An Optimal 1D Vehicular Accident Warning Algorithm for Realistic Scenarios

Marco Roccetti, Gustavo Marfia, Alessandro Amoroso
Dipartimento di Scienze dell'Informazione
Università di Bologna
40126 Bologna, Italia
Email: {roccetti, marfia, amoroso}@cs.unibo.it

***Abstract*—With the development of vehicular networks, it is possible to implement accident warning systems that directly (i.e., without any external supporting infrastructure) alert approaching vehicles. In this scenario, each vehicle is provided with a sensor that, under abnormal conditions, triggers an alarm message. A vehicle that receives this information: (a) warns the driver; and (b) relays such information to following vehicles for safety purposes. We here present an optimal algorithm for one dimensional (1D), multi-lane, strip-shaped portions of roads. It guarantees alarm messages are received along the minimum hop path, even when communications are asymmetric and transmission ranges vary from vehicle to vehicle. To the best of our knowledge, this is the first algorithm capable of reaching such performance under realistic wireless propagation assumptions.**

***Keywords*-VANET; Accident; Broadcast; Farthest relay**

I. Introduction

Important contributing circumstances in highway pileups are unexpected and abnormal events such as queues due to congestion and ahead crashes. Very recently, in July 2009 a pileup involving 259 vehicles occurred near Hannover in Germany [1]. Presumably, most vehicles could have avoided their involvement if they were aware of the event. To reach this goal, the deployment of 802.11p based [2] Vehicular Networks (VANETs) and sensor technologies could be put to good use to spread an accident warning in a few seconds. With such technologies, an alarm message, sent from a vehicle to another vehicle, could easily reach all cars in the area surrounding an accident.

While the 802.11p draft supports the broadcast of alarm messages on separate, dedicated channels, the draft does not suggest any algorithm to compute the relays that should forward an alarm message. To this aim, an extensive body of literature exists discussing algorithms that search suboptimal broadcast strategies on portions of roads including intersections (2D). This is equivalent to the search of the *maximum connected dominating set* on a planar space. Unfortunately, such problem is NP hard and, therefore, only solvable in exponential time. Fortunately, an optimal solution to this problem can be found in polynomial time when vehicles are driven along a multi-lane, strip-shaped portions of roads (1D). This is the typical realistic case with highways.

Even in the 1D case, although clear that choosing the farthest vehicle as an alarm message relay at each hop provides the minimum number of hops when inter-vehicular transmission ranges are equal and constant, such strategy may not work when applied to realistic settings where such assumptions do not hold. In fact, in realistic settings transmission ranges could be different and vary due to a number of reasons which include different hardware settings, multi-path propagation and weather conditions. It is a matter of fact, at least to the best of our knowledge, that no algorithm has been provided in literature which is capable of handling realistic situations while guaranteeing the minimum number of hops in the spread of alarm messages.

Our contribution to this research field has been that of filling this gap. In this paper we present a new inter-vehicular broadcast-based accident warning algorithm and prove it is optimal for vehicles being driven along multi-lane, strip-shaped portions of roads (1D). We prove our algorithm is optimal in terms of the number of hops needed to reach even far vehicles, accounting for the influences of realistic effects, such as variable and asymmetric transmission ranges. The motivation at the basis of the optimality of our algorithm relies upon the fact that 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 proof of its optimality, we also present simulation results which confirm the validity of our algorithm.

The paper is organized as follows. In Section II we provide a brief summary of the literature available on the subject. Our algorithm is discussed at a high level in Section III, while more in depth in Section IV. A formal proof of its optimality is provided in Section V. Simulation results are presented in Section VI and, finally, we conclude with Section VII.

II. Related Work

The problem of broadcasting an alarm message in a VANET is interesting for two main reasons. The first is that, due to the high density of vehicles in certain highway areas, an excessive number of forwarders could cause layer

978-1-4244-7755-5/10/$26.00 ©2010 IEEE

two collisions, and, therefore, cause the loss or the delay of alarm messages. The second is that the difference of speeds between vehicles can be high, and, therefore, inter-vehicular contact times can be very low. Because of these reasons it is necessary to design algorithms capable of choosing the best forwarders. Fortunately, in a highway scenario, a vehicle is constrained to move over a linear, unidimensional (1D) topology, constraint which has been frequently put to good use in literature to design efficient accident warning systems.

