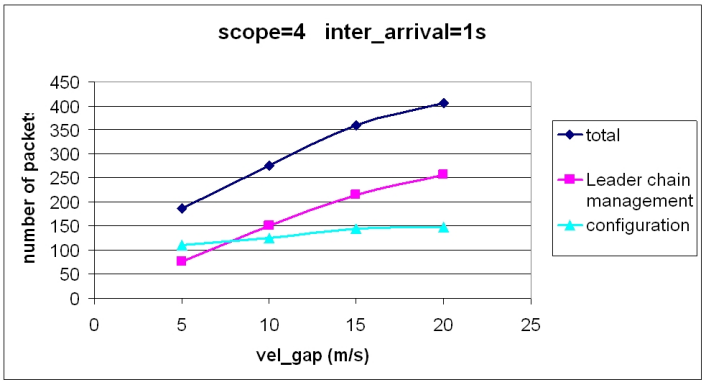


Fig. 13. Number of signalling packets per vel_gap for several values of $inter\_arrival$ time and with constant $scope$.

VII. Conclusion

In this paper we have discussed a fundamental issue in VANETs: the IP address configuration of vehicles. Common systems for dynamic assignment of addresses in wired networks are obviously not suitable in VANETs. Furthermore, the unique characteristics of vehicles preclude us from directly applying techniques developed for traditional ad-hoc networks. Consequently, we have analyzed features of VANETs in order to draw directions for designing address configuration protocols. With these considerations in mind, we have developed VAC, an efficient protocol for the IP address configuration in VANETs. VAC is based on a network backbone in which leaders offer an enhanced DHCP service to all the other nodes in the network. Our approach guarantees a reliable configuration service that is characterized by low signaling overhead and low configuration time. Simulation results convincingly confirmed the efficacy of our scheme.

As future direction, we plan to extend our protocol also for an urban environment and evaluate it in a campus setting with some actual applications running on top of it; for instance, interactive multiplayer online games over a vehicular network.

References

[1] ITU-T Recommendation E.800 (08/94). Terms and definitions related to quality of service and network performance including dependability, August 1994. Revision 1.

[2] EN 301. intelligent wide-band WLAN chipset. http://www.engim.com/.

[3] M Fazio, M Villari, and A. Puliafito. IP Address Autoconfiguration in Ad Hoc Networks: Design, implementation and Measurements. *Computer Networks Journal, Elsevier Science Publisher*, 50:898–920, 2006.

[4] R. Jain, A. Puri, and R. Sengupta. Geographical Routing Using Partial Information for Wireless Ad Hoc Network. *IEEE Personal Communication*, February 2001.

[5] D. B. Johnson, D. A. Maltz, Y-C. Hu, and J. C. Jetcheva. The dynamic source routing protocol for mobile ah hoc networks (dsr), July 2004. IETF Internet draft, MANET working group, draft-ietf-manet-dsr-10.txt.

[6] M. Mohsin and R. Prakash. Ip address assignment in mobile ad hoc networks. In *Proceedings of IEEE MILCOM*, September 2002.

[7] A. Nandan, S. Das, G. Pau, M.Y. Sanadidi, and M. Gerla. Cooperative Downloading in Vehicular Wireless Ad Hoc Networks. In *Wireless On-Demand Networks and Services*, St. Moritz, Switzerland, January 2005.

[8] S. Nesargi and R. Prakash. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In *INFOCOM 2002*, New York, June 2002.

[9] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (aodv) routing, June 2002. IETF Internet draft, MANET working group, draft-ietf-manet-aodv-11.txt.

[10] C. E. Perkins and E. M. Royer. *Ad hoc Networking*, chapter Ad hoc On-Demand Distance Vector Routing. Addison-Wesley Publishers, 2000.

[11] A.K. Saha and D.B. Johnson. Modeling Mobility for Vehicular Ad Hoc Networks. In *poster at ACM VANET 2004*, October 2004.

[12] Y. Sun and E. M. Belding-Royer. A Study of Dynamic Addressing Techniques in Mobile Ad hoc Networks. In *Wireless Communications and Mobile Computing*, pages 315–329, April 2004.

[13] Scalable Network Technologies. Qualnet family of products. http://www.scalable-networks.com/products/qualnet.php.

[14] S. Toner and D. OMahony. Self-Organising Node Address Management in Ad-hoc Networks. In *Personal Wireless Communications (PWC 2003)*, Venice, Italy, September 23-25 2003.

[15] N. H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Boston- Massachusetts, June 2002.

[16] K. Weniger. PACMAN: Passive Autoconfiguration for Mobile Ad Hoc Networks. *IEEE Journal On Selected Areas In Communications*, 23(3), March 2005.