Computer Networks 55 (2011) 2504–2519

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Going realistic and optimal: A distributed multi-hop broadcast algorithm for vehicular safety

Alessandro Amoroso, Gustavo Marfia, Marco Roccetti *

Computer Science Department, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy

ARTICLE INFO

Article history:
Received 12 July 2010
Received in revised form 19 April 2011
Accepted 21 April 2011
Available online 30 April 2011
Responsible Editor: E. Ekici

Keywords:
Vehicular networks
Distributed multi-hop broadcast algorithm
VANET
Vehicular safety

ABSTRACT

A subject of great interest and important investments by governments, navigation system companies and street management authorities is highway safety. In this context, an important role is played by applications designed to warn drivers of upcoming dangers. An example is vehicular accident warning systems, which advertise accident events to approaching vehicles. The effectiveness of currently in use vehicular accident warning systems can be jeopardized by their: (a) inability to provide an accident warning to the closest approaching vehicles; and, (b) high delays in advertising an event. In fact, such systems are unable to reach the vehicles that are closest to an accident site due to the absence of any deployed automatic detection and broadcast mechanisms. The future deployment of Vehicular Ad hoc Networks (VANETs) can fill this gap. By leveraging on the distributed nature of ad hoc networks, accident warning systems can rapidly alert the vehicles which most risk their involvement in a crash. To reach this goal, VANET-based accident warning systems require the design of efficient broadcast algorithms. A number of solutions have been proposed in the past few years. However, no such proposals, to the best of our knowledge, assume realistic wireless propagation scenarios. The scope of this paper is to present an optimal distributed algorithm, working at the application layer, for the broadcast of safety messages in VANETs. Optimality, in terms of delay, is achieved in unidimensional highway scenarios and under realistic wireless propagation assumptions. This is the only algorithm, to this date, capable of reaching all vehicles with the minimum number of transmissions within a realistic setting.

1. Introduction

Currently in use accident warning systems risk being late in alerting the vehicles approaching an accident site. The motivation is twofold: (a) their infrastructure is centralized, thus resulting in an increased delay for distributing alerts; and, (b) they rely on human intervention to detect and communicate accident events to a control center. Typically, an accident occurrence is communicated by humans to a traffic control center. This information is then displayed to drivers in the area surrounding the accident site, mainly leveraging on two technologies: Personal Navigation Devices (PNDs) and Dynamic Message Signs (DMSs).

Essentially, PNDs integrate an FM radio interface through which each vehicle on the road can receive traffic related information from the control center. DMSs are electronic devices, set along the road in specific positions, which display traffic information coming from a centralized entity (with DMSs, the problem is further exacerbated by the fact that only a small subset of potentially interested vehicles can be warned). Either way, all this results in traffic information which may come to potentially interested drivers too late to prevent multiple vehicle collisions from occurring.

Using VANETs and related transmission technologies, such as IEEE 802.11p, may change the show. We anticipate

* Corresponding author. Tel.: +39 0512094503.
E-mail addresses: amoroso@cs.unibo.it (A. Amoroso), marfia@cs.unibo.it (G. Marfia), roccetti@cs.unibo.it (M. Roccetti).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.04.011

here the positive effect that VANET-based technologies may have in guaranteeing vehicular safety with the following example. We observe the number of vehicles that could avoid being involved in an accident when a VANET-based accident warning system is deployed. In particular, we simulated a vehicle crash on a three lane highway and estimated how many vehicles, on average, could be involved (with and without an inter-vehicular accident warning system). Vehicles speeds and related time-headway distributions are realistic as drawn from [1]. We also considered two realistic traffic flow scenarios, two realistic pavement conditions, realistic driver response times, and vehicles lengths as well [2–5].

We ran 100 simulations. The average number of crashed vehicles involved in an accident are shown in Fig. 1 (with their [5%–95%] confidence intervals). The leftmost bar results from a scenario where no accident warning system is deployed. The rightmost one represents the average number of vehicles involved in an accident when a VANET-based accident warning system is exploited. As a representative case, consider the situation when the pavement is dry and braking is, hence, more effective (leftmost experiment in Fig. 1). In such a case, the average number of crashed vehicles is reduced by nearly the 40%. This clearly shows that employing VANET technologies may be effective, as the spread of alert information occurs more rapidly. This is confirmed also in [6], where authors show that the latency of an alert message sent from one vehicle to a subsequent one along the road is on average 2.5 ms, but never exceeds 11 ms.

As a proof that a VANET can be put to good use in this application field, an extensive body of literature exists [7,8,11,12]. The problem, however, is that no existing approach considers realistic wireless propagation effects. The main two problems are that: (a) different vehicles may have different transmission ranges and receiver sensitivities depending on the hardware devices they mount, and; (b) transmission ranges between a pair of vehicles can be asymmetric: that is, it may happen that vehicle *a* can receive packets from vehicle *b*, but not *vice versa*. Furthermore, all this may be exacerbated by the fact that, with the passage of time, transmission ranges can vary due to a number of reasons, such as intervening obstacles and physical propagation effects (including multi-path and fading).

Our contribution is to present a new inter-vehicular distributed broadcast accident warning algorithm (working at the application layer) and prove it is optimal in multi-lane, strip-shaped portions of roads (also termed one-dimensional or 1D). The idea behind our approach is that of exploiting a Relay mechanism to spread alert messages from car to car. Differently from any other existing approach, we have chosen to exploit the *farthest spanning Relay*, rather than the *farthest Relay* to disseminate alerts. This innovative strategy is at the basis of the optimality of our algorithm. Nonetheless, this choice poses the additional problem of identifying such *farthest spanning Relay*. We have originally solved this problem by devising a distributed Oracle service, able to assess a realistic picture of the transmission setting. Along with a formal optimality proof of our algorithm, we also present many simulation results confirming the validity of our idea.

The paper is organized as follows. In Section 2 we provide a summary of the literature available on the subject. Our algorithm, its implementation and a running example are discussed in Sections 3–5, respectively. The Oracle

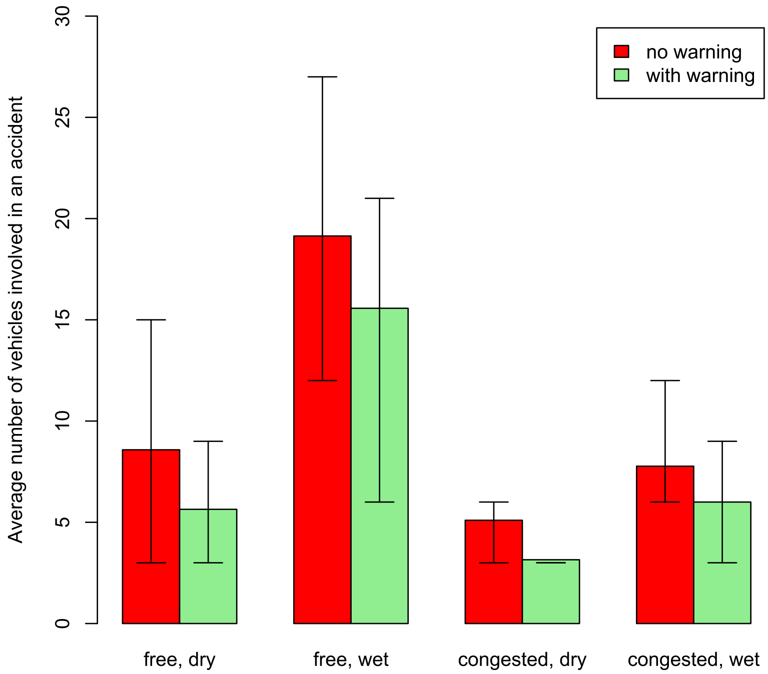

Fig. 1. Average number of vehicles involved in an accident.

service performance, the optimality theorems and a performance comparison between the *farthest spanning* and the *farthest Relay* strategies are respectively presented in Sections 6–8. Simulation results are presented in Section 9. Section 10 concludes this paper.

2. Related work

Broadcasting in VANETs differs from broadcasting in traditional mobile ad hoc networks in a number of aspects [7,13,14]. First, vehicle speeds are usually higher. Second, vehicles are mainly constrained to move over linear, unidimensional (1D) topologies. Therefore, we limit our literature review to schemes specifically designed for vehicular scenarios.

Indeed, a slow broadcast delivery can be due to a non-optimal number of hops experienced by a message to cover the whole vehicular network and, more in general, by an excessive number of simultaneous forwarders [15]. We review prominent approaches proposed in the literature that address this issue.

One of the most recent, as well as promising, ideas to minimize the number of transmissions required to inform a platoon of vehicles, is that of spreading alert information by using a Relay. A common approach, to this date, has been that of choosing such Relay as the farthest among all the potential forwarders. A solution that aims at selecting the farthest vehicle as a Relay is introduced in [11]. In brief, on receiving a message, each vehicle sends a jamming signal of length directly proportional to its distance from the sender. If a receiver finds the communication channel free, after sending its jamming signal, it assumes it is the farthest vehicle from the sender, and, therefore, its Relay.

