This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Connectivity-Aware Routing (CAR) in Vehicular Ad Hoc Networks

Valery Naumov
ETH Zurich, Switzerland

Thomas R. Gross
ETH Zurich, Switzerland

***Abstract*— Vehicular ad hoc networks using WLAN technology have recently received considerable attention. We present a position-based routing scheme called Connectivity-Aware Routing (CAR) designed specifically for inter-vehicle communication in a city and/or highway environment. A distinguishing property of CAR is the ability to not only locate positions of destinations but also to find connected paths between source and destination pairs. These paths are auto-adjusted on the fly, without a new discovery process. "Guards" help to track the current position of a destination, even if it traveled a substantial distance from its initially known location. For the evaluation of the CAR protocol we use realistic mobility traces obtained from a microscopic vehicular traffic simulator that is based on a model of driver behavior and the real road maps of Switzerland.**

I. Introduction

Ad hoc (or self-organizing) networks operate without a predefined fixed (managed) infrastructure. Vehicular ad hoc networks (VANETs) using 802.11-based WLAN technology have recently received considerable attention in many projects (e.g., VIC'S [1], CarTALK 2000 [2], NOW (Network-on-Wheels)) and industry groups (e.g., the Car2Car Communication Consortium [4]).

Among the ad hoc routing protocols, position-based routing is known to be scalable with respect to the size of the network and is therefore a good candidate for inter-vehicle communication. However, many geographic routing (GR) protocols are designed assuming a random and uniform distribution of nodes, which move freely in an area that is larger (or much larger) than the nodes' average coverage range. Previous studies of ad hoc network protocols show that performance depends heavily on the chosen mobility model [5]–[8] and that a uniform distribution of nodes in a rectangular area is clearly favorable for graph traversal and shortest path algorithms [5], [9], [10]. When the distribution of nodes is more complex and less random (e.g., cars on city roads), then many of the suggested GR protocols experience performance problems.

For our study of VANETs we use realistic mobility traces [5]. These traces are obtained from a multi-agent microscopic traffic simulator that is capable to simulate public and private traffic over real regional road maps of Switzerland [11], [12]. The simulator models the behavior of people living in the area, reproducing their movement (using vehicles) within a workday. For our mobility model we use a 24 hour detailed car traffic trace file in ns-2 movement format with a detailed movement description of an area that includes the main country highways joining the city of Zurich (Switzerland). The advantages of using realistic vehicular traces are elaborated in [5], [13], [14] and are not a topic of this paper.

Several GR protocols are proposed or adopted specifically for VANETs, e.g., [10], [15], [16]. However, the vast majority of these protocols have never been evaluated with a realistic location service. Commonly an idealized mechanism is used such that for every originated data packet the true position of the destination is known, e.g. based on the simulator's global view. This kind of protocol evaluation may easily hide the effect of inconsistent destination positions on protocol performance. The overhead of a realistic location service also remains unknown.

Another problem is that, to the best of our knowledge, all of the GR protocols focus on geographically existing paths but do not take into account if a path between source and destination is populated.

We present a novel position-based routing scheme called Connectivity-Aware Routing (CAR) designed specifically for inter-vehicle communication in a city and/or highway environment. CAR integrates locating destinations with finding connected paths between source and destination. Once a path is found, it is auto-adjusted on the fly to account for changes, without another discovery process. "Guards" help to track the current position of a destination, even if it travels a substantial distance from its initially known location.

This paper is organized now as follows. We talk about the motivation for this work in Section II. In Section III we introduce the Connectivity-Aware Routing (CAR) protocol. The common simulation setup and the evaluation results of CAR are given in Section IV. Related work is considered in Section V. Section VI concludes the paper.

II. Motivation

Many applications for VANETs have been suggested [4] that require multi-hop communication, so a routing protocol should handle fast changes of the ad hoc network topology. A study of VANETs for realistic scenarios shows that AODV [17] (not a GR protocol) combined with Preferred Group Broadcasting (PGB), an optimization of broadcasting, provides better results than GPSR, a GR protocol, GPSR [18] even when GPSR is improved with Advanced Greedy Forwarding (AGV) [5]. At

This research was supported, in part, by the NCCR "Mobile Information and Communication Systems", a research program of the Swiss National Science Foundation.

0743-166X/07/$25.00 ©2007 IEEE

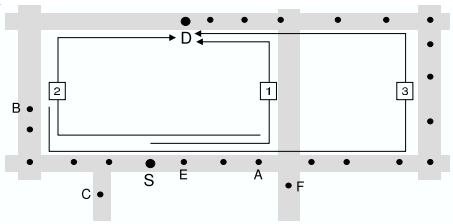


Fig. 1. Find path examples.

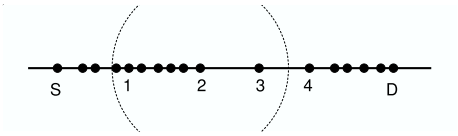


Fig. 2. Influence of the neighbor table accuracy. The accuracy of node 1 neighbor table is far less important for the communication between nodes S and D than those of nodes 2, 3, and 4.

the same time, when non-GR protocols are compared to geographic ones in random rectangular networks with uniformly distributed nodes, then typically GR protocols demonstrate better performance and scalability [5], [9], [18] since GR protocols are not strictly bound to the choice of a node on each hop and make forwarding decisions based on the current neighborhood situation of a forwarding node. However, irregular patterns of road networks often prevent GR protocols from finding existing and populated paths between source and destination pairs, whereas the broadcast-based route discovery of AODV normally reaches the destination, if any connected path exists.