One of the most widely exploited ideas to minimize the number of transmissions required to inform a platoon of vehicles is that of spreading an alarm message by having the farthest forwarder forward it at each hop. Authors of [7], for example, devise an original solution to let a vehicle recognize if it is the farthest relay. In brief, by sending a jamming signal of length directly proportional to its distance from the sender after receiving an alarm message, a receiver that finds the communication channel free assumes it is the farthest vehicle from the sender, and, therefore, its relay. Another scheme that follows the same lines is devised in [5], [6]. When receiving a message, a vehicle waits for a time which is inversely proportional to its distance from the sender. In case no duplicate message has been received by the end of this time, the vehicle forwards the message.

All the aforementioned schemes are affected by the assumption that a unique, constant, and well known transmission range is exploited for all vehicles in every moment. Authors of [8] fix this problem by devising an algorithm that assesses the actual transmission range for every car in the platoon. A result of this algorithm are in general lower delays in forwarding alarm messages.

We conclude this Section observing that the approaches here presented ignore that, due to the highly stochastic nature of a vehicular wireless channel, transmission ranges are variable in space and time and transmission channels are asymmetric. In the following, we aim at filling this gap.

III. The Idea

In a realistic setting, it is easy to understand that, in a given direction, the best relay of a vehicle does not coincide with the most distant vehicle in the same direction. In fact, in a situation where transmission ranges can vary, it may happen that the most distant vehicle, and, therefore, the *farthest relay*, has a very short transmission range. Instead, it may happen that another closer vehicle in the same direction, has, unfortunately, a transmission range that exceeds the segment spanned by the transmission range of the most distant vehicle. In such a case, this second vehicle would be the best relay. We then have that, in general, the best relay does not coincide with the *farthest relay*.

The focus of this work, then, is to find the best relay, which is the *farthest spanning relay*. Let us begin with an intuitive definition of what a *farthest spanning relay* is. Indeed, it is that vehicle that is able to span farthest among all other vehicles. Although very simple to define, it may be not trivial to identify the *farthest spanning relay* for the following reason. In fact, due to the variability of transmission ranges, it may happen that a vehicle hears the other, but not vice versa. Therefore, it may be the case that a vehicle is not able to hear, and, therefore, discover directly its *farthest spanning relay*. For this reason we have devised an original mechanism able to permit to a vehicle to become aware of the existence of its *farthest spanning relay* (while the *farthest spanning relay* hears it). This is performed through a distributed Oracle service, which builds, when this is possible, a communication chain between a vehicle and its *farthest spanning relay* through the vehicles that are between them in the case where that given vehicle and its *farthest spanning relay* are not directly and mutually aware of each other. A Relay service, instead, uses the information provided by the Oracle to disseminate alarm messages along a series of subsequent *farthest spanning relays*.

IV. Implementation

The mechanism that identifies the *farthest spanning relay* at each vehicle is composed of two functional blocks which deal with both variable and asymmetric transmission ranges: an Oracle and a Relay.

The Oracle runs in the background and provides a vehicle with: (a) an estimate of its current position; (b) its backward and forward transmission ranges; and, (c) the positions and transmission ranges of the surrounding vehicles in range.

The Relay computes an ordered list of relays which is piggybacked in all sent broadcast messages. The list is ordered in terms of the positions and transmission ranges of the vehicles in range, beginning with the vehicle with the highest *(distance + transmission range)*. In brief, on receiving this list a vehicle waits for a time interval which is related to its position in the list, before relaying the packet. The farther down a vehicle is in the list, the longer it will wait to transmit.