Another scheme that aims at selecting the farthest Relay may be found in [8–10], where this kind of strategy is exploited over unidimensional topologies. A vehicle, upon receiving a message, computes a contention window which is inversely proportional to its distance from the sender. The farther the receiver, the shorter its contention window. However, this scheme is affected by the assumption that a unique, constant, and well known transmission range is exploited for all vehicles in every moment. To fix this, authors of [12] have devised an algorithm where, differently from [8–10], an automatic transmission range estimator is exploited to assess the actual transmission range for every car in the platoon. This generally results in lower delays in forwarding alert messages.

Unfortunately, all these approaches fall short in recognizing that the highly stochastic nature of a vehicular wireless channel can, not only induce variable transmission ranges in space and time, but also asymmetries (e.g., one vehicle can hear the other, but not *vice versa*). Our approach definitely fixes this problem.

3. Description of the algorithm

This story can be told in two different parts. The former is simple, the latter is trickier. Let us now tackle the simple one.

It is easy to understand that in a realistic setting there is no guarantee that the *best Relay* coincides with the *farthest Relay*. In fact, if vehicle *b* hears vehicle *a* (and *a* hears *b*), and *b* is the farthest vehicle among those that hear *a*, the decision to relay the alert message from *a* to *b* may not be optimal, as *b* could have a very short transmission range. Instead, there could be a given vehicle *c*, between *a* and *b*, whose transmission range could span farther than that of *b*. The cause of all this is simply that transmission ranges can vary depending on the given vehicle.

Let us now focus on the trickier part and begin with an intuitive definition of what a *farthest spanning Relay* is. Indeed, it is that vehicle *b*, that hears *a*, able to span farthest among all other vehicles. In the real world there is a problem with such farthest spanning Relay. The problem is that, due to the asymmetry in transmission ranges, *a* could not hear *b*, thus ignoring its existence. If *a* has to give its permission to *b* before *b* can relay the information coming from *a*, our story ends here unsuccessfully. Hence, to solve this tricky problem, we have devised an original mechanism able to permit *a* to become aware of the existence of *b*. This is done through a distributed Oracle service, which exploits, as possible intermediaries, vehicles set along the portion of road between *a* and *b*, in order to build a communication chain through which *a* eventually hears *b*.

Thus, if a farthest spanning Relay *b* exists for *a*, and if it is somehow possible to make *a* aware of this, our system will always allow *a* to spread farthest its alerts, using the *farthest spanning Relay*. The idea that *a* must give the green light to *b*, before *b* relays messages coming from *a*, is not an extravaganza, but it follows that stream of works which exploit this strategy to minimize the number of transmissions [16,17].

Let us anticipate how the Oracle works. If *a* cannot hear *b* (while *b* hears *a*), the Oracle at *a* is that software component which puts *a* in the condition of knowing that *b* can be reached in a single hop. This information can arrive at *a* through Oracle messages sent by possible intermediaries, placed in between *a* and *b*. It is worth mentioning here that Oracle messages contain the position of their sources, thus providing vehicles with the information necessary to know when they can play an intermediary role between any couple of cars.

To make this clearer, let us extend our example supposing that in between *a* and *b* lie two additional vehicles, say *x* and *y*. Assume the general case where *a* precedes *x* which precedes *y* which precedes *b*. Obviously, both *x* and *y* hear *a*, but *a* does not know this. What happens with our Oracle service is the following (see also step 1 in Fig. 2).

Eventually, *y* receives from *b* an Oracle message which allows *y* to know that it hears both *a* and *b* directly, and that its own position is between these two vehicles (step 2).

Later, eventually, *x* receives an Oracle message from *y*, which permits to *x* to know that it hears *a* and *y* directly. Further, *x* now knows from *y* that *y* itself and *b* have heard directly from *a*. Moreover, *x* now knows also its relative position, that is, the fact that *x* precedes *y* which precedes *b*, while *a* precedes *x* (step 3).

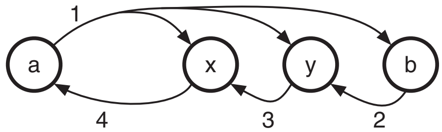

Fig. 2. Sequence of Oracle messages that make *a* aware of the existence of *b* (as well as of *x* and *y*).

Finally, at some point in time, *a* receives an Oracle message from *x*. This message permits to *a* to know that it has directly heard *x*. Further, *a* now knows from *x* that *x* itself, *y* and *b* hear directly from *a*. Furthermore, *a* now knows the relative positions of all vehicles involved, and their transmission ranges, as well. At this very final point, *a* can put to good use all this information to construct an ordered list of possible Relays (*x*,*y* and *b*), to be passed to the Relay software component running on *a*.

Let us anticipate now how the Relays work. An alert message at vehicle *a* may originate from its accident warning layer (in case *a* generated the alert), or from a preceding Relay that has selected *a* as its *farthest spanning Relay*. Either way, the Relay at *a* creates a new message appending to the original alert message an ordered list of next possible Relays, computed based on the information its own Oracle has passed to it. It is worth noticing that this list, at each vehicle *a*, is ordered based on the value given by the sum of the positions and the transmission ranges of all the vehicles that can hear *a*. The first vehicle of the list is the one that has been designated to relay the message farther. If a fault occurs preventing the first in the list from relaying the message, the following entries in the list are exploited to this aim.

Now, before heading on, we should explain why we omitted considering the differences that may be found, in terms of receiver sensitivity, between wireless interfaces when devising our algorithm. In fact, in theory, utilizing our scheme there could exist one or more vehicles between the sender and the *farthest spanning Relay* which are unable to receive an alert message, due to the limited sensitivity of their receivers in comparison to that of the *farthest spanning Relay*. In simple words, there could be the theoretical possibility that *b* hears *a*, while *y*, for example, does not. In the reality, however, receiver sensitivity differences cause transmission range increments that do not exceed a few tens of meters and can hardly be noticed in relevant highway scenarios where headway distances fluctuate around hundreds of meters, given the speeds that are involved [19–21].

Nonetheless, it can occur that cars are packed within a few tens of meters along a certain portion of road, due to traffic congestion or to drivers that do not obey to traffic rules. In this specific situation, even receiver sensitivity differences of a few dBm can cause a circumstance where vehicle *b* would receive the alert message coming from *a*, but *y* does not, even if *y* is only a few tens of meters away from *b*.

However, also this specific case is easily taken care of by our scheme: in fact, vehicle *y* will receive the alert message (it has lost), as soon as vehicle *b* relays that message. At that precise moment, it is no longer possible that *y* remains unaware of the real situation, as it is only a few tens of meters away from *b*.

In the following we discuss in more detail both the Oracle, the Relay and their implementation.

4. Oracle, Relay and their implementation

Our system is composed by two stacked functional blocks. This stack runs on each vehicle. The upper functional block, called *Relay*, interfaces with the atop accident warning application, and sends and receives alert messages. The lower functional block, called *Oracle*, provides necessary data to the Relay.

Fig. 3 represents the stack running on three different vehicles. In particular, the accident warning layer at vehicle 1 generates an alert message to be relayed through the service offered by the *Relay*. As it can be noticed from the figure, the Relay module of a generic vehicle *k* receives this message, delivers it to its accident warning layer, but does not relay it farther, as vehicle *k* has not been identified as the *farthest spanning Relay* of vehicle 1. Instead, vehicle *n* receives the alert message, delivers it to its accident warning layer and finally relays it to all the ahead vehicles, as it has been selected as the *farthest spanning Relay*.

Summarizing, the alert warning produced by vehicle 1 is relayed by that specific vehicle which has been identified as the best Relay. This motivates the fact that in Fig. 3: (a) the Relay at vehicle *k* does not use the information provided by its Oracle (depicted as an absence of communication between the two software components in Fig. 3), while; (b) the information provided by the Oracle is exploited by the Relay of vehicle *n* to determine the next *farthest spanning Relay*.

4.1. The Oracle

The main purpose of the Oracle, at each given vehicle *v*, is to assess the positions and the transmission ranges of all the vehicles that can hear *v*.

Given a vehicle *v* its Oracle maintains three lists:

- *Out*: the list of vehicles that hear *v*. This is the list of potential Relays of an alert message;

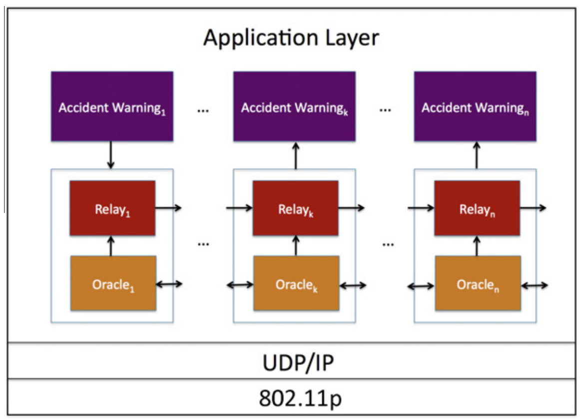

Fig. 3. Functional blocks of the architecture.

- *In*: the list of vehicles that have been heard by v;
- *Aware*: the list of known sender–receiver pairs, characterized by v laying between them. This list is necessary to face asymmetries, as it will become clearer with the example in Section 5.