Consider the road situation on Figure 1. S wants to send a packet to D (assume S already obtained D's coordinates from a location service). A routing protocol based on the graph faces traversal (e.g., GPSR) sends the packet from S to E, where a local maximum is detected (no neighbor of E is closer to D than E itself). Thus perimeter mode is activated, and the packet travels to B, where another local maximum is detected. The same way the road paths starting with C and F are tried, before the packet is eventually sent over a populated path to D. Obviously, this routing strategy is unattractive for a vehicular scenario.

Some GR protocols take into account the existing road network (feedback from a navigation system is assumed) and compute paths by using Dijkstra's shortest path algorithm [9], [15]. However, these GR protocols are not always able to find populated paths between S and D. Such a protocol may tell S that there is the shortest path 1 to D (Figure 1). However, after traveling to A, the protocol learns that path number 1 is currently disconnected. Another shortest path found from A is 2, but after traveling to B, path number 2 also appears to be unpopulated. Finally, path number 3 leads the packet from B to D. Since no notifications of the disconnected paths are generated by the nodes, S continues to send packets over path number 1, and the whole routing loop is repeated.

These observations motivated a connectivity-aware GR protocol that uses the advantages yet does not share the disadvantages described above of these protocols.

III. Connectivity-Aware Routing (CAR)

The CAR protocol consists of four main parts: (1) destination location and path discovery; (2) data packet forwarding along the found path; (3) path maintenance with the help of *guards*; and (4) error recovery.

A. Neighbor tables and adaptive beaconing

In CAR all nodes include in the periodic HELLO beacons information about their moving directions and speeds – *velocity vectors* (introduced in AGF [5]) .

When a node receives a HELLO beacon, it adds the sender of that beacon in the node's neighbor table, estimates its own and the neighbor's velocity vectors, and sets the expiration time for the entry in the neighbor table. The entry expires after a time when estimated positions of the current node and the neighbor become separated by more than 80% (configurable) of the average coverage range, or after two HELLO intervals (whatever is smaller). A new HELLO beacon from the neighbor updates the entry.

The HELLO beaconing with a fixed period (with and even without jitter) may have several drawbacks such as: waisted bandwidth, delaying of data packet, increased network congestion. At the same time, if the velocity vector information helps to estimate the availability of a node, the beaconing rate can be made adaptive.

Consider the situation in Figure 2. Node 1 has several neighbors nearby and Node 3 has only a few neighbors (i.e., only 2 and 4). At the same time Nodes 2 and 4 cannot communicate directly. If now S wants to send a message to D, the accuracy of the nodes' neighbor tables around Node 1 is far less important than the accuracy of the tables around Node 3. On the path segment between Nodes 1 and 2, almost any intermediate node can be chosen to relay packets. At the same time, if Node 2 has no (or false) information about Node 3, then communication between S and D will be impossible.

Thus, the CAR protocol uses an adaptive beaconing mechanism where the beaconing interval is changed according to the number of the registered nearby neighbors. The fewer neighbors there are, the more frequent is a node's HELLO beaconing. The basic beaconing interval of 0.5 s is multiplied by a weight proportional to the number of registered neighbors. Therefore Node 3 in Figure 2 beacons more frequently than Nodes 2 and 4 and much more frequently than Node 1. In other words, the beaconing mechanism of the CAR protocol adapts to the changing traffic conditions so that the beaconing rate per km of road stays more or less constant, no matter how many nodes are located on each of the road segments. At the same time, the beaconing intervals of the data traffic destination nodes stay constant to continuously indicate their presence.

We assume that nodes know their locations with the help of GPS or another positioning system.

This technique effectively reduces the beaconing overhead especially in densely populated areas, and at the same time it does not reduce the efficiency of routing.

As with DSR and GPSR, the CAR protocol disables MAC address filtering on the IEEE 802.11 hardware and treats overhead data packets as implicit beacons, to reduce the rate at which HELLO beacons must be sent. Every node that participates in data traffic appends the regular information for HELLO beacons (a node's position and velocity vector) to the data packets. When a node forwards a data packet, it resets the timer for transmitting the next beacon, since data packets carry beacon-equivalent reports.

A node may choose to use this procedure with every forwarded data packet when the node's rate of sending/forwarding of data packets is roughly the same as the node's beaconing rate. Otherwise, if the data packet sending rate is higher, the node adds HELLO beacon information only to some data packets, to keep the implicit beaconing rate close to the 1-2 messages/s, so that the nodes' neighbor tables are maximally current in regions with data traffic load.

B. Guards

To capture key components of a path, we introduce the concept of a *guard*. There are two types of guards: *standing guards* and *traveling guards*. A standing guard (or guard for short) represents temporary state information that is tied to a geographical area, rather than to a specific node. A guard is kept alive by the nodes located in the area. A guard exists as an entry in the periodic HELLO beacon of a node. This entry contains an id, a time-to-live (TTL) counter, a guarded position and radius, and some information that is naturally communicated to the neighbors by the nodes' usual periodic HELLO messaging. A node with a guard can filter or redirect packets or adds information to a packet that will eventually deliver this information to the packet's destination.

The id of the guard is generated by the node that activates the guard and consists of the IP of the node and the node's guard counter (incremented every time a node activates a guard). The age of the guard is a TTL value in ms and is initially set by the activating node. A node that receives a guard adds it to the *Guards-Table* and retransmits it with its own beacon when the time comes to send another beacon signal. The age of the guard is decremented every time a node retransmits it with a HELLO beacon. Once TTL reaches zero, the guard is removed from the node's HELLO beacon. Only nodes within the specified radius around the guarded position are allowed to add a guard into their HELLO beacons; these nodes will be denoted as *guarding nodes*.

