An application-level framework for information dissemination and collection in vehicular networks

Diego Borsetti\textsuperscript{a}, Marco Fiore\textsuperscript{b}, Claudio Casetti\textsuperscript{a,}\textsuperscript{*}, Carla-Fabiana Chiasserini\textsuperscript{a}

\textsuperscript{a} Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy
\textsuperscript{b} Université de Lyon, INSA Lyon, INRIA, CITI, F-69621, France

\textbf{Abstract}

Mobility of wireless network nodes is increasingly regarded as a fundamental resource for future pervasive communication systems. In this paper, we leverage the movement of communication-enabled vehicles to implement an original Application-level Role Mobility (ARM) framework. ARM allows nodes to share a generic assignment, by handing each other the associated application-level role. The handover of the role is performed according to the mobility patterns of the vehicles, following rules that are specific to the objective of the application. We employ the ARM framework for two different tasks: dissemination of information to traveling cars, and data collection from roadside sensors. For each application, we provide dedicated role handover rules and show, via simulation, that ARM can successfully perform the required operations in a lightweight, fully distributed way.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The future of urban mobility is increasingly intertwined with the breathtaking developments in Information Technology, to the point that it is expected that always-on multi-interface network devices will be carried around by vehicles and pedestrians alike. Such a dramatic increase in mobile node density will provide enough additional resources to effectively complement a network infrastructure, or even replace it.

Such a scenario opens up countless opportunities to ease into the picture new ground-breaking applications that exploit the widespread availability of mobile nodes to disseminate or collect information in the urban environment. Examples of information dissemination abound: drivers may report hazardous or congested traffic conditions to nearby areas and let fellow drivers provide updates as they drive by; a parking lot may advertise information on empty parking spots to nearby passing cars; a restaurant may want to advertise its lunch menu to potential customers. Likewise, information collection without the need of an infrastructure can be boosted: sensor networks can be deployed with the purpose of continuous monitoring of traffic conditions or air quality sensing; their data can be collected by passing mobile nodes and shuttled back to a gateway, without energy-consuming multi-hop transmissions from sensor to sensor.

This paper addresses the scenario outlined above by proposing a distributed framework implementing an innovative concept: Application-level Role Mobility, ARM. In ARM, mobile nodes take up a temporary role as “bearers” of information, or “collectors” of information, depending on their position, direction and speed: in short, depending on their mobility pattern. In the following, we will call the node upon which either one of such roles is bestowed as the Carrier. A Carrier can thus collect and dispatch data to an assigned location or act as a provider of those data to nearby nodes. The Carrier...
role is temporary and it is relinquished to another node depending on the mobility pattern of the Carrier: when the current Carrier is no longer a good choice (i.e., its mobility forces the Carrier out of the area where the corresponding application-level task should be performed), a distributed election procedure selects the best choice among the Carrier’s neighbors and requires the current Carrier to hand the role (and whatever information is coming with it) over to the newly selected one. Note that ARM does not implement a specific application, rather it provides support to the application layer for the development of cooperative services, like those outlined. In the paper, we will identify and discuss two possible variants of ARM, i.e., ARM-d (ARM-dissemination) that supports information dissemination in the absence of an infrastructure and ARM-c (ARM-collection) that provides a data mule-like support to collect and deliver information from specific areas of the network to a gateway. Additionally, depending on the application, the Carrier has the possibility to enhance the received content by adding/updating information. Finally, beside being implemented through a very lightweight communication protocol, ARM requires no knowledge of the network’s environment.

Arguably, the goals outlined above could be achieved by using existing or envisioned pervasive infrastructure nodes, like cellular telephony base stations, WiFi access points, or multi-hop sensor networks. However, we note that

(i) operators may rightfully charge for the used bandwidth and the information may only be available to customers of a single operator, while ARM relies on an ad hoc communication paradigm;
(ii) a cellular telephone or WiFi infrastructure only provides cell-level granularity in the information delivery; our framework instead dispatches information over an area of arbitrary size;
(iii) large-scale multi-hop sensor-based data gathering has a high complexity in terms of node auto-configuration, which translates into high energy costs and low scalability; on the contrary, mobile nodes can efficiently collect data from sensors with minimal resource consumption;
(iv) a fixed infrastructure constrains a service to the areas where base stations, access points, or sensors are deployed, while our framework, by using mobile nodes as service providers, allows flexibility and easy reconfigurability.

We point out that our solution is especially appealing since it aims at an easily implementable, lightweight, open framework that hinges upon existing technology (802.11-based MAC, a GPS localization system). In this paper, we focus on a vehicular context, for which the ARM framework appears particularly suited for a number of reasons: cars are the most likely candidates to have an always-on GPS/infotainment system on board; they can be expected to achieve a fairly high density; their movements are predictable for longer time periods than pedestrian.