Each vehicle v is characterized by a tuple $\langle id_v, p_v, f_v, b_v\rangle$. id_v is its unique identifier, p_v the position, f_v the forward transmission range and b_v the backward transmission range. As to the problem of computing the values of p_v, f_v and b_v, the solution is as follows. The value of p_v can be simply obtained by means of any device, installed on the vehicle, which is able to provide the geographical position of that vehicle along the road. The values of f_v and b_v, instead, can be computed based on a simple algorithm, ran by the Oracle, which exploits the relative positions between communicating vehicles.

While the elements in the *Out* and *In* lists represent a vehicle ($\langle id_i, p_i, f_i, b_i\rangle$), the elements of the *Aware* list, instead, have the following structure: ($\langle id_r, p_r, f_r, b_r\rangle, id_s$). This simply represents the information, managed by v, that vehicle r has received an Oracle message from sender s. Moreover, if v stores an *Aware* list with the above information, this entails that v is moving along its lane between r and s. As we are concerned with unidimensional portions of roads, this may be also formally expressed with the following formula: $min\{p_s, p_r\} < p_v < max\{p_s, p_r\}$.

To this point we have provided details about the data managed by an Oracle. From now on, we discuss how an Oracle works. Each Oracle exploits Oracle messages to keep up with surrounding vehicles. Its activity is that of sending and receiving Oracle messages as discussed below.

The messages Oracles exchange between each other have the following format $\langle sender, In, Aware\rangle$ (see Fig. 4): where *sender* amounts to the data representing the sender of the message ($\langle id, p, f, b\rangle$), while *In* and *Aware* are the lists discussed above. Note that the length of an Oracle message may vary depending on the number of vehicles whose information it is managing.

4.1.1. Sending an Oracle message

Each Oracle waits a random time T (whose value will be discussed later) and then sends out an Oracle message as shown in Fig. 5. In light of the considerations that we have shown up to this point, the code in Fig. 5 does not need any further explanation. Rather, it is interesting to notice that the activity of an Oracle has the aim of taking a picture of what really happens along the road. It goes without saying that the higher is the frequency according to which Oracle messages are exchanged, the higher is the accuracy of this picture. Obviously, an augmented accuracy should be traded for a larger bandwidth occupancy (in the following Section 7 we provide details and formulas which assess this trade-off).

```
upon sendOracleMessage()
{ for ever
  { sender = list(id, p, f, b);
    msg = append(sender, In, Aware);
    send(msg);
    sleep(random(0,Tmax));
} }
```

Fig. 5. Oracle sending a message.

4.1.2. Receiving an Oracle message

Upon receiving an Oracle message, an Oracle updates its three lists as shown in Fig. 6.

As a first operation, the Oracle running on v updates the *In* list of vehicles it has heard. In particular, if the sender was never heard before by v a new entry for this sender is created in the *In* list of v. Otherwise, the $\langle id, p, f, b\rangle$ elements, which represent the sender, are updated within the *In* list of v (line 2). Essentially, with this operation v becomes aware of the most recent information concerning the sender.

Then, within a cycle running from lines 3 to 9, v checks whether it has been heard by the sender (line 4). Only in the positive case, v can estimate how far its own alert message could go if it were relayed by that sender (line 5). This task is accomplished by the function `UpdateOutList ()` shown in Fig. 7, whose explanation will follow later. Lines 6 and 7, instead, account for the situation where v lies in between the sender and one of the vehicles the sender has heard. In this case, the *Aware* list of v is updated to include a new entry representing the fact that v lies in between two vehicles, say the sender and vehicle e, where the sender hears e. This may be used in case e is not able to hear the sender directly. In such case, e may learn from v that it can reach the sender. In fact, function `InBetweenl()`, at line 6, checks whether v is in between e and the sender. We then have a second cycle, lines 10 through 15. In the first part of the cycle, at line 11, v checks whether its id_v is contained in the sender *Aware* list. More precisely, let us assume an element of type ($\langle id_r, p_r, f_r, b_r\rangle, id_v$) is found in the sender *Aware* list. In such case, v updates its *Out* list with the information about vehicle r. All this means that, through the information provided by the sender, v becomes aware that it can reach r. In the final part of this cycle, at line 13, the Oracle checks whether the position of v is between the positions of two different vehicles, say s_k and r_k. In the positive case, the *Aware* list of v is updated to incorporate this new information. The impor-

sender	In	Aware
$\langle id, p, f, b\rangle$	$\langle id_i, p_i, f_i, b_i\rangle$ $\langle id_j, p_j, f_j, b_j\rangle$ ...	$\langle\langle id_{r_k}, p_{r_k}, f_{r_k}, b_{r_k}\rangle, id_{s_k}\rangle$ $\langle\langle id_{r_n}, p_{r_n}, f_{r_n}, b_{r_n}\rangle, id_{s_n}\rangle$ ...

Fig. 4. Format of an Oracle message.

```
upon receivingOracleMessage(om)
{merge1(om.sender, this.In);
 for each element e in om.In
 {if(this.id == e.id)
    updateOutList(om.sender);
  if(this.inBetween1(om.sender, e))
   {tmp = list(om.sender, e.id);
    merge2(tmp, this.Aware);
 } }
 for each element k in om.Aware
 {if(k.sender == this.id)
  updateOutList(k.receiver);
  if(this.inBetween2(k.sender,k.receiver))
    merge2(k, this.Aware);
}}
```

Fig. 6. Oracle receiving a message.

```
1 function updateOutList(r)
2 { merge1(r, this.Out);
3   distance = abs(this.p - r.p);
4   if (this.p > r.p)
5     this.b = max(this.b, distance);
6   else
7     this.f = max(this.f, distance);
8   return;
9 }
```

Fig. 7. Oracle updating an *Out* list.

tance of this latter operation touches upon the fact that if, eventually, s_k will receive this Oracle message from v, it will be automatically informed that r_k hears it. Note that this is relevant as, in case of asymmetric communication, s_k could not have been made aware of this before.

For the sake of brevity, we will not provide here details about the `Merge1()`, `Merge2()`, `InBetween1()` and `InBetween2()` functions, as their utility and meaning are straightforward from the specific context we have discussed. It suffices here to say that the results they get can be achieved with the available information.

Instead, we conclude this Section by discussing the `updateOutList ()` function whose code is reported in Fig. 7. The argument of this function, r, represents the tuple of vehicle r: $\langle id_r, p_r, f_r, b_r \rangle$. Note that in the code of Fig. 7 `r.p` represents p_r. The first operation that is carried out, at line 2, is to update the information on r in the *Out* list of v. Clearly, in case r were not already known, a new entry would be created for r. In the opposite case, the data regarding r is simply updated. With lines 3 through 7, the Oracle of v updates its transmission range values. For example, assuming vehicles moving forward along a road, satisfying the condition at line 4 means that vehicle r is behind vehicle v. If the distance of r from v is larger than the previously known backward transmission range (line 5), the new backward transmission range of v is updated accordingly. Lines 6 and 7 perform a similar operation for the frontward transmission range.

4.2. The Relay

We describe the Relay assuming that an Oracle is running at each vehicle. With such assumption, each vehicle has a complete view of the positions and transmission ranges of the vehicles hearing it.

4.2.1. Sending an alert message

When a Relay sends an alert message, say m, it fetches its set of Relays from its *Out* list, appends this list to m and broadcasts the resulting message. A simplified version of these operations is shown in Fig. 8. Function `fetch ()` (omitted for the sake of brevity) returns the sorted list of vehicles *ids*, extracted from the *Out* list. Depending on the direction of propagation, we have two possible orderings. In case the message goes forward, its appended list is sorted according to a decreasing order, as a function of the distance plus the forward transmission range of each entry. Similarly for the backward direction.

4.2.2. Receiving an alert message

When a Relay receives an alert message from another vehicle, it first checks whether it has previously received a copy of the same message (line 2 in Fig. 9). If the message is new, it first delivers it to its application layer (line 3). Then, the Relay searches for its *id* in the Relays list (lines 5 through 7). If the *id* is not in the list, the function halts at line 8. Otherwise, it waits a time proportional to its position in the list (line 9). For example, if the receiver were the first Relay in the list (i.e., `count == 0`), the message would be immediately relayed at line 10. The constant value `W` represents the contention window.

```
upon sendAM(m)
{ Relays = fetch(Out);
  am = append(m, Relays);
  send(am);
}
```

Fig. 8. Relay sending an alert message.

```
1  upon receivingAM(am)
2  {if (! alreadyReceived(am))
3    {deliver(am.m);
4     count = -1;
5     for each vehicle v in am.Relays
6       if (v.id != this.id) count++;
7       else break;
8     if(count==length(am.Relays)-2)exit(1);
9     sleep(W * count);
10    sendAM(am.m);
11   } else
12   {discard(am);
13    abort(AnyInstance(this,am));
14 } }
```

Fig. 9. Receiving and Relaying an alert message.

It may happen that while a vehicle waits for its turn to relay, another vehicle relays the message. In such case the waiting vehicle aborts any hanging sending procedure for that given alert message. The final `else` branch, lines 11 through 14, takes care of this case.

4.3. Time of validity

To avoid data obsolescence, Oracles add a Time Of Validity value, namely *TOV*, to each item in their lists. *TOV* is a positive integer, set to a predefined value every time an item is inserted, or updated, in a list. When an Oracle sends a message, it decreases by one the *TOV* values of all items in its lists. Finally, an entry is deleted when its *TOV* reaches 0.