Nodes that receive a guard but are not within the guarded radius from the guarded position follow the same procedure as guarding nodes, except that the information from the guard is not added to HELLO beacons.

A traveling guard contains also a velocity vector, in addition to the guarded position and radius. Each node that receives a traveling guard records the time when the guard was received (or last sent). As it is time for the next HELLO beacon, the node computes the new guarded position based on the old guarded position, the velocity vector of the guard, and the time passed since this guard was received. Traveling guards allow the information carried by the guard to travel with a certain speed along the road. The age counter of the traveling guard is decreased with every retransmission.

A node may contain several guards with different tasks activated by different nodes. Once the task is fulfilled, the guard can be removed even before it expires. Then the node that activated the removal procedure adds in its HELLO beacon the id of the guard to be removed. All other guarding nodes retransmit this information once with their HELLO messages (before the guard expires). If the guarding node is actively participating in data packet forwarding, then in addition to the usual HELLO packet, any guards are appended to the data packets.

C. Destination location discovery

To find a destination and a path to it, CAR uses PGB in data dissemination mode. PGB optimizes broadcasts specifically for VANETs, it reduces control messages overhead by eliminating redundant transmissions.

In this mode, a node N (after forwarding the broadcast packet) starts listening if the packet is rebroadcasted further. If no further rebroadcasts happen (no next hop neighbors are currently present), N repeats the broadcast after waiting for a set timeout. This process is repeated until a next-hop neighbor appears and rebroadcasts the packet.

1) Adaptation of PGB to CAR: A source S that requires a path to a destination D initiates a PGB path discovery (PD) and records its own velocity vector into the header of a broadcast packet. To avoid routing loops, a "PD id" is also added. The "PD id" is generated the same way as the id of a guard. Each node that receives a PD broadcast adds its id into the *Received-Path-Discoveries-Table*. The PD packet is not forwarded if the node received it a second time; the entry expires after 60 s.

Each node forwarding a PGB path discovery request rewrites the "Previous forwarder coordinates/velocity vector" fields of the packet by its own data. Also, the information about the packet travel time is added into the field "Travel time" of the packet. Every node forwarding a broadcast packet adds its own processing time into the field.

To estimate the connectivity on the traveled path, each forwarder changes three other packet fields: "Number of hops", "Average number of neighbors", and "Minimum number of neighbors" .

Two velocity vectors are *non-parallel* if the smallest angle between the vectors is less than α (configurable, currently equal to 18°). Nodes that have neighbors with non-parallel velocity vectors identify themselves as being near a crossing or road curve and can serve as relays .

In addition other parameters such as available bandwidth, packet queue status, interference or congestion level, nodes average speeds, neighborhood changes rate, error rate can be recorded

Another way to identify crossings is to compute correlation coefficients based on coordinates of the neighboring nodes [16].

A node adds an *anchor* to a broadcast packet if the direction of the node's velocity vector is different (non-parallel) from the "Previous forwarder velocity vector" field. An anchor contains two *anchor points* – the coordinates of the current node and the coordinates of the previous forwarder as well as their velocity vectors.

As the broadcast passes a new crossing, another anchor is added to the packet (a road curve might be identified as crossing; however such a false-positive identification does not influence the accuracy of the protocol).

In the end, when the broadcast finally reaches its destination, the destination node has the whole path to the source node recorded as a set of intermediate anchor points.

The destination node, after receiving the first PD request, analyzes the available information collected by the request on its path. A decision is taken to issue a route reply immediately or to wait some more time for additional PD requests (that possibly took a different path) to arrive. When several PD requests are received, the destination chooses the path that provides better connectivity and lower delays.

Eventually a route reply is sent from the destination back to the source. A route reply is a unicast packet that contains the destination's coordinates and velocity vector, together with the information collected by the route request on its way to destination. AGF is used to forward the route reply back to the source via the recorded anchor points. When the source receives the route reply, it records the path to the destination and starts transmitting data. Data packets are forwarded in a greedy manner toward the destination through the set of anchor points using the same AGF algorithm. The next subsection describes this forwarding in more detail.

The advantages of this approach to discover a destination's location are (1) it finds the paths that are not only geographically possible but exist in reality; (2) it takes the connectivity into account; (3) there is no need for expensive trial-and-error route tests based on data packet transmissions. (3) only source-destination pairs keep anchored paths to each other. Basically, CAR's discovery algorithm finds the paths in a similar way as Dijkstra's algorithm would do when the connectivity and the packet travel time on each road segment were known in advance to the source and were then used to estimate the link cost.

D. Greedy forwarding over the anchored path

assumes that both the source and the destination inform each other with the help of the location discovery service about their velocity vectors so that

The CAR protocal extents AGF to work with anchor points. Instead of forwarding a data packet to a neighbor that is geographically closer to the destination, a neighbor closest to the next anchor point is chosen. To avoid multiple attempts to gradually get closer to the next anchor point, each forwarding node checks if its position and the position of the next anchor point is separated by less than half the node's coverage range. If so, then this anchor point is marked and the next one is chosen as target. The process continues until the packet reaches its destination.

E. Path maintenance

Any path may become invalid. Let us first assume that path segments between anchor points stay connected. Anchor points are bound to crossings, thus the only possibility for a route to break is a changed positions of the source and destination (the *end point* nodes).