The rest of the paper is organized as follows. We discuss previous work in Section 2 and present the concept of application-level role mobility in Section 3. Section 4 introduces the two indices used for data dissemination and data collection, respectively, while Section 5 describes the communication protocol that is needed in order to implement our framework. Performance evaluation of ARM-d and the ARM-c is provided in Section 6; conclusions are drawn in Section 7.

2. Related work

Our work relates to geocasting, to the selection of forwarders in delay-tolerant networks (DTNs), to information dissemination in VANETs and to the use of mobile elements, also referred to as “data mules” to collect data in wireless sensor networks.

As far as ARM-d is concerned, the works in [1–3] address the problem of extending the coverage of fixed gateway nodes through gateway or relay vehicles. However, none of them aims at replacing infrastructure. Related to our study are also the routing schemes for vehicular networks in [4,5], which propose hybrid geographic routing protocols to provide a reliable path towards the destination. In particular, in [5] next hops are selected based on a routing metric that depends on the vehicles’ speed, distance from the destination and movement direction. We point out that routing techniques allow the information to reach a desired location, but they do not confine it to a target area or along a trajectory, nor do they allow the specification of a target speed.

On the contrary, in [6], several approaches to maintain information within a geographic area are proposed. The Election is the one closest to the ARM-d framework, as it employs one of the mobile nodes as the information carrier and runs an election process among nodes in the targeted geographic area when the current carrier leaves such a region. However, the solution in [6] requires a geocast routing protocol to deliver the information to the elected node, with all the associated overhead, while ARM-d is self-contained. In the context of geocasting in vehicular networks, it is also worth mentioning the GeoNet project [7], which aims at implementing geographic addressing and routing protocols with support for IPv6.

Of particular relevance to the problem we address are the works in [8,9,6,10–13]. In various degrees, they address the topic of data dissemination in VANETs exploiting the notion that mobile nodes will occupy a different position in time and they may be used to carry data to a target area. They mainly differ from our approach either in terms of overhead or for the lack of support for information hovering in the target area. In [10], data generated by a gateway node are broadcast along busy roads using vehicles as data forwarders: the current forwarder hands its role over to the vehicle among its neighbors that is farther away in the data propagation direction. Whenever a fixed wireless device can be installed at the intersections, forwarders can pour the data on it; the device will then retransmit the information. Several aspects differentiate our work from [10]: (i) the goal of the two studies is different, since we do not rely on any fixed device, (ii) unlike [10], our broadcast application is not limited within the area nearby an intersection, and (iii) we do not resort to information on the road topology or road traffic characteristics.
The work in [11] presents a scheme to create a geo-localized virtual infrastructure for data broadcast in VANETs. However, there the information is assumed to be already within the target area and is handed over from vehicle to vehicle at every data broadcast, thus yielding a significant overhead. Also, unlike our solution, the scheme in [11] requires the target area to be centered at an intersection and to be characterized by high vehicle density; if the latter condition is not satisfied, the information dies out.

In [12], a scheme for information dissemination in large-scale vehicular networks is proposed. Although not directly related to our framework, this work has a similar goal to that of the delay-tolerant broadcast we employed in the ARM-d variant. Also, in [13] a navigation system is used to predict the vehicles’ future paths so as to route a message toward a specific location and keep it there. Although similar in spirit, ARM-d and the work in [13] differ, since the latter requires knowledge and advertisement of vehicles’ future routes, which may raise serious privacy issues and cause higher overhead.

As far as ARM-c is concerned, several works are related, having dealt with the use of mobile elements to collect data in wireless sensor networks. Gandham et al. in [14] propose a method to place a mobile base stations so as to evenly drain energy from sensors: time is divided into rounds during which base stations remain stationary, at the end of each round base stations are relocated with the objective of minimizing the maximum sensor energy expenditure. Other works addressing mobile base station relocation are [15,16].

Opportunistic sensor networks have been recently proposed to achieve sensing coverage and sensor data collection through uncontrolled mobile elements [17]. The concept of data mules has been first introduced in [18], where mobile nodes collect data from sensors, carry them and deliver the data to a fixed sink node. There, a two-dimensional random walk is assumed to model the mules’ mobility and an analytical model with data success rate and required buffer capabilities as performance metrics is derived. A study on predictable observer mobility is introduced in [19], where public transportation vehicles (e.g., buses, train) act as mobile observers in a wide area sensor network. The data collection process is modeled as a queuing system to evaluate the benefits of using predictable observers. In [20] Somasundara et al. suggest that controlled mobile elements can be intentionally introduced into the system to improve its lifetime, and they prove that the problem of scheduling the route of mobile elements to avoid sensor buffer overflow is NP-complete. In [21], a new approach is proposed to compute mobile trajectories, taking into account also the mule’s speed. Rendezvous-based solutions are presented in [22], where a subset of sensors act as rendezvous points at which data are buffered, and then they drop the data off to mobile base stations.