The *TOV* mechanism is introduced to manage faults and membership variations. If vehicle *a* undergoes a communication fault, or leaves the platoon of connected vehicles, its *TOV* values recorded in the lists of all the vehicles that heard *a* rapidly decrease to zero, eventually causing its removal. On the other hand, a vehicle that joins a platoon, has its information added to the data structures of the vehicles that hear it by simply sending an Oracle message.

The *TOV* mechanism also triggers the procedure that leads to the correct estimation of the actual transmission ranges. In fact, note that the transmission ranges are computed on the basis of the *Out* list content. When an item is deleted from the *Out* list, the backward and forward transmission ranges need to be recomputed.

5. Running example

In the following example we show how the Oracle works when transmission ranges are asymmetric. Fig. 10 shows a situation where vehicle v_4 can hear vehicle v_1, but not *vice versa*. As we shall see, v_1 will eventually learn that it can reach v_4 from in between vehicles.

The left hand side of the figure represents the positions of the vehicles. The right hand side, instead, shows the join of v_1 performed with a combination of six possible consecutive Oracle messages (*om 1*-*om 6*).

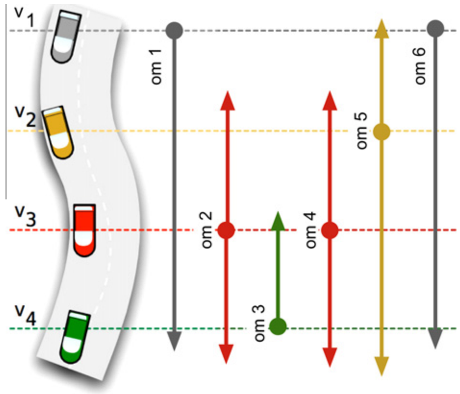

Fig. 10. Vehicle v_1 joins the platoon by sending an Oracle message.

In particular, the positions of the vehicles are marked with horizontal dotted lines, while the origin and the span of an Oracle message are respectively marked with a small circle and a directed link. In Fig. 10, v_1 sends an Oracle message (*om 1*) that reaches v_4.

To simplify the discussion, the shown Oracle messages are numbered based on their order of transmission. Moreover, we suppose that vehicle speeds and transmission ranges are constant. The speeds are such that all vehicles move by one step each time a new Oracle message is sent.

The following discusses how each vehicle modifies its local data structures when receiving an Oracle message.

Message om 1: The Oracle of v_1 sends *Message om 1*, as represented in Fig. 11. Note that the forward and backward transmission ranges are set to 0, since v_1 ignores which vehicles hear it. Upon receiving this message, the vehicles update their local lists as shown in Fig. 12, where the data structure maintained at each vehicle is represented in the rightmost part of the figure. The leftmost part, instead, shows a circled solid arrow which represents the range over which the *Message om 1* extends. In particular, all the vehicles that receive *Message om 1* add the tuple representing v_1 to the local *In* list. Hence, with *Message om 1*, v_2, v_3 and v_4 become aware of v_1.

sender	In	Aware
<1,11,0,0>	∅	∅

Fig. 11. *Message om 1*, sent by v_1.

	Out	In	Aware	P	F	B
v1	∅	∅	∅	11	0	0
v2	<3,30,30,15> <4,40,10,10>	<3,30,30,15> *<1,11,0,0>*	∅	21	20	0
v3	<2,20,25,0> <4,40,10,10> <5,50,10,15>	<2,20,25,0> <4,40,10,10> *<1,11,0,0>*	<4,40,10,10>,2	31	20	10
v4	<3,30,30,15> <5,50,10,15>	<2,20,25,0> <3,30,30,15> <5,50,10,15> *<1,11,0,0>*	<5,50,10,15>,3	41	10	10

Fig. 12. Effects of *Message om 1* on the lists of the receiving vehicles.

Message om 2: Suppose, now, that vehicle v_3 sends *Message om 2*, shown in Fig. 13.

On receiving *Message om 2*, vehicles v_2 and v_4 copy the sender field of the message in their *In* lists (see Fig. 14). The final result on the *In* lists is a new value for the position of v_3. Upon finding their tuples in the *In* field of this message, the Oracles of v_4 and v_2 copy the tuple representing v_3 into their *Out* lists. As an effect of this, v_2 and v_4 know the actual position of v_3.

Furthermore, through *Message om 2* vehicle v_2 discovers it is between v_3 and v_1. In fact, v_1 is found in both the *In* lists of v_2 and *Message om 2*. This means that both v_2 and v_3 have recently received an Oracle message from v_1 (i.e., *Message om 1*). At this point, v_2 adds this information to its *Aware* list.

Message om 3: Vehicle v_4 sends *Message om 3*, whose content is shown in Fig. 15.

Message om 3 is very similar to *Message om 2*, in terms of its effects on the data structures of the receiving vehicles. Fig. 16 shows that the only effect on the *In* and *Out* lists is an update on the position of v_4. An entry is added to the *Aware* list of v_3, when v_3 discovers to be between vehicles v_4 and v_1.

Message om 4: Vehicle v_3 sends *Message om 4*. Figs. 17 and 18 show its content and its effects, respectively. We omit an explanation of the mechanisms that bring to the update of the involved lists, for the sake of brevity. We only highlight the fact that v_2 becomes aware of the fact that v_4 received a message from v_1.

sender	In	Aware
<3,32,30,15>	<2,20,25,0> <4,40,10,10> <1,11,0,0>	<4,40,10,10>,2

Fig. 13. *Message om 2*, sent by v_2.

	Out	In	Aware	P	F	B
v2	*<3,32,30,15>* <4,40,10,10>	*<3,32,30,15>* <1,11,0,0>	*<3,32,30,15>,1*	22	20	0
v3	<2,20,25,0> <4,40,10,10> <5,50,10,15>	<2,20,25,0> <4,40,10,10> <1,11,0,0>	<4,40,10,10>,2	32	20	10
v4	*<3,32,30,15>* <5,50,10,15>	<2,20,25,0> *<3,32,30,15>* <5,50,10,15> <1,11,0,0>	<5,50,10,15>,3	42	10	10

Fig. 14. Effects of *Message om 2* on the lists of the receiving vehicles.

sender	In	Aware
<4,43,10,10>	<2,20,25,0> <3,32,30,15> <5,50,10,15> <1,11,0,0>	<5,50,10,15>,3

Fig. 15. *Message om 3*, sent by v_4.

	Out	In	Aware	P	F	B
v3	<2,20,25,0> *<4,43,10,10>* <5,50,10,15>	<2,20,25,0> *<4,43,10,10>* <1,11,0,0>	*<4,43,10,10>,2* *<4,43,10,10>,1*	33	20	10
v4	<3,32,30,15> <5,50,10,15>	<2,20,25,0> <3,32,30,15> <5,50,10,15> <1,11,0,0>	<5,50,10,15>,3	43	10	10

Fig. 16. Effects of *Message om 3* on the lists of the receiving vehicles.

Message om 5: Vehicle v_2 sends *Message om 5*, as shown in Figs. 19 and 20. Omitting the mechanisms we have already discussed with the previous messages, it is now worth noticing that with this message v_1 learns its forward range and the set of vehicles reached by its transmissions.

Message om 6: Vehicle v_1 sends *Message om 6*, shown in Fig. 21. It is only worth mentioning now that, with this Oracle message, v_2 assesses its backwards transmission range (Fig. 22). This concludes the example.

sender	In	Aware
<3,34,30,15>	<2,20,25,0> <4,43,10,10> <1,11,0,0>	<4,43,10,10>,2 <4,43,10,10>,1

Fig. 17. *Message om 4*, sent by v_3.

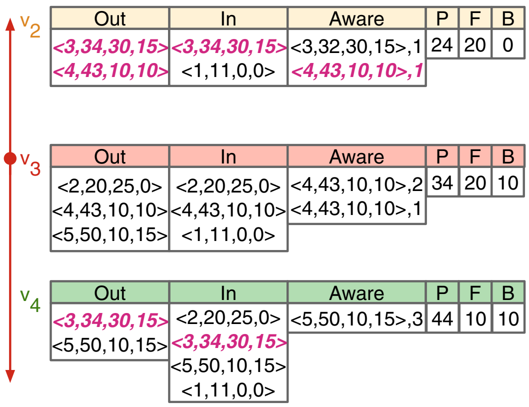

Fig. 18. Effects of *Message om 4* on the lists of the receiving vehicles.

sender	In	Aware
<2,25,25,0>	<3,34,30,15> <1,11,0,0>	<3,32,30,15>,1 <4,43,10,10>,1

Fig. 19. *Message om 5*, sent by v_2.