From now on, we will distinguish application or alarm messages, the messages that carry application data and which are managed by the Relay, from control or Oracle messages, the messages which originate from and end at the Oracle module. Although the system may be implemented with messages originating at different layers of the networking stack, we place them at the application layer in our design.

In the following we provide a brief explanation of how the Oracle and the Relay work, more insights may be found in [4].

A. The Relay

When a vehicle receives a new alarm message (i.e., never seen before) this is managed by its Relay. First, it strips the list of relays piggybacked in the alarm message. Second, it finds the position of its vehicle in the list and, based on it, it

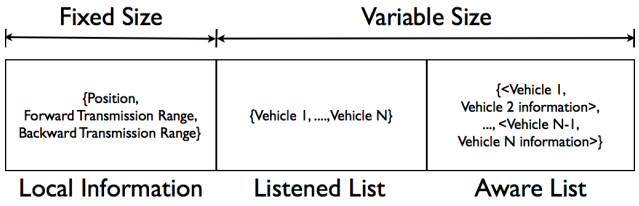

Figure 1. Oracle message structure.

computes the time interval it should wait before relaying the message. Third, if the wait time elapses without any other copy of the message being seen (in such case the message can be discarded), it reads the ordered list of the best relays from the Oracle (we will see this information is stored in the *Reached* list at the Oracle), and appends it to the alarm message. Finally, it sends the message.

We now move on to explain how the *Oracle* fills the *Reached* list which is used by the *Relay*.

B. The Oracle

Our algorithm devises the cooperation of intermediate vehicles to cope with varying and asymmetric transmission ranges. Such strategy is implemented maintaining three data structures, the *Listened*, *Reached* and *Aware* lists. The three data structures are updated exchanging local information via Oracle messages. In brief, Oracle messages are variable size control messages periodically broadcast by all vehicles. Their structure is shown in Figure 1. If a vehicle lacks any knowledge of the vehicles which surround it, it sends an Oracle message which contains its position and its forward and backward transmission ranges set to zero. On receiving such type of Oracle message a vehicle updates its *Listened* data structure, merging the address of the vehicle that sent the message in it (and saving its current position), just as shown in Step 1.

```
   input : Vehicle B receives an empty Oracle
           Message (OM) from vehicle A.
   output: Updated Vehicle B Listened list (VB_LL)
           local data structure.
1.1 VA.Addr ← GetSenderAddr (OM);
1.2 VA.Pos ← GetSenderPos (OM);
1.3 VB_LL ← Merge (VB_LL, VA.Addr);
```

Step 1: Vehicle B receives an Oracle message which contains the position of vehicle A.

In general Oracle messages may contain:

1) The position of a vehicle;
2) The position, the transmission ranges of a vehicle and its *Listened* data structure;
3) The position, the transmission ranges of a vehicle, its *Listened* and its *Aware* data structures.

Assuming now that vehicle B receives a message containing a *Listened* list from vehicle A, the Oracle of vehicle B first performs Step 1 (additionally it updates its knowledge of the forward and backward transmission ranges of vehicle A), and then inspects the *Listened* list. If it finds its address in the list, the address and the transmission ranges of A are merged into the local *Reached* list. Moreover, if it determines that B is in an intermediate position between vehicle A and a vehicle C, listed in the *Listened* list, the tuple $< C, A >$ is merged into the local *Aware* list. In this context, a tuple $< C, A >$ means that vehicle A can hear vehicle C. The pseudocode for such behavior can be found in Step 2.

```
   input : Vehicle A Listened list (VA_LL)
   output: Updated Vehicle B Aware list (VB_AL)
           and Vehicle B Reached list (VB_RL) local
           data structures
2.1 if VB.Addr ∈ VA_LL then
2.2 |  VB_RL ← Merge (VB_RL, VA.Addr);
2.3 end
2.4 foreach V ∈ VA_LL do
2.5 |  if VB.IsInBetween (V, VA) then
2.6 |  |  VB_AL ← Merge (VB_AL, < V, VA >);
2.7 |  end
2.8 end
```

Step 2: The Oracle Module of vehicle B processes the *Listened* list received from vehicle A.