If an end point node moves a substantial distance from its known position (or takes a turn at a crossing), previous protocols fail and must start a new location discovery. Here we present how guards help to adapt to such a situation without loosing data packets and avoid a new location discovery phase.

1) Guards in path maintenance: If an end point node changes its direction, then the node activates a *standing guard*. The guard contains the old and the new velocity vectors of this node. Whenever a guarding node receives a data packet addressed to the node that activated the guard, the guarding nodes adds the guarded location as an anchor point to the header, updates the estimated position of the destination, and retransmits the packet.

However, if – before changing direction – an end point node A was moving against the direction of communication, and the packet travel time (contained in the data packet header) between the end point nodes is larger than the time needed by A to cover a distance equal to the node's average coverage range, then a *traveling guard* is activated. This guard travels in the node's old direction with the node's old speed, rerouting data packets that arrive at the old estimated position of the destination.

Right after activating a guard the node sends a notification packet: the source adds notifications to data packets, the destination sends a special packet . While a notification is on the way, all data packets are rerouted with the help of the guard. Upon receiving the notification, the counterpart recomputes the set of anchor points for future communication. A similar notification is generated if an end point node passes an anchor point or crosses a straight line between two subsequent anchor points.

If an end point node notices that due to speed changes its estimated position known to the communication counterpart and its true position become separated by more than 60% (configurable) of the average coverage range, the node broadcasts a traveling guard, letting the guard travel with the old speed of the node. The guard contains the information about the new node's position and velocity vector.

By default the lifetime of a guard for path maintenance is equal to three times the packet travel time between the end point nodes. The short life time (few seconds) removes the possible scalability problem, as the maximum number of guards coexisting in the same area is limited by the number of vehicles that take a turn on the same crossing within the life time of a guard (considering the worst case scenario all turning

If the protocol includes ACK packets, these may be used to carry such notifications.

vehicles are end point nodes). In case of guards activated due to speed changes, the maximum number of guards is limited by the number of neighbors (moving in the same direction) that changed their speed almost simultaneously.

If the line of sight distance is decreasing and the path length is increasing, then the source node may trigger a path optimization request if the line of sight does not cross the path and the path length is at least two times larger than the line of sight distance. The scope of this request is limited to the current path length.

Guards help adjusting the connected path without employing new path discoveries even if the end point nodes change their moving speeds and/or directions. However, in case any routing errors occur (e.g., path disconnections) the CAR protocol possesses a recovery mechanism described in the next subsection.

F. Routing error recovery

CAR's destination location and path selection mechanism chooses connected paths and avoids routing to a dead-end. However, routing errors may still occur, and CAR should be able to recover from such a situation.

There are two possibilities for routing error to occur. First, the AGF algorithm may fail to forward a packet between two anchor points due to: (a) a temporary gap between vehicles (or raised interference level), such gaps may appear and disappear with time at any place on a road; or (b) long-term disconnections due to a suddenly closed road or an unusually big gap in the vehicular traffic. When we use the term *gap*, we refer always to a gap between the cars on a road that makes communication impossible – the gap is larger than the coverage range.

Second, a packet may reach the estimated destination position after passing the last anchor point but fails to find the destination there. The reason for that event could be that the destination changed direction but could not activate a guard due to a lack of neighbors within communication range; or the guard was activated but later could not be retransmitted due to the same problem.

The next subsections discuss how the CAR protocol handles the these routing errors.

1) Timeout algorithm with active waiting cycle: One approach to tackle temporary gaps (or a raised interference level) is the use of timeout with packet buffering and an active waiting cycle. The forwarding node suspends the packet and periodically checks if the next hop neighbor has appeared. A long-term disconnection recovery algorithm should be invoked when a simple timeout approach failed.

Whenever a node detects a gap, the node first broadcasts a non-propagating next-hop request. This request serves two goals: (1) it tells other nodes that a disconnection is detected and that the current node started buffering packets (became a temporary destination); (2) it tries to detect a next hop node. The request contains the coordinates of the sending node and of the next anchor point (or of the destination). A node that receives this request and identifies itself to be in between the two coordinates replies with a HELLO beacon. If no answer is received, we assume that a node encountered a temporary disconnection due to the minimum possible gap in the vehicular traffic. Since small gaps occur more often than large ones, this assumption is reasonable. Let R be the node's average coverage range and $v0$ its current speed. If feedback from the navigation system is available, the speed limit V on the current road segment can be obtained. If there is no navigation system feedback, we still can estimate the possible speed limit for the current segment, knowing speed limits for city, national, and highway roads and taking a speed limit that is higher than the current node speed.

Figure 3 shows the pseudo-code for the timeout algorithm. To prevent several simultaneous answers from more than one new neighbor, jitter is introduced before replying back.

The idea behind the choice of the wait times is the following: first, we do not want too frequent probing so that a node does not jam the network neighborhood; second, we do not want to miss the opportunity to send the buffered packets further. Although there may be more sophisticated calculations of the wait times suggested (e.g., probabilistic based on collected statistical data), the presented timeout algorithm is simple to implement, does not involve expensive computations, and successfully improves protocol performance.

```
p - incoming packet
/* local maximum detected (e.g., no neighbors
 *  closer to the next anchor point) */
buffer p
probe for undetected next hop neighbor
if found new next hop neighbor X
  de-buffer p; send p to X; return
endif
/* get speed limit V for the current road segment */
v = min (v0, V);
/* assume gap is minimal ~ R */
/* time needed to halve the minimal gap */
wait( R/2 / (2*v) )
/* active waiting cycle */
for i=0; i < MAX_RETRIES; i++
  probe for new next hop neighbors (RREQ)
  if answer received from X
    de-buffer p; send p to X; return
  endif
  wait R/4 / (2*v)
endfor
walk-around();
```

Fig. 3. Timeout algorithm

2) Walk-around error recovery: If the AGF algorithm fails to find the destination at its estimated position (case 1), or the timeout algorithm could not find the next hop host (case 2), the node that detected the problem informs the source about the error and starts a local destination location discovery process. In case 1 the scope of this discovery is limited to half the number of anchor points in the old source-destination path. The broadcast is allowed to travel no more than one half of the old path length. In case 2 the scope is limited to the number of anchor points in the old path to the destination (from the current node) plus 50%. The same applies to the path length.