Even if the mule discovery process and transmission protocol go beyond the scope of this paper, it is worth noticing that in [23,24] Anastasi et al. propose an ARQ-based data-transfer protocol that is able to reduce the average transfer time while providing good performance in terms of energy saving. There, they also provide interesting results on mule discovery time and missed contact probabilities.

Finally, we would like to mention the numerous works addressing the problem of forwarder selection in DTNs. One of the early studies on this topic is presented in [25], while more recent works can be found in [26,27]. In particular, in the former the network topology is distributed and updated through a link-state routing protocol, and a message is forwarded to a neighbor only if this is closer to the destination than the node currently storing the message. The latter, instead, introduces the delegation forwarding approach, which creates information replicas at the network nodes, with the aim to increase the data delivery probability.

As a closing remark, we mention that early versions of our study can be found in [28,29].

3. The ARM framework

The goal of the ARM framework is to organize the communication among mobile nodes so that a selected set of them can effectively, and reliably, replace fixed infrastructure, providing data dissemination or collection in a specific area. ARM procedures are initiated by a source node, in our scenarios typically represented by a fixed road-side unit. Every mobile node participating in the ARM framework is supposed to carry a localization device, such as a GPS, thus enabling it to gauge its own suitability as an application-level role carrier, i.e., as a Carrier.

When the source node adhering to ARM bestows the Carrier role on to a mobile node, it can provide it with two different goals (and related sets of instructions), thus defining the following two ARM versions:

- **ARM-d**: the goal is to carry an information object into a circular target area and have it linger there for the purpose of dissemination. The information object is tagged with the coordinates of the center of the target area (target point) and with the radius \( R \) of the desired target area.
- **ARM-c**: the goal is to follow a desired trajectory for the purpose of data collection. The trajectory is represented by segments joining a sequence of targets, identified by geographical coordinates, called a route vector. Movements of the Carrier with respect to the ideal trajectory are allowed within a tolerance zone: the area within which the Carrier should be confined is named a road pipe, i.e., a pipe of width \( w_p \) centered around the ideal trajectory. Additionally, a desired nominal speed for the Carrier is set, yielding a predictable latency in reaching each target.

In either version of ARM, a handover process among nodes allows a newly elected node to adhere to the constraints set forth for the information dissemination or collection. If the vehicle finds itself unfit for the Carrier task, it starts an election process among its neighbors and hands the Carrier role over to a better candidate, if any. This procedure is iterated over time, as any mobile node acting as a Carrier must periodically check its fitness for the role and start a new election if needed. The
decision on whether and, if so, to which neighbor to hand over the Carrier role clearly is a key aspect in the implementation of ARM. As our goal is to have the Carrier linger within a given area or reach a target point, the fitness of a mobile node as a Carrier depends on how well its current and future routes meet the aforementioned objectives. Since exact knowledge of future movements of each vehicle is not possible, the election process is performed exploiting the information available to any mobile node equipped with a position system, such as location, speed, acceleration, and heading. Such values are used to compute an index that is representative of the suitability of a vehicle as a Carrier: the higher the index value, the better the fitness for the role.

Therefore, a vehicle currently acting as a Carrier periodically computes its own index, and, if its value is below a minimum threshold, it starts an election process, by advertising the set of ARM instructions. Neighboring vehicles compute their own indices and send them back to the current Carrier, which hands over its role to the mobile node with the highest index. Ideally, the index formulation should be such that the Carrier meets as closely as possible the ARM objectives, while keeping the number of handovers (hence the control overhead) low.

In the next section, we further detail the dissemination and collection scenarios, and discuss the indices used in the two cases.

4. Index formulation

We introduce the ARM-d index and the ARM-c index, briefly outlining the context in which they are used. For each of them, we describe the formulation and show the suitability to handle all situations of practical interest.