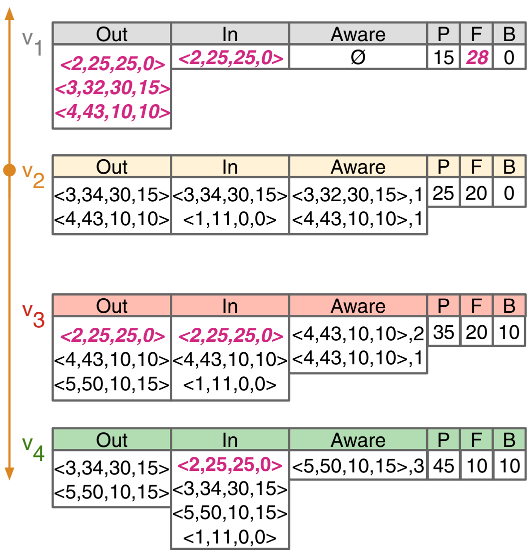

Fig. 20. Effects of *Message om 5* on the lists of the receiving vehicles.

sender	In	Aware
<1,16,28,0>	<2,25,25,0>	Ø

Fig. 21. *Message om 6*, sent by v_1.

6. Performance of the Oracle service

We discuss the performance of the Oracle service we have designed, in terms of time to join and time to live (Section 6.1), and temporal validity of the information recorded at each Oracle and Oracle message sending rate (Section 6.2). In Section 6.3, instead, we provide actual values for the parameters mentioned before. Finally, in Section 6.4 we analyze the overhead due to the Oracle service.

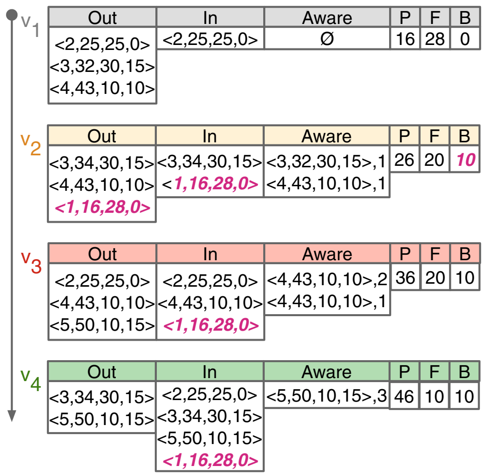

Fig. 22. Effects of *Message om 6* on the lists of the receiving vehicles.

6.1. Join and leave timing

Let us now discuss the time a vehicle needs to join or leave a platoon, through the two following propositions.

Proposition 1. *Assume that vehicle b hears directly from vehicle a (that is b is in the range of a). Take, instead, the case that a needs the intervention of $N-1$ intermediaries to hear from b. Take the minimum frequency $1/T_{max}$, with which the slowest intermediary sends its Oracle messages, then a will become aware of the existence of b after, at most, $(N+1) \times T_{max}$ units of time.*

All this is obvious based on the consideration that the Oracles at different vehicles are not synchronized between each other.

Proposition 2. *Assume that b hears directly a and a hears b (possibly through the help of $N-1$ intermediaries). Assume, also, that a sends its Oracle messages at a frequency of $1/T$. The fact that vehicle b leaves eventually the range of a is decided by a after $TOV \times T$ units of time.*

This holds since *a* decreases by 1 the *TOV* value concerning *b* each *T* time units.

6.2. Choosing T and TOV

We now provide theoretical insights on the value of *T* and *TOV*, through the three following proposition.

Proposition 3. *Given a platoon where vehicle b hears directly a at an initial time $t_0 = 0$ (that is, b received at time t_0 an Oracle message from a). Assume that at t_0 the inter-vehicular distance between a and b be d_0. Assume also that the transmission range of a is R. Take also the difference of speed between a and b equal to Δv. Assume, finally, that at time $t_1 = T$ a subsequent Oracle message arrives at b coming* *from a (that is, with this second Oracle message it is confirmed to b that it is within the reach of vehicle a).*

Then, the maximum time T at which b has to receive the second message is:

$$T = \frac{R - d_0}{\Delta v}. \tag{1}$$

This does not need any proof as b will not be any longer able to hear a after a time which is equal to the ratio between the difference $(R - d_0)$ and their relative speeds Δv.

Proposition 4. *Given the same assumptions taken in Propositions 1 and 2, if we want to give enough time to a, before its information on b expires (that is, before that the recorded TOV at a concerning b goes to 0), the following must hold:*

$$TOV \geqslant \frac{NT_{max}}{T}. \tag{2}$$

This is straightforward based on the consideration that the temporal validity of the information concerning b and recorded at a amounts to $TOV \times T$ units of time, while a new Oracle message coming from b takes at most $N \times T_{max}$ units of time to reach a.

Proposition 5. *Given the same assumption of Proposition 3, if we want that the information concerning b, and recorded at a, do not last longer than the moment in time when b stops hearing a, then the following must hold:*

$$TOV \leqslant \frac{R}{\Delta v T}. \tag{3}$$

This is straightforward considering that the period of time during which b hears a directly is $R/\Delta v$, while the temporal validity of the information concerning b, and recorded at a, is $TOV \times T$.

6.3. Setting the Oracle parameters

We provide now actual values for the T and TOV parameters, respectively. These values are intended to be used on a real implementation of our system on vehicles.

Let us begin with T. In choosing T, we consider a harsh vehicular scenario, given by a great inter-vehicular speed difference and a low range of transmission. Take as reference the IEEE 802.11p standard, which has 300 m as the nominal minimal transmission range value. We half it and take 150 m. As to the relative speed, we take the difference between the maximum and the minimum speeds allowed in most European countries, that is, approximately 50 km/h. Consider that d_0 is negligible with respect to the value chosen for the transmission range, and substitute all this in Eq. 1 of Proposition 3. We get that T equals approximately 10 s. Note that this is obviously a very large value which could be profitably further reduced in realistic conditions.

As to TOV, Propositions 4 and 5 did not give us a precise value to choose but two constraints to satisfy. Let us begin with that of Proposition 5. Considering the same value of R and Δv as before, but exploiting a more realistic and smaller value for T, say 2 s for example, we obtain from Eq. 3 that TOV has to be not larger than 5.

Considering that the ratio $T_{max}/T = 1$ in the case when all vehicles use the same identical value for T, Eq. 2 of Proposition 4 tells us that with a TOV value of 5, a number of intermediaries ranging from 1 to 4 is able on average to keep platoons connected under realistic conditions.

All this is also confirmed by the simulations where we used a more stringent value of 3.

6.4. Oracle overhead

We provide an estimation of the Oracle overhead, in terms of the average number of messages received at a given vehicle and their size. In the following Propositions 6 and 7 we provide information about the two aforementioned figures.

Proposition 6. *The average number of received Oracle messages, per unit time, at a given vehicle, is:*

$$O1 = \frac{2\lambda R}{T}. \tag{4}$$

This is straightforward simply assuming: λ vehicles per meter, an average transmission range of $R/2$ meters and an average sending rate of $2/T$ Oracle messages per second which result from taking range (i.e., R) and Oracle message sending time (i.e., T) as two independent random variables, uniformly distributed in $U[0,R]$ and $U[0,T]$, respectively.

Proposition 7. *Assuming each entry in any of the lists managed by the Oracle is L bytes long, the average size of each Oracle message is bound as follows:*

$$O2 \leqslant \left(\lambda R + \left(\frac{\lambda R}{2}\right)^2 + 1\right)L, \tag{5}$$

where 1 accounts for the size of the sender field in an Oracle message, λR is the average number of entries in the In list and, finally $\left(\frac{\lambda R}{2}\right)^2$ is a bound on the size of the Aware list.

Now, take the following typical figures of an urban scenario, where vehicles travel at 50 km/h along a single lane: $\lambda = 20 \times 10^{-3}$ vehicles per meter, $R = 600$ m, $T = 1.5$ s and $L = 12$ bytes. With these values we get: $O1 = 16$ messages per second and $O2 \leqslant 4704$ bits. Hence, a bound on the total overhead introduced by the Oracle service is 75 kbps.

Consider another realistic case which, instead, touches upon the vehicular flow on an eight-lane highway. Here we typically observe a type of traffic which is more fluid than that experienced in an urban scenario. Accidents, in this situation, are less frequent even if more dangerous. All this considered, we can hence take vehicles running at a 120 km/h speed, with a consequent average inter-vehicular headway distance equal to 100 m. As highway traffic is less nervous than in an urban scenario, we have chosen a realistic value of $T = 5$ s. All this yields: $O1 = 19$, $O2 = 60$ kb and hence $O1 \times O2$ that equals 1.14 Mbps. Considering this in some sense an extreme case (eight-lanes), we can still consider acceptable the obtained overhead.

7. An optimality proof

What we need is to show that our algorithm broadcasts an alert to all vehicles with the minimum number of transmissions, considering without any loss of generality only one direction in the alert propagation flow. The idea is to take the *farthest spanning Relay* algorithm working on a vehicular network and to transform it into an equivalent algorithm running on a Directed Interval Graph (DIG) with weights on edges. Our optimality proof will show that our algorithm is able to compute a shortest path in a weighted DIG, as it corresponds to a minimum number of transmissions in a vehicular network.

To do this we operate two transformations. The former transforms the 1D vehicular network model into a DIG, and the latter translates the *farthest spanning Relay* algorithm into an algorithm P which computes a path between two nodes on a DIG. If a graph G is a DIG deriving from a 1D vehicular network, and a and b are the two farthest connected nodes in G, we will show that P computes a shortest path between a and b. It is also worth noticing that, under such metaphor, the nodes along this shortest path computed by P correspond exactly to the chain of the *farthest spanning Relays* in the original vehicular network.