At the same time, the node with the help of a guard announces itself as a buffering node (or temporary destination).

It receives and stores all data packets that were on the way, before the error notification arrived to the source node and it suspended sending new data packets on the route.

If the location discovery is unsuccessful, the negative result is sent to the source node and it starts a new path discovery from its current position. The packets temporarily buffered by the node can be either dropped or sent back to the source.

On the other hand, if the location discovery is successful, the newly discovered path is communicated to the source node. The path contains the concatenation of the anchor points on the way to the node that detected the problem and the anchor points in the new path from that node. When the source receives this information, it analyzes the new path and if it happens to be that the returned new position of the destination is now closer to the source (closer than to the node that detected the problem), it may decide to launch a new path discovery, especially if the previous path was crossing the line of sight between the source and the destination node and the new one does not.

If navigation system feedback is available, it can be used to optimize the process. While waiting for the new path discovery to complete, the last found route is used to forward data packets.

IV. Evaluation of the CAR protocol

A. Simulation setup

In our experiments we use version 2.28 of the ns-2 simulator with the probabilistic Shadowing model. Unlike the Free Space and the Two-ray Ground Reflection models, the Shadowing model does not predict the receiving power as a deterministic function of distance. Instead, the Shadowing model uses a statistical approach to calculate the receiving power and takes into account multi-path propagation effects.

The maximum possible communication range in Shadowing Model is 400 meters for the city scenarios and 500 meters for the highway; $\beta = 2.5$ (path loss exponent), $\sigma_{dB} = 4$ (standard deviation of the Gaussian distributed random variable with zero mean) for the highway scenarios, and $\beta = 3.5$, $\sigma_{dB} = 6$ for the city scenarios; we model the 2.4 GHz band with 2 Mbps data rate (see [5] for a discussion).

The evaluated protocols are: GPSR, GPSR+AGF [5], and the CAR protocol without and with enabled walk-around error recovery (CAR+WA).

In this paper we present results for three different densities of nodes (*low* – less than 15 vehicles per km of road, *medium* – 30-40 vehicles/km, *high* – more than 50 vehicles/km) in the following movement scenarios: **highway** (averaged over 3 different highway areas, each 10 sub-scenarios for every density of vehicles) and **city** (averaged over 3 different city areas, each 10 sub-scenarios for every density of vehicles) [5]. These aggregate names (*highway* and *city*) will be used further in the paper to refer to the whole subset of corresponding scenarios. 20 CBR traffic sources with a sending rate of 4 packets/s are considered. Sources stop generating data packets 50 seconds before the simulation end. Source/sink nodes stay inside the simulated area (do not leave the area and do not park) for the duration of the simulations (300 seconds).

B. Metrics

We present the following metrics for comparing the performance of the evaluated protocols.

- *Packet delivery ratio (PDR)* – the fraction of the data packets originated by an application that each routing protocol delivers.
- *Average delay of a data packet* – the average difference between the time a data packet is originated by an application and the time this packet is received at its destination.
- *Routing overhead* – the total absolute number of routing packets transmitted during the simulation. Again each hop in a multi-hop route is counted separately (for both metrics).

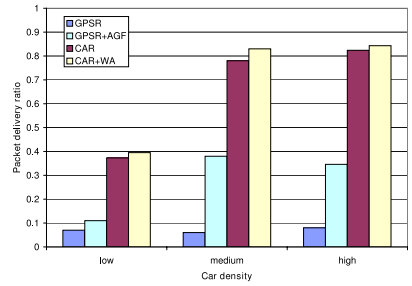
C. Packet Delivery Ratio and Delay

Figure 4 shows packet delivery ratio for city and highway scenarios with different densities of vehicles. For all traffic densities, GPSR performs very poorly in the city scenario, with 5-7% of data packets delivered. Also, the advanced greedy forwarding algorithm (GPSR+AGF) shows moderate performance (up to only 38% of data packets delivered), although performance is noticeably higher than for standard GPSR. Note that GPSR and GPSR+AGF use an idealized location service in the simulation: source nodes obtain the true location of destinations each time a data packet is originated. Despite the additional overhead to discover the real paths and to obtain destination coordinates, CAR and CAR+WA demonstrate much better results than GPSR+AGF . The highway scenarios are geographically less sophisticated than the city scenarios, thus all studied protocols show better PDR in highway areas. Again, CAR and CAR+WA outperform GPSR and GPSR+AGF, despite the need to obtain and maintain paths between source-destination pairs.