4.1. ARM-d index

The operations performed by the ARM-d framework aim at data dispatching and dissemination in areas where fixed infrastructure is unavailable. By complementing ARM-d with an application-layer beaconing function that advertises and disseminates the information object carried by the Carrier in the target area, a “virtual hot spot” can be mimicked. To this end, an information source entrusts the data, and the Carrier role, to a passing mobile node that, according to its mobility profile is currently the fittest to reach a target location. Whether the Carrier retains its role is, indeed, chiefly determined by the consistency of its mobility profile with the target location. Together with role handover, carried data are transferred from the old on to the new Carrier. Broadly speaking, the selection of the Carrier is therefore aimed at: (i) picking the node that, at any time, appears more likely to reach a target area, i.e., a circular region of radius $R$ whose center is identified by a set of coordinates; (ii) upon reaching the target area, picking the node that is more likely to linger within the target area, and thus confine the information within it. Once there, the beaconing function can advertise the information object to nearby nodes.

To achieve the above objectives, we design the ARM-d index according to the following general rules: (i) the index is positive when a node is well within the target area or it is approaching it; (ii) the index is negative when a node is at the area border or outside the target area. Thus, a negative ARM-d index is associated with the need to look for a better Carrier.

We first introduce the location parameters that the ARM-d index accounts, then we outline several requirements and purposes that the index should serve, identifying appropriate mathematical functions of the localization parameters that realize those requirements.

4.1.1. Location parameters

Given a tagged node, its ARM-d index is computed considering the following input, a sample depiction of which is in Fig. 1:

- the distance $d$ of the node from the target location;
- the heading $\theta$ of the node, computed as the angular distance of its speed vector with respect to the straight line connecting the node to the target. The heading is 0 when the node moves exactly towards the target, while it reaches its maximum value, equal to $\pi$, when the node travels exactly away from the target;
- the value of the speed $v$ of the node;
- the range $R$ of the target area, within which the information object should linger.

Note that the node's position is obtained by means of a localization technology; the target position and the value of $R$, instead, need to be communicated to the node, a task accomplished as described in Section 5.
4.1.2. Distance contribution

The distance \( d \) between a node and the target is the first parameter to consider in building the ARM-d index. The relationship between \( d \) and the node’s suitability as a Carrier is fashioned after the following rules:

f.1 a node in the target area \( (d \leq R) \) is always considered a better Carrier than a node out of the target area \( (d > R) \);

f.2 the quality, as a Carrier, of a node outside the target area decreases as its distance \( d \) from the target increases;

f.3 to minimize the number of role handovers, a node at distance \( d \), which is neither too close to 0 nor too close to \( R \), should be favored. Indeed, a node moving toward, and already nearby, the target location would spend a shorter time within the target area than another moving along the same direction but that is farther away from the target. However, a vehicle that is too close to the border of the target area (i.e., at distance \( R \) from the target) is not convenient either: even if the vehicle is currently speeding toward the target, it may move out of the target area very quickly if it changes direction. In addition, considering a Carrier that is speeding away from the target location, selecting as next Carrier a node that is close to the border implies that the current Carrier should start the handover procedure when close to the border itself. This would cause the object to leave the target area whenever the current Carrier cannot find any nearby node moving toward the target.

Accordingly, by using simple linear\(^1\) functions, we define \( f(d) \), which quantifies the distance contribution to the index, as

\[
f(d) = \begin{cases} \frac{d}{R}, & \text{if } 0 \leq d \leq \frac{R}{2} \\ 2 \left(1 - \frac{d}{R}\right), & \text{if } d > \frac{R}{2}. \end{cases}
\]  

The resulting function is depicted as the thicker curve in Fig. 2. Notice that \( f(d) \) returns values in \((-\infty, 0)\) for nodes outside the target area, while \( f(d) \in [0, 1] \) inside the target area, which realizes the rule f.1. Moreover, the negative derivative of the second linear function in (1) guarantees that rule f.2 is satisfied, and, jointly with the positive derivative of the first linear function in (1), determines a maximum in \( R/2 \) which fulfills rule f.3. The coefficients in (1) are chosen so that \( f(d) \) is continuous for all values of \( d \).

4.1.3. Heading contribution

Nodes traveling towards or away from the target area must carry different ARM-d indices, even when at similar distance from the target. We base the contribution of heading \( \theta \) on the following criteria:

\( g.1 \) nodes traveling towards the target (i.e., \( \theta \) tends to 0) are preferable over those moving away from the target (i.e., \( \theta \) tends to \( \pi \));

\( g.2 \) the impact of the heading becomes of less importance when nodes are close to the target itself.