We model a 1D vehicular network as a series of intervals $\{[x_i,y_i]\}_{i=1,\ldots,n}$, where $x_i < y_i$, x_i represents the position of vehicle v_i, and y_i is the farthest position where it can be heard. In fact, interval $[x_i,y_i]$ describes the span of vehicle v_i. Any vehicle v_j can receive data from vehicle v_i if x_j lies in between x_i and y_i. Based on this model, it is possible to define an interval $[x_i,y_i]$ on the real axis per each element of the series $\{[x_i,y_i]\}_{i=1,\ldots,n}$.

We now transform the series of intervals into a weighted DIG. A DIG is a graph constructed out of a series of intervals $[x_i,y_i]$ as follows. Assuming node i correspond to interval $[x_i,y_i]$, node j correspond to interval $[x_j,y_j]$, and x_j lie in between x_i and y_i, we draw a directed edge from i to j with weight $1/y_j$. This choice is obviously motivated by the fact that in the vehicular network our *farthest spanning Relay* algorithm selects as the Relay, for any given node i, the node k which is able to span farthest (that is y_k). Fig. 23 illustrates this transformation.

We now proceed with the second step.

Take a vehicle v_i and the algorithm that identifies the *farthest spanning Relay* of v_i as v_j, as discussed in Section 3. We now construct an algorithm P, which is equivalent to the *farthest spanning Relay* algorithm, but operates on a weighted DIG. This algorithm is simple and works as follows. It constructs a path from i to its farthest destination node j by selecting at each step that node, among all those which have an incoming edge from i, whose edge has the minimum weight. With the following theorem, we show that that computed by P is exactly the shortest path between i and j.

Theorem 1. *With the aforementioned assumptions, given a weighted DIG G, algorithm P finds the shortest path between i and j.*

Proof. The proof goes through two steps. The first is by induction, the second is simple. Let us now begin with the by-induction proof.

Base case: assume we have a simple DIG where exists only a node i with m different outgoing edges, each directed to a different destination node. The shortest path is constructed out of that single edge which connects i to that destination node having the incoming edge with minimum weight.

Recursive case: Assume that after S iterations a shortest path constructed out of S nodes has been computed, starting from node i to node j. The next node of the shortest path will be chosen as the one that has an incoming edge from j with minimum weight, among all possible neighbors of j.

Hence, this resulting path is composed by two sub-paths. The first of which is shortest by assumption, while the second is the shortest one as the base case has been applied. In conclusion, the path we have computed is the shortest one.

As a second step it needs to demonstrate that j is the farthest node from i, but this is obviously true, as j has been reached by always adding edges with minimum weight, which, under our metaphor, correspond to the largest possible span. □

8. Farthest vs. farthest spanning

We provide further evidence that choosing the farthest spanning Relay instead of the farthest Relay is always better. Indeed, we introduce a figure of merit based on the ratio between the number of hops exploited by the latter method divided by the number of hops required by the former one. We will show that this value is always equal or larger than one. Obviously, under the unrealistic case of transmission ranges equal for all vehicles, this figure amounts exactly to one. In the realistic case, instead, this figure exceeds one, as discussed below. In the following we provide a formal theorem for this and its proof.

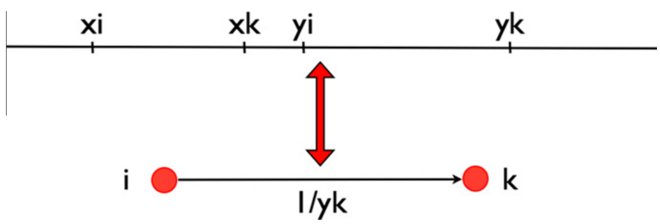

Fig. 23. Modeling two intersecting intervals as a weighted directed interval graph.

Theorem 2. *Take vehicle transmission ranges as independent and identically distributed uniform random variables $R \in U[R_{min}, R_{max}]$. If the portion of considered road is long enough, the ratio, in terms of number of hops, between the farthest Relay and the farthest spanning Relay algorithms is:*

$$\frac{\textit{Farthest Relay Hops}}{\textit{Farthest Spanning Relay Hops}} = \frac{2R_{max}}{R_{max} + R_{min}}. \tag{6}$$

Proof. Consider the farthest Relay algorithm. Each and every vehicle delivers its alert message to its farthest Relay. As transmission range $R \in U[R_{min}, R_{max}]$, that alert will reach, on average, the distance of $\left(\frac{R_{max}+R_{min}}{2} - \Delta\right)$, where Δ is a positive value accounting for the relative position of the Relay. With N Relays, we get $H1 = N \times \left(\frac{R_{max}+R_{min}}{2} - \Delta\right)$.

Take now the farthest spanning Relay algorithm. In this case, the first vehicle will span again its information at $\left(\frac{R_{max}+R_{min}}{2} - \Delta\right)$, instead all other vehicles will be able to span the alert at a distance of $(R_{max} - \Delta)$, thus yielding a total amount of $H2 = \left(\frac{R_{max}+R_{min}}{2} - \Delta\right) + (N-1) \times (R_{max} - \Delta)$ for N Relays. If we divide $H1/H2$ and consider Δ negligible with respect to the value of R_{min}, we get:

$$H3 = \frac{N \times \left(\frac{R_{max}+R_{min}}{2}\right)}{\left(\frac{R_{max}+R_{min}}{2}\right) + (N-1) \times (R_{max})}. \tag{7}$$

Now comes into the picture the assumption that considers cars moving along a sufficiently long portion of road. This assumption mathematically entails that we can manipulate $H3$ as if N were very large (i.e., $N \to \infty$). With this assumption we obtain the result of Eq. 6. □

Fig. 24 compares the experimental results of a simulation we have carried out with the theoretical value of Eq. 6. Each circle in Fig. 24 represents the average over 50 simulation runs, where a platoon of 1000 vehicles moves at the speed of 100 km/h and the source and destination vehicles are 50 km apart and the minimum transmission range R_{min} is 100 m. The motivation behind the choice of 50 km is that of resembling a very long portion of road with a very large number of vehicles (thus satisfying the condition of Theorem 2). The solid line represents Eq. 6. The x-axis of the figure is the R_{max}/R_{min} ratio, while the y-axis accounts for the hop ratio. It is easy to realize that the simulations confirm the results of our theoretical study.

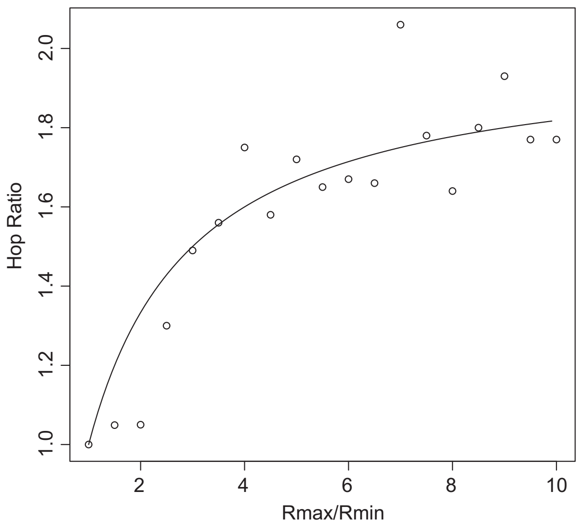

Fig. 24. Farthest vs. farthest spanning.

9. An experimental assessment

The main scope of these experiments is to understand to what extent can our algorithm behave similar to the warning system discussed in Fig. 1.

To this aim, an experimental study has been carried out based on the use of the GTNetS simulator [18]. We exploited the two-ray propagation model at the physical layer and we modified the original MAC layer (802.11b) to make it behave similar to 802.11p, using a fixed 11 Mbps rate.

Clearly, end-to-end delay and the amount of lost alert messages are the main performance metrics to assess the validity of an accident warning system. Hence, Section 9.2 is devoted to analyze these parameters under different realistic conditions.

Since the prominent factor of our system is that of being based on an Oracle service, Section 9.3 measures the performance of our Oracle in terms of its ability in: (a) identifying the farthest spanning Relay, (b) identifying the accurate position of any vehicle involved, and, (c) accurately estimating the transmission ranges of the involved vehicles.

As our proposed accident warning system should work on top of a layer 2 service, we also measured the number of layer 2 collisions that our system suffers while in action. Section 9.4 accounts for this.

Finally, we contrast the performance of our scheme to the performance of the algorithm described in [12] in Section 8.

9.1. Simulation settings

Since it is widely recognized that a high density of vehicles represents one of the most stressful scenarios for broadcast algorithms, we here simulated a highway scenario where the vehicular density is on average of 50 vehicles per kilometer, yielding a platoon of 400 vehicles, initially distributed over 8 km.

At the beginning of the simulation the vehicles were placed following a uniform distribution, therefore the average initial distance between two vehicles was of 20 m, on average.

The vehicle speed distribution was defined following [22]: 50% of vehicles moved at a constant speed V = 30 m/s, 15% moved at a constant speed randomly chosen in $U[V+4, V+8]$, 10% in $U[V+8, V+12]$, 15% in $U[V-8, V-4]$ and 10% in $U[V-12, V-8]$ m/s.

Forward and backward transmission ranges were randomly and independently drawn from the $U[100, 600]$ m interval.