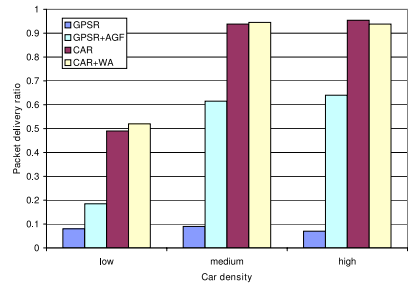
In terms of the average data packet delays (Figure 5), the original GPSR and the GPSR+AGF are always worse than CAR and CAR+WA. For CAR, the route discovery process precedes every first data transmission to an unknown destination, this step adds to the delay of the first data packets. However, the average delay of the data packet for CAR and CAR+WA is much lower than for GPSR and GPSR+AGF. This result is a consequence of CAR's use of real connected paths between source and destination pairs, whereas GPSR and GPSR+AGF often fail due to local maximum resolution encountered by the perimeter mode. CAR easily tolerates short-term disconnections due to gaps or a temporary high interference level (e.g., frequent MAC collisions).

In the scenarios with low density of nodes the network often becomes disconnected, leading to the low PDR and high delays of all tested protocols. However we can see that the CAR

The performance problems of GPSR and GPSR+AGF are discussed in [5], here we just use the performance results of both protocols for assessing the effectiveness of CAR.

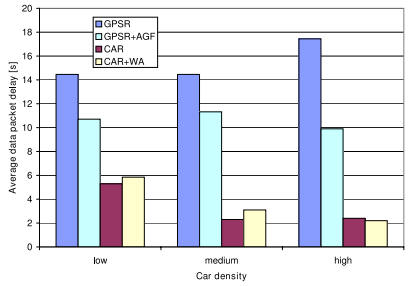


(a) City

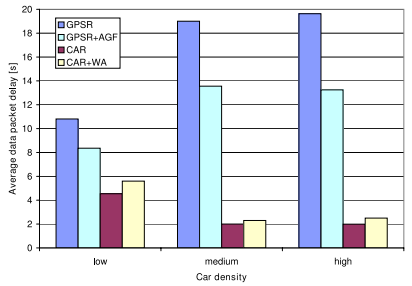


(b) Highway

Fig. 4. Packet delivery ratio.

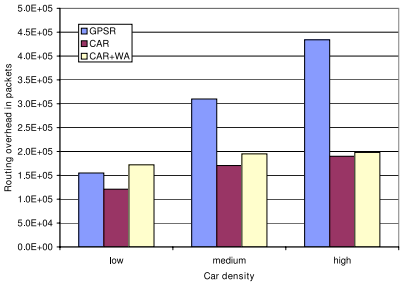


(a) City

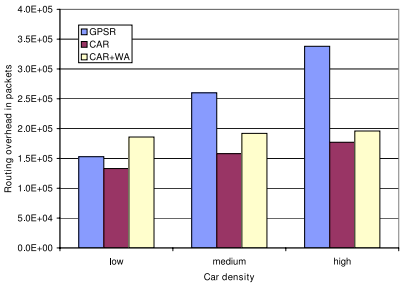


(b) Highway

Fig. 5. Average delay of a data packet.



(a) City



(b) Highway

Fig. 6. Routing overhead in packets.

timeout strategy, which forces the nodes that detect a next-hop disconnection to wait for some period before dropping a packet, helps – PDR of CAR is more than two times higher than PDR of GPSR+AGF in low density scenarios.

D. Routing Overhead

The routing overhead of GPSR consists mainly of the periodic beaconing that the nodes must perform. In CAR, in addition to beaconing, nodes may initiate path discoveries and may activate guards. GPSR can be also configured to pro-actively probe the perimeter for the new routes and to use this knowledge of the local topology to attempt to recover from greedy forwarding failure by routing around voids. In our experiments this option is switched off, since it harmed GPSR performance a lot.

GPSR and GPSR+AGF do not differ noticeably in the overhead they create. With AGF the nodes at the last hop towards the destination sometimes send a non-propagating route request in search for the destination, causing the same overhead as a single HELLO beacon. However, in all the simulated scenarios the number of these requests was less than 0.1% of the total number of beacons. Thus, here we present the data for GPSR only as representative routing overhead for both GPSR and GPSR+AGF.

Figure 6 shows the total routing protocol overhead, measured in total number of routing packets sent network-wide during the entire simulation. For the CAR protocol, the overhead is presented as a cumulative contribution of (1) beaconing, (2) path discoveries, and (3) path maintenance with the help of guards. Below we also discuss the contribution of all kinds of routing overheads to the total protocol routing overhead.

At all simulated traffic densities, CAR generates less routing overhead traffic than GPSR, although no idealized lookup is used. Even when walk around route error recovery is activated (CAR+WA), the CAR protocol overhead stays lower than for GPSR. The reasons for that are explained in the following subsections.

1) Beaconing overhead: Because GPSR's (as well as GPSR+AGF) beacons are sent pro-actively, the level of routing protocol traffic depends on the number of nodes in the area (density of nodes). Thus, the beaconing overhead of GPSR grows directly proportional to the vehicular traffic density. The CAR protocol uses an adaptive beaconing mechanism, and the beaconing interval depends on the node's neighborhood. Thus, in low traffic density scenarios, nodes beacon more frequently than in high traffic density ones. On the other hand, fewer nodes are involved in the low traffic density scenarios, as a result the beaconing overhead of CAR shows smaller dependence on the node density than the one of GPSR.

The use of adaptive beaconing allows CAR to keep the average beaconing overhead from 1.5 to 3 times lower than the beaconing overhead of GPSR, without harming the performance. Moreover, when the adaptive beaconing mechanism is applied to GPSR, this protocol drops up to 30% fewer data packets.