Such considerations lead to the formulation of a functional \( g(\theta, d) \), which, to comply with the second criterion above, is necessarily a function not only of \( \theta \), but of \( d \) as well, and can be expressed as

\[
g(\theta, d) = \alpha(d) \nu(\theta) 
\]

\[
\alpha(d) = \begin{cases} \frac{d}{R}, & \text{if } 0 \leq d \leq \frac{R}{2} \\ 1, & \text{if } d > \frac{R}{2} \end{cases}, \quad \nu(\theta) = -\frac{\theta}{\pi}. 
\]

\(^{1}\) Note that other functions (e.g., parabolic and trigonometric functions) provided similar results (omitted for brevity); thus, we opted for the simplest, i.e., for linear, functions.
Function $\nu(\theta)$ is linear with $\theta$ with negative derivative, thus fulfilling rule g.1. The effect of distance imposed by criterion g.2 is modeled by $\alpha(d)$, that multiplies $\nu(\theta)$ in $g(\theta, d)$ in Eq. (2). From (3), for $d > \frac{R}{2}$, $\alpha(d)$ has value 1, and thus no impact on $g(\theta, d)$. However, for $d \leq \frac{R}{2}$, it is directly proportional to $d$, tending to 0 for decreasing distances: thus, the closer the node is to the target, the more $\alpha(d)$ mitigates the differences that diverse values of $\theta$ induce on $g(\theta, d)$. Again, the coefficients in (3) ensure that the index is continuous for all values of $d$ and $\theta$.

Fig. 2 shows the sum of $f(d)$ and $g(\theta, d)$ for different values of $\theta$. We highlight that, in the inner core of the target area ($d \leq R/2$), the contributions of $f(d)$ and $g(\theta, d)$ cancel each other out when $\theta = \pi$, i.e., the Carrier is moving away from the target. In this case, the node’s fitness as a Carrier is maintained at a low, constant value; this is desirable since the node is traveling away from the target and its being closer or farther from the target hardly matters. Conversely, when a node is approaching the target ($\theta = 0$), the sum of the functionals matches the shape of $f(d)$, and thus follows the principles considered in Section 4.1.2.

4.1.4. Speed contribution

Finally, we factor in the node speed. Indeed, the ARM-d index should tell apart nodes that are at similar distance from the target and travel in the same direction, but with different speeds. The rationale behind the contribution of speed $v$ is based on the following considerations:

h.1 if a node is approaching the target, its ARM-d index should be

- increasing with its speed, if the node is outside the target area, as a fast moving node is expected to reach the area before a slow one;
- independent of speed, if the node lies close to the target range $R$. Indeed, in this case, both high and low speeds can lead to potential advantages: on the one hand, a fast node would reach the target first, on the other, a slow one would spend more time within the target area;
- decreasing as the speed grows, if the node is within the target area, since at lower speeds the node will linger within a distance $R$ from the target position for a longer time;

h.2 if a node is moving away from the target, its ARM-d index should be higher at low speeds than it is at high ones, independently of the distance $d$ from target; indeed, lower speeds allow the information to linger for a longer time close to the target.

Bearing these remarks in mind, we define the speed functional as a function of the node speed $v$, the heading $\theta$ and the distance $d$, its analytical representation being

$$h(v, \theta, d) = A(\theta, d) \frac{v}{v_{\text{max}}} + B(\theta, d)$$

$$A(\theta, d) = \frac{\delta(d)}{R} [v(\theta) + 1] - 1; \quad B(\theta, d) = -\frac{\delta(d)}{2R} [v(\theta) + 1] + 1$$

$$\delta(d) = \min\{d, 2R\}.$$  (6)

From (4), the functional $h(v, \theta, d)$ is linear with $v$, with coefficients $A$ and $B$ which vary with $\theta$ and $d$. The expressions of such coefficients in (5) shape the line in a way to satisfy the above requirements. Also, note that the computation of the coefficients requires knowledge of the maximum speed $v_{\text{max}}$ that a node can reach, which is fair to assume nodes know.

Let us first consider the case of a node moving towards the target ($\theta = 0$), highlighted by the dashed line in Fig. 3: we note that, when the distance from the target is small, as in Fig. 3(a), $A(\theta, d)$ is negative and $h(v, \theta, d)$ decreases linearly as the speed $v$ increases. However, as $d$ grows, the coefficient in the expression of $A(\theta, d)$ tends to zero making $h(v, \theta, d)$ less and less sensitive to the speed, up to independence from $v$ for $d = R$ (as from the constant behavior of the dashed line in Fig. 3(c)). For distances larger than $R$, the relationship between $h(v, \theta, d)$ and $v$ becomes directly proportional, up to a value $d = 2R$: for even greater distances, the value of $h(v, \theta, d)$ does not vary, thanks to the condition imposed by (6). Summarizing, the impact of speed on the index of a node approaching the target is exactly that described in criterion h.1.