Alert messages were of different sizes and sent at different frequencies. The frequencies were chosen as follows. We supposed to use two different cases: the *Lazy* and the *Intensive*. In the *Lazy* one, a source immediately sends a first message and then pauses before the transmission of any successive alerts. The pause time is randomly chosen between 1.0 and 1.5 s. The *Intensive* one is similar to the *Lazy* one, except for the fact that it pauses for a period in time randomly ranging from 0.5 to 0.75 s.

We used two possible message sizes: 1 KB long messages, termed *Slim*, and 2 KB long messages, termed *Fat*. Therefore, we got four different scenarios: Lazy–Slim (L–S), Lazy–Fat (L–F), Intensive–Slim (I–S), and Intensive–Fat (I–F).

We analyzed 6 different cases for each of which the number of vehicles issuing an alert were, respectively, 1, 20, 40, 60, 80 and 100.

We run 10 different simulations for each of the 24 scenarios we have described before. Confidence intervals (5%–95%) are not represented in the following figures, as they are very close to the average values computed.

9.2. Accident warning system performance

End-to-end Delay: The broadcast delay was computed observing two reference vehicles at the opposite ends of the platoon and measuring the time elapsed for a message generated at one end to reach the other end. Fig. 25 shows a bar diagram of the average latencies, in milliseconds, with respect to the four scenarios of application, as a function of the number of senders. The most prominent fact to notice in Fig. 25 is that the amount of time needed to span an alert message all, over the platoon, is almost independent of the number of considered scenarios and of the number of sources. Furthermore, it never exceeds 150 ms.

Fault tolerance: Unfortunately, reality is not always perfect and faults may occur. We carried out experiments to measure the impact of faults intended as Relays that abort and do not perform their duty. Specifically, Fig. 26 shows how the end-to-end value increases if the three best Relays fail in retransmitting any given alert message. It is worth noticing that, even in this stressful condition, the end-to-end delay is almost independent of the number of possible sources, and its increment never exceeds 36%.

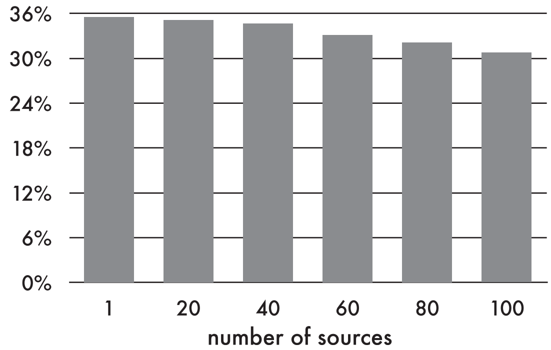

Fig. 26. End-to-end delay increase in presence of faults.

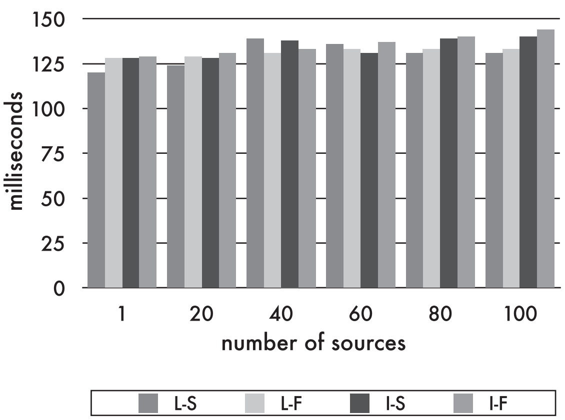

Fig. 25. End-to-end delay.

Tunnel: We tried to take into account one of the harshest situations that may occur to a platoon of cars: entering a tunnel. We simulated a 1 km long tunnel. Transmission ranges of vehicles inside the tunnel are simply halved. Instead, a car entering the tunnel has halved only its forward transmission range. Conversely, a car leaving a tunnel has halved only its backward transmission range. Fig. 27 again shows that the overall delay is independent of the number of sources, while its overall increment, due to the passage inside the tunnel, is roughly 20%.

Fraction of Lost Messages: An alert message is defined to be lost when even one vehicle does not receive it. Fig. 28 shows the percentage of lost messages due to multiple possible causes, as a function of the number of sources. The first good news is that the percentage of lost messages we experienced never exceeds 3%. Then, not surprisingly, this percentage of lost messages increases with the number of sources. Finally, it is worth noticing that the I–F scenario is the most prone to lost messages.

9.3. Oracle performance

Identifying Relays: In Section 3, we explained that the Oracle sub-layer produces a list of Relays candidate to

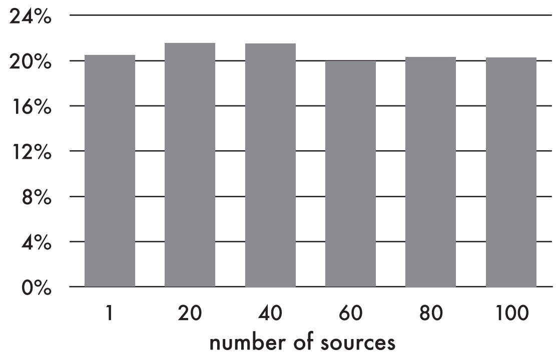

Fig. 27. End-to-end delay increase due to a tunnel.

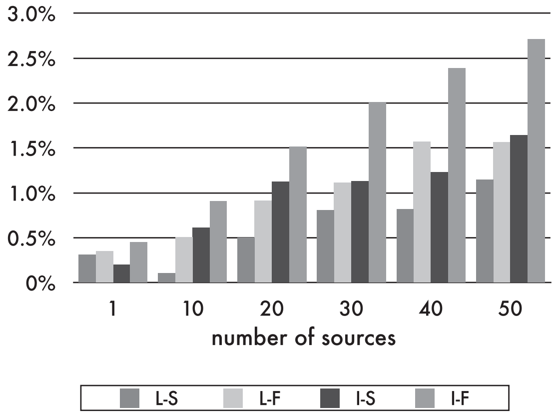

Fig. 28. Percentage of lost alert messages.

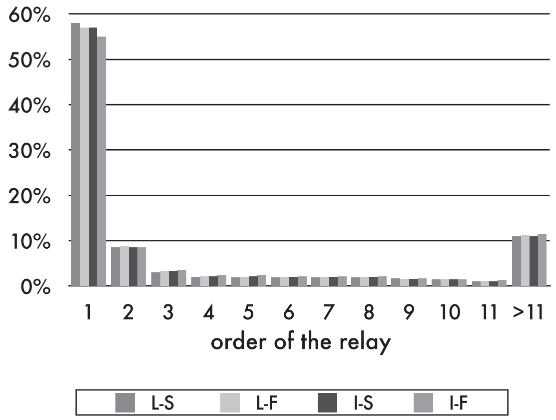

Fig. 29. The actual Relay of alert messages.

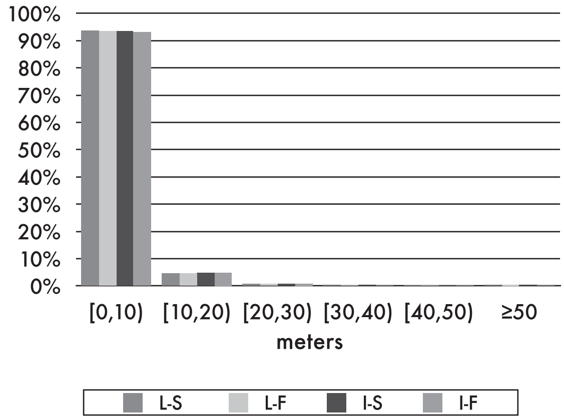

Fig. 30. Percentage of vehicles whose position is identified with an error of $[x,y]$ meters.

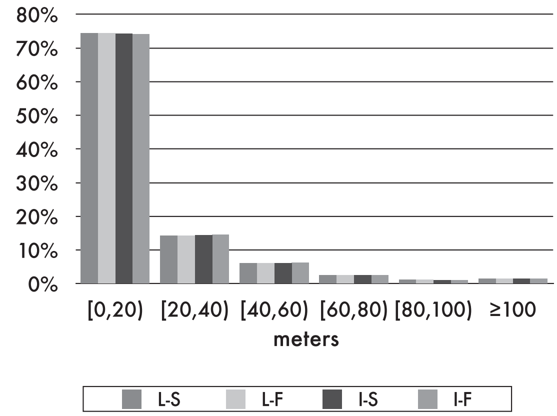

Fig. 31. Percentage of vehicles whose transmission range is estimated with an error of $[x,y]$ meters.

forward an alert ordered on the basis of their capacity to span farther. Again, in the real world there is no guarantee that the Relay sub-layer be able to keep its promise. In Fig. 29 we show the percentage of times the *i*th Relay in the list forwards an alert message. The fact that almost the 60% of the alert messages are forwarded by the first ranked Relay confirms the validity of our mechanism, even in the real case.

Accuracy: Figs. 30 and 31 confirm that our Oracle works very accurately. In fact, in more than the 90% of cases it accurately identifies the positions of all vehicles involved (Fig. 30), and in more than the 70% of cases it accurately estimates the range of transmissions of all the vehicles involved. On the x-axis we report the absolute errors on these estimates.