The last step is performed when a vehicle also receives an *Aware* list. An *Aware* list is bounded to be received together with a *Listened* list, so we here assume a vehicle first runs Steps 1 and 2. There are three possible options per each tuple in the *Aware* list: (a) the vehicle is the sender; (b) the vehicle is between the sender and the receiver, and; (c) the vehicle is not the sender and is not in between the sender and the receiver. Let us now assume the *Oracle* module is processing the tuple $< C, A >$. If vehicle C receives it, it adds vehicle A to its local *Reached* list. If vehicle B, a vehicle between vehicle C and vehicle A, receives it, it merges the tuple $< C, A >$ into its local *Aware* list. Finally, if option (c) occurs, the tuple is simply ignored. All this is coded in Step 3.

Figure 2 summarizes the steps that have been described so far. Vehicle A can hear vehicle C, but not vice versa, so an intermediate vehicle B cooperates to make this known to vehicle C. The *Reached* list at each vehicle contains the positions and the transmission ranges of the vehicles which can hear it. The information stored in this data structure is used to write the list piggybacked to all alarm messages. Therefore, the algorithm converges when the *Reached* lists at all vehicles are updated and stable.

```
input : Vehicle B Aware list (VB_AL)
output: Updated vehicle C Aware list (VC_AL)
        and vehicle C Reached list (VC_RL) local
        data structures.
3.1 foreach < VX, VY >∈ VB_AL do
3.2   if VX.Addr = VC.Addr then
3.3     VC_RL ← Merge (VC_RL, VY.Addr);
3.4   end
3.5   else if VC.IsInBetween (VX, VY) then
3.6     VC_AL ← Merge(VC_AL, < VX, VY >);
3.7   end
3.8 end
```

Step 3: The Oracle at vehicle C processes the *Aware* list received from vehicle B.

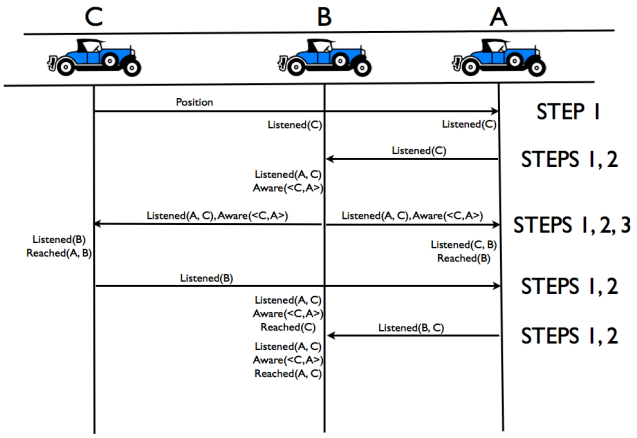

Figure 2. Example of the interactions between three vehicles. A and B can hear their entire neighborhood, but C cannot hear A. Therefore, B cooperates informing C. The Steps performed at each oracle message receipt are shown on the right hand side.

V. An optimality proof

What we need is to show that our algorithm broadcasts an alert to all vehicles with the minimum number of transmissions. 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 graph 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.

We make two transformations. The first transforms a 1D vehicular network model into a Directed Interval Graph (DIG), and the second translates the *farthest spanning relay* algorithm into an algorithm A which computes a path between two nodes in 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 A computes a shortest path between a and b. In this way, the nodes along the shortest path computed by A correspond exactly to the chain of the *farthest spanning relays* in the original vehicular network.

Transformation 1: We can 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. Therefore, 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, we define an interval $[x_i, y_i]$ on the real axis per each element of the series $\{[x_i, y_i]\}_{i=1,\ldots,n}$. A weighted 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 follows from the fact that in a 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).

Transformation 2: Consider a vehicle v_i and the algorithm that identifies the *farthest spanning relay* of v_i as v_j, as discussed in Section III. We now construct an algorithm A, which is equivalent to the *farthest spanning relay* algorithm, but operates on a weighted DIG. This algorithm 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 the path computed by A is exactly the shortest path between i and j.