The case of a node that leaves the target ($\theta = \pi$) is instead shown in Fig. 3 by the solid line. This time, the inverse proportionality between $h(v, \theta, d)$ and $v$ is unaffected by the distance, as required by h.2. Indeed, as the heading tends to $\pi$, the presence of a factor $v(\theta)$ in (5) makes the coefficients $A$ and $B$ more and more independent of distance $d$, as portrayed by the surfaces in Fig. 3.
4.1.5. Rounding up the contributions

The ARM-d index is obtained as the sum of the three functionals \( f(d), g(\theta, d), \) and \( h(v, \theta, d) \), normalized so as to restrict the possible values of the index to the interval \((-\infty, 1]\). The complete formulation of the ARM-d index is thus

\[
i(d, \theta, v) = K \left[ f(d) + g(\theta, d) + h(v, \theta, d) \right]
\]

where \( K \) is the normalization constant.\(^2\) The ARM-d index is plotted in Fig. 4, as a function of the distance from the target, for the four boundary combinations of \( \theta \) and \( v \) values.

It can be noticed that, thanks to the ranges of values returned by the different functionals, the ARM-d index satisfies all the sets of rules which drove the design of \( f(d), g(\theta, d), \) and \( h(v, \theta, d) \). In particular, we underscore the following properties of the ARM-d index that match the goals we set at the start of the section:

- **outer far area** \((d > 2R)\): the heading \( \theta \) yields the predominant contribution, as a node approaching the target exhibits a higher index than one leaving it. Such a difference is further influenced by speed \( v \), since nodes that quickly carry the information towards the target area \((\theta \to 0, v \to v_{\text{max}})\) are preferable, while nodes leaving the target area at high speed \((\theta \to \pi, v \to v_{\text{max}})\) are not good choices;
- **outer near area** \((R < d \leq 2R)\): ARM-d indices of nodes approaching the target area \((\theta \to 0)\) at different speeds tend to become similar, since, as discussed in Section 4.1.4, different speeds hold complementary advantages. In any case, such indices are higher than those of nodes leaving the target area \((\theta \to \pi)\). Nodes leaving the target area slowly \((v \to 0)\) are obviously better Carriers than those leaving it rapidly \((v \to v_{\text{max}})\);
- **inside the target area** \((d \leq R)\): for a given distance \( d \), the most suitable Carriers are always nodes moving towards the target at slow speed \((\theta \to 0, v \to 0)\), as they are expected to spend more time within the target area. More precisely, in the outer rim \((R/2 < d \leq R)\), it is desirable that the information is brought nearer to the target, while in the inner core \((d \leq R/2)\), the goal becomes to move the information object as slowly as possible. Also, as shown in Fig. 4, the index becomes negative as nodes approach the border of the target area.

4.2. ARM-c index

The ARM-c version is ostensibly aimed at replacing the fixed infrastructure for the purpose of collecting data from sensors that are placed along the roads in an urban environment. Such sensors can perform a number of monitoring functions, e.g., traffic conditions, parking slot availability, or levels of air pollution. Sensed data should then be conveyed from the sensors to nodes that can process them and make them available. Relying only on sensor-to-sensor communication would raise issues in terms of connectivity, reliability, timeliness, and energy efficiency. A more efficient solution, instead, consists of exploiting the high number of mobile nodes that populate urban areas, to perform a mule-based data collection that solves the aforementioned problems. By following a desired trajectory, mules collect the measured data and deliver them to a central controller; furthermore, controlling the speed of mules allows the system to determine the periodicity of visits to sensors as well as the mule-sensor contact time. Under traditional data muling, this would require to find, for each trajectory, a mobile node that continuously travels along the desired route, possibly with a certain constant speed. Obviously, such a mobility profile does not match that of any pedestrian or vehicular user, making the solution unfeasible. The same solution becomes, instead, feasible if *virtual* data muling is employed, i.e., by associating the data mule with the Carrier. A careful selection of mobile nodes allows then the Carrier to follow the desired trajectory thus solving the data collection problem.

With the above requirements, the ARM-c handover decision must be based on both the desired trajectory and the desired cruise speed of the Carrier. To build the ARM-c index, a possible approach is to resort to fuzzy logic, which achieves the important objective of controlling the time when the Carrier visits the single sensors. That is, the fuzzy-based technique enables sensors to follow a pre-determined sleep/wake-up schedule, hence to save energy.