2) Path discoveries overhead: Source nodes launch PGB-based destination location discovery whenever they have data packets to transmit. Ideally only one discovery is needed per source/destination pair; afterwards the discovered paths can be maintained with the help of guards and notifications about direction changes. In fact, route optimizations and route repairs may occur, and in our experiments on average 1.2 path discoveries are initiated in addition to the initial one. However, one path discovery broadcast even in the worst case causes much less routing overhead than the overhead created by sending one HELLO beacon by each node.

In each scenario we have 20 data sources; an average CAR node sends around 150 beacons during the simulation time. Thus, the upper bound estimation for the overhead created by path discoveries and optimizations is (150 + 2.2*20)/150 =+30% compared to the beaconing overhead. In reality it is noticeably lower, since PGB eliminates redundant rebroadcasts. Path discoveries are followed by replies, but replies are unicast packets and do not load the whole networks as broadcast packets may do. Only destination nodes may send relies, thus the overhead created by replies stays in the noise compared to the beaconing overhead.

3) Guards overhead: Guards do not add any overhead in packets, since they are appended to the HELLO beacons of nodes. Although this addition increases the HELLO beacon packet by around 20 bytes, guards have a very short life time (1-6 s) and on average are transmitted only 1-2 times by each node. An average CAR node sends around 150 beacons and needs to transmit only 2-3 guards during the simulation time.

V. Related work

Many GR protocols are described in the literature. Planar graph face traversal algorithms, like perimeter routing [18] or FACE-2 routing [19] (walking around a void) assume a nearly perfect circular radio range of nodes, and all nodes have identical coverage ranges. Although these algorithms are able to deliver data to destinations in an all-time connected network, for every data packet sent they may require the traversal of almost the whole network graph to find a path. These protocols suffer a lot from destination location inconsistency. Also, an irregular radio may lead to routing loops during the face traversing phase [5], [21].

Another class of GR protocols – shortest path algorithms – takes advantage of road map knowledge. These protocols have similar behavior with a navigation system, returning the shortest path between source and destination [9], [15], [16]. However, there is no guarantee that the computed path is currently populated. The protocols need to send bulky data packets along the computed route until the local maximum is reached. Then a new shortest path is calculated, and so on. This process may take quite some time and may waste network capacity. A more advanced algorithm adds weights to roads (e.g., bigger roads with bus traffic have higher weights than secondary roads) that may improve connectivity on the computed path [10]. Although the probability of such a route to be connected increases, there still can be periods of disconnection. Also all data flows will be attracted to these main roads, increasing there the congestion and contention level, decreasing the throughput. At the same time side-roads, even with currently good connectivity, may be left without attention. Some shortest path protocols call a computationally expensive Dijkstra algorithm for each registered neighbor before a forwarding decision is taken [15]. These may noticeably increase the data packet delay in a real life situation, where computation overhead is not zero.

Typically, before the data forwarding process starts, GR protocols require the coordinates of the destination nodes to be known. A vast majority of the suggested GR protocols assumes destination coordinates are known any time a node wants to send a packet [10], [15], [16], [18], [22], [23]. The performance of these protocols is then evaluated using an idealized zero-overhead location service. However, it remains unclear how the real destination discovery service may influence the performance of these protocols. If an end point node (typically a destination node for UDP traffic) moves a substantial distance from its known position, all suggested protocols fail and need to start a new location discovery. However in the vast majority of publications that are based on an idealized location service, this situation almost never occurs, since every originated data packet is stamped with the true position of the destination. With the send rates of 2-10 packets/s typically used in these studies, an idealized location service must be called 2-10 times/s. It is hard to assume that the overhead created by these queries does not influence the protocol performance at all.

The use of anchors is not completely new. DSR [24] records at the route discovery phase the set of intermediate nodes that are later used in data forwarding. A-STAR [10] and GSR [9] protocols require road map knowledge to pre-compute the path to the destination. Then the source node includes into each packet a route vector composed of a list of anchors or fixed geographic points through which packets must pass. Terminode Remote Routing is another example of the anchor-based routing [23].

The use of timeouts is also mentioned in Spatially Aware Routing (SAR) [15]. However the authors did not present any details on the algorithm (e.g. how often a node tries to resend the failed packet). In SAR, the routes are computed by Dijkstra's shortest path algorithm based only on road-map graphs; no connectivity information on the computed path is available. Thus many data packets may be sent over a disconnected path and are forced to wait before eventually recomputing a route.

In this paper we introduced guards – the entries in the node's periodic HELLO beacons that are naturally communicated to neighbors without creating an additional overhead. In LANMAR [25] for each group of nodes a landmark is elected. The coordinates of landmarks are proactively known to all nodes and are used to route packets globally, and then each landmark knows how to route locally. A "FORWARDING POINTER" in HELLO beacons is used in ELF (Efficient Location Forwarding) [26] to redirect between home regions,

and in the Grid Location Service [27] to keep track of to which cell the node moved to from the current cell. Since every node needs to inform the home region or cell about its new location, the number of these messages may grow fairly large.

VI. Concluding remarks

In this paper we presented CAR, a new Connectivity-Aware Routing protocol for VANETs. The CAR protocol is based on PGB and AGF to provide a scalable low overhead routing algorithm for inter-vehicle communication both in the city and on the highway.

CAR is able to locate destinations without using an idealized location service. Rather than relying solely on knowledge of the road layout, CAR adapts to current conditions to find a route with sufficient connectivity so as to maximize the chance of successful delivery.

Route maintenance is organized with the help of guards, so that route changes are corrected on the fly.