Theorem V.1 *Based on the preceding assumptions, algorithm A finds the shortest path between i and j on a weighted DIG G.*

Proof: We conduct our proof in two steps. The first is by induction.

Step 1: The base case assumes that 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 in this case results from taking that single edge which connects i to that destination node having the incoming edge with minimum weight. The recursive case assumes that after S iterations a shortest path constructed out of S nodes has been computed, say 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.

Step 2: At this point we want to prove that j is the

farthest node from i. This is true, as j has been reached by always adding edges with minimum weight, which, under transformation, correspond to the largest possible span. ■

VI. Simulations

An experimental study has been carried out based on the use of the GTNetS simulator [9]. 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 11Mbps rate. The main objectives of our simulations are to (a) assess the validity of the accident warning system in Subsection VI-B, and; (b) measure the performance of the Oracle in Subsection VI-C.

A. Simulation Settings

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 kilometers. 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 meters, on average. The vehicle speed distribution was defined following [10]: 50% of vehicles moved at a constant speed $V = 30$ meters per second, 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 $U[V-12, V-8]$, meters per second. Forward and backward transmission ranges were randomly and independently drawn from the $U[100, 600]$ meters 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 seconds. 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 seconds.

We used two possible message sizes: 1KB long messages, termed *Slim*, and 2KB 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.

B. 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. Figure 3 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 Figure 3 is that the amount of time needed to broadcast the message to all the platoon never exceeds 150 milliseconds.

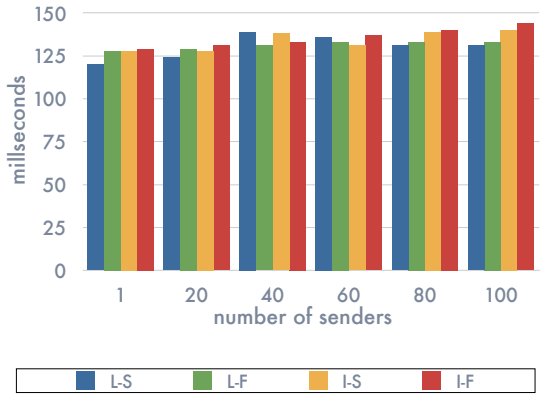

Figure 3. End-to-end delay.

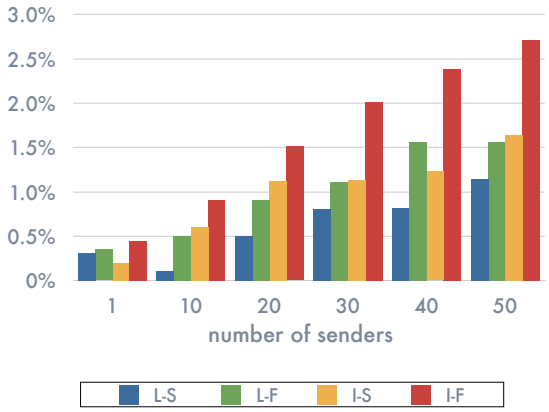

Figure 4. Percentage of lost alarm messages.

Fraction of Lost Messages: An alarm message is defined to be lost when even one vehicle does not receive it. Figure 4 shows the percentage of lost messages, as a function of the number of sources. This value never exceeds 3%. Not surprisingly, the percentage of lost messages increases with the number of sources.

C. Oracle performance

Identifying Relays: In Section III, we explained that the Oracle sub-layer produces a list of Relays candidate to forward an alert ordered on the basis of their capacity to span farther. In the real world there is no guarantee that the Relay sub-layer will be able to keep its promise. In Figure

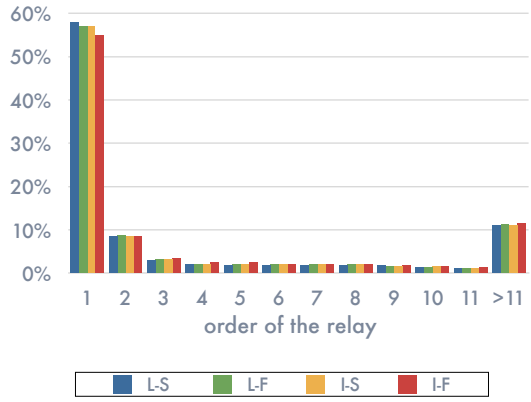

Figure 5. The actual relay of alarm messages.