\(^2\) \( K \) is computed so that the maximum value taken by the ARM-d index is equal to one.
Our fuzzy-based index takes as inputs the desired trajectory as well as the cruise speed that the Carrier should follow. By coupling these two pieces of information, it is possible to determine the exact position $P_e(t)$ where the Carrier is to be found at each time instant. Then, beside the road pipe of width $w_p$, at each time $t$, we define the tolerance distance from $P_e(t)$ along the ideal trajectory. For symmetry, we set the maximum value of the tolerance distance to $w_p$.

Knowing the parameters above, the fitness of a vehicle for the Carrier role is determined by three factors, namely (i) the distance between the geographical position of the node and the exact desired Carrier location $P_e(t)$, (ii) the direction of movement of the node, and (iii) the instantaneous node speed and acceleration. The contribution of each factor is discussed in detail next.

4.2.1. The distance factor

The distance between the location of a node and $P_e(t)$ is employed to compute a distance factor (DF), ranging between $-2$ and $0$. Less negative values of DF correspond to a better positioning of a vehicle with respect to $P_e(t)$. More precisely, the distance between the node and $P_e(t)$ can be decomposed into a lateral error $e_w$ and a longitudinal error $e_l$.

The lateral error is defined as the deviation of the node with respect to the border of road pipe, measured perpendicularly to it. Note that if at time $t$ the vehicle is inside the road pipe, $e_w = 0$; otherwise $e_w > 0$. We also denote by $E_w$ the maximum admissible lateral error: we assume that when $e_w > E_w$, the current Carrier either sends the information to a roadside unit, if any exists within its transmission range, or drops it.

The longitudinal error $e_l$, instead, denotes the distance between the orthogonal projection of the vehicle on the ideal trajectory and the current desired Carrier position, $P_e(t)$. We denote by $E_l$ the maximum acceptable longitudinal error; similarly to the maximum lateral error, the current vehicle will either drop or return the information to a roadside unit when $e_l > E_l$.

The lateral and longitudinal errors are the inputs to a fuzzy logic controller, which yields two numerical output values, namely, TowardPipe and TowardPoint, whose combination determines the distance factor (DF) contribution to the final ARM-c index. For the sake of clarity, a graphical representation of this scheme is provided in Fig. 5.

In order to obtain the first output value, TowardPipe($e_w$), the fuzzy logic controller maps the lateral error onto three input variables: FarFromPipe indicates that the vehicle is well outside the road pipe (i.e., $e_w > E_w/2$), NearToPipe denotes that the vehicle is just outside the road pipe (i.e., $0 < e_w < E_w/2$), while InPipe implies that the vehicle is inside the road pipe (i.e., $e_w = 0$). Such mapping is performed by means of membership functions, denoted by $\mu(\cdot)$, that state how “true” a variable is (i.e., with which probability a variable is true), given the values of the function argument. The functions that are used to map the lateral error on to the three input variables above are called input membership functions, and are depicted in Fig. 6.

The fuzzy logic controller then relates such input variables into three output variables, Ineligible, Inappropriate, and Appropriate, whose intuitive meaning refers to the suitability of a vehicle as a Carrier. The following rules are applied:

- **if FarFromPipe is true then Ineligible is true**
- **if NearToPipe is true then Inappropriate is true**
- **if InPipe is true then Appropriate is true**.

The output membership functions, which map the output variables on to TowardPipe($e_w$), are depicted in Fig. 6. The contributions of the three output variables to the value of TowardPipe($e_w$) are then combined by applying the “center
TowardPipe(e_w) is a well-known technique in fuzzy logic-based systems [30]. The overall relationship between e_w and TowardPipe(e_w) is graphically represented by the surface in Fig. 7(a). There, we can observe that a vehicle traveling outside the road pipe always corresponds to a negative value of TowardPipe(e_w), while such value becomes zero for vehicles inside the road pipe.

The second numerical value contributing to the distance factor DF is TowardPoint(e_w, e_l). In this case, the fuzzy logic controller maps the lateral and longitudinal errors onto three input variables: FarFromPoint tends to be true if the vehicle is inside the road pipe but widely out of the tolerance area centered at the desired Carrier position, P_e(t) (i.e., e_w = 0 and e_l > E_l/2); NearToPoint, indicates that the vehicle is within the road pipe but outside the tolerance distance from P_e(t) (i.e., e_w = 0 and w_p < e_l ≤ E_l/2); finally, InPoint denotes that the vehicle is inside the road pipe and within w_p from P_e(t) along the ideal trajectory. The input membership functions that are used to perform such a mapping are shown in Fig. 6. Note that, in the plot, these membership functions are defined for e_w = 0; for e_w > 0, FarFromPoint is true, while NearToPoint and InPoint are false, with probability 1.