The evaluation of the CAR protocol is based on realistic traces of vehicular movement in an area of 350 km x 260 km with 260'000 vehicles participating in the traffic. The traces are obtained from a microscopic vehicle traffic simulation on the real road maps of Switzerland, and to increase the credibility of our study, we model irregular radio channel behavior by using the probabilistic Shadowing signal propagation model. The comparative evaluation of CAR with other routing schemes shows that the CAR protocol delivers a clear improvement in the data delivery rate and the average data packet delays, despite the overhead that is created by path discovery phase. Further evaluations are necessary to extend these finding, but the preliminary results obtain are encouraging.

CAR is presented here as a unified protocol but the key concepts can also be incorporated into other protocols. E.g., incorporating CAR's adaptive beaconing mechanism into GPSR improves GPSR's performance by up to 30%.

VANETs must deal with changing topologies, and a number of issues must be addressed before VANETs can be become a practical reality. CAR addresses one of the key issues that must be dealt with in such an environment: by constructing a minimal "infrastructure" in the form of guards along a path from a source to a destination, CAR maintains enough state to allow efficient communication between two moving nodes.

Acknowledgements

We thank Kai Nagel for the microscopic traffic simulator that made possible the evalution.

References

[1] S. Yamada, "The strategy and deployment plan for VICS," *IEEE Communication*, vol. 34, no. 10, pp. 94–97, 1996.

[2] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "Cartalk 2000 – safe and comfortable driving based upon inter-vehicle-communication," in *Proc. IEEE IV'02*. http://www.cartalk2000.net, Jun 2002, pp. 545–550.

[3] N. on Wheels, "www.network-on-wheels.de."

[4] C. C. Consortium, "www.car-2-car.org."

[5] V. Naumov, R. Baumann, and T. Gross, "An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces," in *Proc. ACM MOBIHOC'06*, 2006, pp. 108–119.

[6] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," in *Proc. ACM/IEEE MOBICOM'02*, Aug 2002, pp. 483–502.

[7] C. Bettstetter, "Mobility modeling in wireless networks: Categorization, smooth movement, and boder effects," in *ACM SIGMOBILE'01*, 2001, pp. 55–67.

[8] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *INFOCOM'03*, San Fransisco, California, USA, Apr 2003, pp. 1312–1321.

[9] C. Lochert, H. Hartenstein, J. Tian, H. Füssler, D. Hermann, and M. Mauve, "A routing strategy for vehicular ad hoc networks in city environments," in *Proc. IEEE IV'03*, Jun 2003, pp. 156–161.

[10] B. Seet, G. Liu, B. Lee, C. Foh, K. Wong, and K. Lee, "A-STAR: A mobile ad hoc routing strategy for metropolis vehicular communications," in *Lecture Notes in Computer Science*, vol. 3042, no. 0302-9743. Springer Berlin / Heidelberg, Jan 2004, pp. 989–999.

[11] B. Raney, A. Voellmy, M. Vrtic, and K. Nagel, "Towards a microscopic traffic simulation of all of Switzerland," in *Proc.ICCS'02*, 2002, pp. 371–380.

[12] B. Raney, A. Voellmy, M. Vrtic, K. Axhausen, and K. Nagel, "An agent-based microsimulation model of Swiss travel," *Networks and Spatial Economics*, vol. 3, pp. 23–41, 2003.

[13] J. Jetcheva, Y. Hu, S. PalChaudhuri, A. Saha, and D. Johnson, "Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture," in *Proc. IEEE WMCSA'03*, Monterey, CA, Oct 2003, pp. 32–43.

[14] D. Choffnes and F. Bustamante, "An integrated mobility and traffic model for vehicular wireless networks," in *Proc. ACM VANET'05*, 2005, pp. 69–78.

[15] J. Tian, L. Han, K. Rothermel, and C. Cseh, "Spatially aware packet routing for mobile ad hoc inter-vehicle radio networks," in *Proc.ITS'03*, Shanghai, China, Oct 2003, pp. 1546–1551.

[16] C. Lochert, M. Mauve, H. Füssler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE'05*, vol. 9, no. 1, pp. 69–72, 2005.

[17] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. IEEE WMCSA'99*, Feb 1999, pp. 90–100.

[18] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc. ACM/IEEE MOBICOM'00*, Boston, Massachusetts, USA, 2000, pp. 243–254.

[19] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proc. ACM DIAL M'99*, Aug 1999, pp. 48–55.

[20] V. Naumov and T. Gross, "Scalability of routing methods in ad hoc networks," *Performance Evaluation*, vol. 62, no. 1–4, pp. 193–209, Oct 2005.

[21] Y. Kim, J. Lee, and A. Helmy, "Impact of location inconsistencies on geographic routing in wireless networks," in *Proc. ACM MSWIM'03*, 2003, pp. 124–127.

[22] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: multihop performance," *IEEE Trans. on Mob. Comp.*, vol. 2, no. 4, pp. 337–348, Oct 2003.

[23] L. Blazevic, S. Giordano, and J.-Y. L. Boudec, "Self organized terminode routing," *Cluster Computing*, vol. 5, no. 2, pp. 205–218, 2002.

[24] D. Johnson, D. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," *Ad Hoc Networking*, pp. 139–172, 2001.

[25] G. Pei, M. Gerla, and X. Hong, "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility," in *Proc. ACM MOBIHOC'00*. Piscataway, NJ, USA: IEEE Press, 2000, pp. 11–18.

[26] S. Philip and C. Qiao, "ELF: efficient location forwarding in ad hoc networks," in *Proc. IEEE GLOBECOM'03*, vol. 2, Dec 2003, pp. 913–918.

[27] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proc. ACM/IEEE MOBICOM'00*, 2000, pp. 120–130.