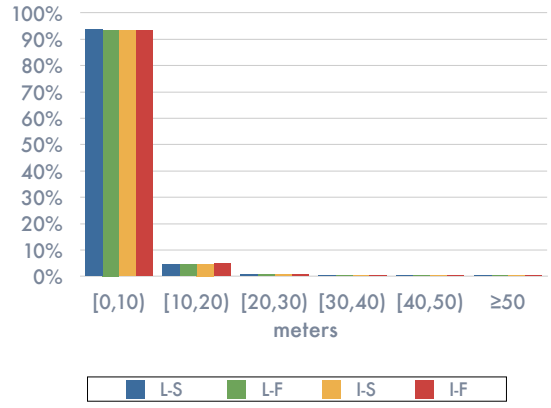

Figure 6. Percentage of vehicles whose position is identified with an error of [x, y] meters.

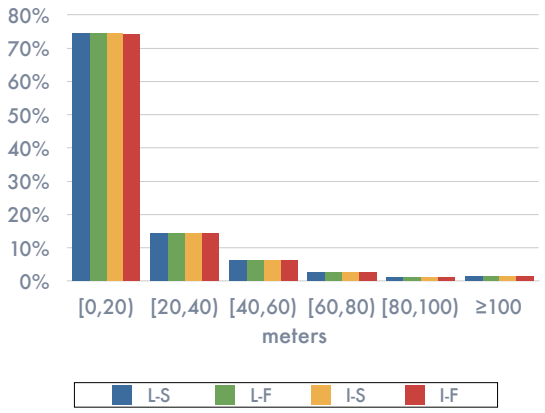

Figure 7. Percentage of vehicles whose transmission range is estimated with an error of [x, y] meters.

5 we show the percentage of times the i-th relay in the list forwards an alarm message. The fact that almost the 60% of the alarm messages are forwarded by the first ranked relay confirms the validity of our mechanism, even in the real case.

Accuracy: Figures 6 and 7 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 (Figure 6), 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.

VII. Conclusion

We propose an optimal algorithm for broadcasting vehicular alerts in highway scenarios. Its optimality has been shown in terms of the hops required to spread an alarm message and its performance has been studied within a realistic simulative scenarios. To the best of our knowledge no other algorithm achieves comparable performance, again, under realistic conditions.

References

[1] http://news.bbc.co.uk/2/hi/europe/8159290.stm

[2] http://grouper.ieee.org/groups/802/11/Reports/tgp_update.htm

[3] A. Amoroso, M. Ciaschini, M. Roccetti. "The farther relay and oracle for VANET. preliminary results". *Proceedings of the 4th Annual international Conference on Wireless internet (Maui, November 17 - 19, 2008).*

[4] A. Amoroso, G. Marfia, M. Roccetti. "Going Realistic and Optimal: A Distributed Multi-Hop Broadcast Algorithm for Vehicular Safety". UniBo Technical Report.

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

[6] J. J. Blum, A. Eskandarian, "A reliable link-layer protocol for robust and scalable intervehicle communications", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 8, N. 1, March 2007, 4-13.

[7] 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, 76-85.

[8] C.E. Palazzi, S. Ferretti, M. Roccetti, "An Inter-Vehicular Communication Architecture for Safety and Entertainment", *IEEE Transactions on Intelligent Transportation Systems*, to appear.

[9] http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[10] P. Dey, S. Chandra, S. Gangopadhaya, "Speed Distribution Curves under Mixed Traffic Conditions", *Journal of Transportation Engineering*, Vol. 132, N. 6, June 2006, 475-481.