As a final consideration of this section we also mention that we carried out simulative experiments to assess the size of Oracle messages. We do not report them here for the sake of brevity, we simply remind that the obtained outcomes fully confirm the result of the example discussed in Section 6.4 (that is, Oracle messages have sizes included in the [2,4] KB interval).

9.4. Layer two collisions

Collisions: We measured the overall number of layer 2 collisions as a function of the number of sources. Not surprisingly, as we deal with a deterministic algorithm, the number of collisions is very low ($\leqslant$5% of the total amount of the Relay messages). Obviously, this figure increases with the number of sources (Fig. 32).

9.5. Comparison

At this final point, it is time to compare our proposal with an existing solution, in terms of broadcast speed. Nonetheless, to provide a fair comparison, we selected a competing protocol using the following criteria: (a) the competing protocol aims at minimizing the number of transmissions required to disseminate an alert message under the assumption that transmission ranges vary from vehicle to vehicle, and, (b) the competing protocol was known to be the one which offered the best performance so far. We found an ideal candidate in the work

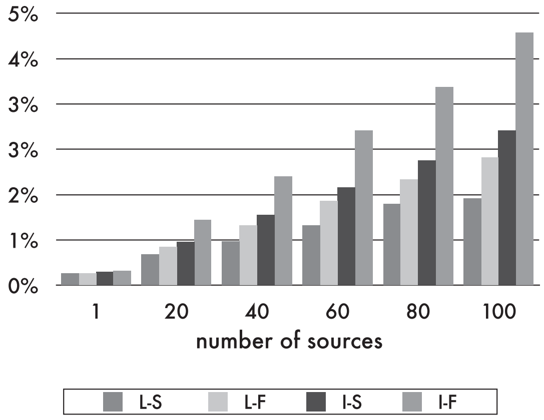

Fig. 32. Message collisions.

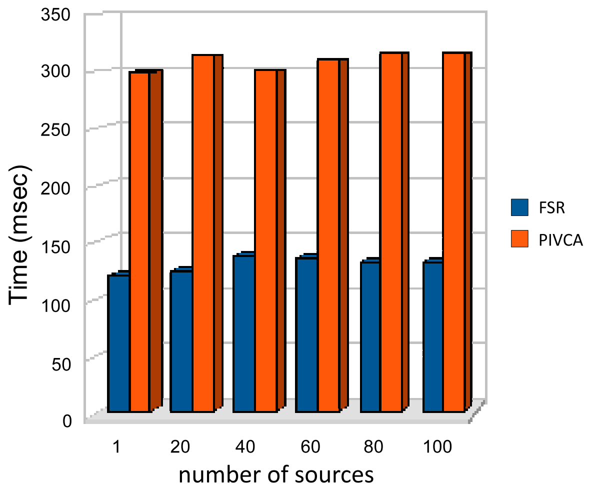

Fig. 33. Broadcast speed comparison.

presented in [12], termed Privileged Inter-Vehicular Communication Architecture (PIVCA), as it introduces an automatic transmission range estimator which is exploited to estimate the actual transmission range experienced by a vehicle in a platoon, providing better performances than those given by schemes that assume transmission ranges to be fixed and known [8], and by flooding based ones as well. Fig. 33 reports a comparison of the end-to-end values obtained utilizing the two protocols, clearly showing that ours, here termed farthest spanning Relay (FSR), requires less than half of the time required by its competitor to spread an alert message to the entire platoon of vehicles.

10. Conclusion

We have presented a distributed multi-hop broadcast algorithm for vehicular safety [23]. Formal proofs and experimental results that confirm the optimality of our approach have been provided.

Acknowledgements

We thank the italian FIRB DAMASCO project for funding this work.

References

[1] L. Neubert, L. Santen, A. Schadschneider, M. Schreckenberg, Single-vehicle data of highway traffic: a statistical analysis, Physical Review E (1999) 6480–6490.

[2] C.-G. Wallman, H. Astrom, Friction measurement methods and the correlation between road friction and traffic safety, Swedish National Road and Transport Research Institute, Technical Report, 2001.

[3] M. Bilban, A. Vojvoda, J. Jerman, Age affects drivers' response times, Journal Collegium Antropologicum (2009) 467–471.

[4] R. Akcelik, On the validity of some traffic engineering folklore, in: Proceedings of 22nd Conference of Australian Institutes of Transport Research, December 2000.

[5] A. Amoroso, G. Marfia, M. Roccetti, A Distributed multihop broadcast algorithm for vehicular safety, University of Bologna, Computer Science Technical Report.

[6] C. Wewetzer, M. Caliskan, K. Meier, A. Luebke, Experimental evaluation of UMTS and wireless LAN for inter-vehicle communication telecommunications, in: Proceedings of 7th International Conference on ITS, August 2007, 1-6.

[7] N. Wisitpongphan, O.K. Tonguz, J.S. Parikh, P. Mudalige, F. Bai, V. Sadekar, Broadcast storm mitigation techniques in vehicular ad hoc networks, IEEE Wireless Communications 14 (6) (2007) 84–94.

[8] A. Zanella, G. Pierobon, S. Merlin, On the limiting performance of broadcast algorithms over unidimensional ad-hoc radio networks, in: Proceedings of Wireless Personal Multimedia Communications, 2004, pp. 165–169.

[9] E. Fasolo, R. Furiato, A. Zanella, Smart broadcast algorithm for inter-vehicular communication, in:Proceedings of 2005 Wireless Personal Multimedia Communication, September 2005.

[10] J.J. Blum, A. Eskandarian, A reliable link-layer protocol for robust and scalable intervehicle communications, IEEE Transactions on Intelligent Transportation Systems 8 (1) (2007) 4–13.

[11] G. Korkmaz, E. Ekici, F. Ozguner, U. Ozguner. Urban multi-hop broadcast protocol for inter-vehicle communication systems, Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, October 2004, pp. 76–85.

[12] C.E. Palazzi, S. Ferretti, M. Roccetti, An inter-vehicular communication architecture for safety and entertainment, IEEE Transactions on Intelligent Transportation Systems 11 (1) (2010) 90–99.

[13] F. Ye, M. Adams, S. Roy, V2V wireless communication protocol for rear-end collision avoidance on highways, in: Proceedings of IEEE 2008 Vehi-Mobi Workshop, May 2008, pp. 375–379.

[14] S. Biswas, R. Tatchikou, F. Dion, Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety, IEEE Communication Magazine 44 (1) (2006) 74–82.

[15] R.A. Santos, R.M. Edwards, A. Edwards, Cluster-based location routing algorithm for vehicle to vehicle communication, in: Proceedings of 2004 IEEE Radio and Wireless Conference, September 2004, pp. 39–42.

[16] S. Min-Te, F. Wu-Chi, L. Ten-Hwang, K. Yamada, H. Okada, K. Fujimura. GPS-based message broadcasting for inter-vehicle communication, in: Proceedings of 2000 IACC International Conference on Parallel Processing, August 2000, pp. 279–286.

[17] H. Alshaer, E. Horlait, An optimized adaptive broadcast scheme for inter-vehicle communication, in: Proceedings of IEEE 61st Conference on Vehicular Technology, June 2005, pp. 2840–2844.

[18] GTNetS. <http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/>.

[19] D. Jiang, L. Delgrossi, IEEE 802.11p: Towards an international standard for wireless access in vehicular environments, in: Proceedings of the 68th Semi-Annual Vehicular Technology Conference, 2008, pp. 2036–2040.

[20] ASTM E2213 – 03 standard specification for telecommunications and information exchange between roadside and vehicle systems 5 GHz band dedicated short range communications (DSRC) medium access control (MAC) and physical layer (PHY) specifications. <http://www.astm.org/Standards/E2213.htm>.

[21] L. Cheng, B.E. Henty, D.D. Stancil, F. Bai, P. Mudalige, Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 GHz dedicated short range communication (DSRC)

frequency band, IEEE Journal on In Selected Areas in Communications Selected Areas in Communications 25 (8) (2007) 1501–1516.
[22] P. Dey, S. Chandra, S. Gangopadhaya, Speed distribution curves under mixed traffic conditions, Journal of Transportation Engineering 132 (6) (2006) 475–481.
[23] A. Amoroso, M. Ciaschini, M. Roccetti, The farther Relay and Oracle for VANET. Preliminary results, in: Proceedings of the 4th Annual International Conference on Wireless internet, November 2008, pp. 4859:1–4859:7.

Alessandro Amoroso got his Laurea degree in Physics at the University of Bologna in 1987. He is currently an Associate Professor at the Department of Computer Science of the University of Bologna. His research interests include: distributed systems, multimedia systems and mobile devices.

Gustavo Marfia got his Laurea degree in Telecommunications Engineering at the University of Pisa in 2003. He received a Ph.D. in Computer Science from the University of California, Los Angeles, in 2009. He is currently a Postdoctoral Researcher at the Department of Computer Science of the University of Bologna. His research interests include: ad hoc networks, transportation systems and multimedia streaming.

Marco Roccetti received the Italian Laurea degree in electronic engineering from the University of Bologna, Bologna, Italy. He has been a Full Professor with the Dipartimento di Scienze dell'Informazione, University of Bologna since 2000. His research interests include digital audio and video, computer entertainment, and web-2.0-based applications, fields in which he has authored almost 200 technical refereed papers.