Such input variables are tied to the output variables Insufficient, Adequate, and Suitable, which again relate to the fitness of the vehicle as a Carrier. In this case, the following rules hold:

- **if FarFromPoint is true then** Insufficient **is true**
- **if NearToPoint is true then** Adequate **is true**
- **if InPoint is true then** Suitable **is true.**

The output membership functions, mapping the output variables on to TowardPoint(e_w, e_l), are represented in Fig. 6.

By combining the contributions of the three variables as before, we obtain the bidimensional function TowardPoint(e_w, e_l) shown in Fig. 7(b). Note that TowardPoint(e_w, e_l) is always equal to −1 if e_w > 0 and takes its maximum value (i.e., zero) for e_w = 0 and e_l ≤ w_p.

As already stated, the distance factor DF is computed by summing up the values of TowardPipe and TowardPoint obtained for the current errors e_w and e_l; the result is shown in Fig. 7(c). There, we can observe that the composition of the TowardPipe and TowardPoint functions leads to a DF equal to zero when the vehicle is exactly at the desired location (i.e., e_w = e_l = 0). When e_w grows, the distance factor decreases very quickly at first, and more slowly afterwards. Indeed, 3
a lateral displacement often implies that the vehicle steered away from the desired route, and thus weights negatively in DF. This is especially true for small values of $e_w$, meaning that, when the vehicle is quite close to the road pipe, a recovery is possible if the role is handed over quickly. Longitudinal errors have a less dramatic impact on the distance factor: only large values of $e_l$ significantly reduce DF. Moreover, the contribution of $e_l$ becomes negligible in the presence of overwhelming lateral errors. This reflects the fact that it is much easier to recover from longitudinal than from lateral errors.

4.2.2. The steer factor

The direction of movement of the vehicle is then used to determine a steer factor (SF), which takes values in $[-1, 1]$. More positive values of SF are representative of car headings closer to that of the ideal trajectory. As detailed next, the design of our steer factor follows a similar rationale to that behind real-world car steer control systems, targeted at keeping a vehicle along a desired trajectory.

The input parameters now are the lateral error $e_w$ and the angular error. In this case, the lateral error is considered to be positive when the vehicle travels on the left side of the trajectory and negative if it is on the right side (the side of the trajectory can be identified by taking as reference a vehicle that travels in the proper direction along the trajectory). The input variables used by the fuzzy logic controller are: LateralRight and LateralLeft, denoting that the vehicle is traveling on the right side and on the left side of the trajectory, respectively; AngularRight and AngularLeft, identifying, respectively, a vehicle traveling with a negative and positive angle with respect to the reference trajectory. The relative input membership functions are shown in the left plot of Fig. 8.

The output variables describe instead the action that the automatic control should perform on the heading of the Carrier, TurnRight and TurnLeft. The corresponding rules to be applied are:

- If AngularLeft is true or LateralLeft is true then TurnRight is true
- If AngularRight is true or LateralRight is true then TurnLeft is true,

while the shape of the membership functions used to map the output variables onto the steer factor SF is shown in the right plot of Fig. 8.

We point out that the rules above, as well as the shape and thresholds of input and output membership functions, are derived from [31]. Fig. 9 shows the surface that describes the mapping between the lateral and angular errors and SF.

As depicted in the figure, SF takes a zero value in absence of lateral and angular errors. Similarly, the steer factor tends to be null if the vehicle (i) is currently displaced on the right of the road pipe ($e_w > 0$) but it is also steering towards the left (angular error $< 0$), or (ii) is currently displaced on the left of the road pipe ($e_w < 0$) but it is also steering towards the right
Fig. 10. Input (left) and output (right) membership functions for the throttle factor (TF).

Fig. 11. Throttle factor function.

(angular error > 0). In other words, SF tends to zero when the direction of a vehicle that is out of the road pipe lies on the correct trajectory. Conversely, SF tends to 1 or −1 when the vehicle moves out of the road pipe, and its direction brings it further away from the desired route: in such a case, a role handover is necessary.

4.2.3. The throttle factor

The vehicle’s speed and acceleration are employed to compute a throttle factor (TF). The values of TF range between −1 and 1, where positive values map on to speed/acceleration pairs that are closer to those needed to follow the movement of $P_e(t)$ over time. Again, the TF design takes inspiration from control systems devised to manage the throttle of cars [30,32].

By denoting the vehicle’s speed by $v$ and the desired Carrier cruise speed by $v_d$, we define the input parameters as the speed error $v_e = v - v_d$, and the acceleration of the vehicle. Within the fuzzy logic controller, the first is mapped onto the input variables Slow and Fast (indicating that the vehicle is either too slow or too fast with respect to $v_d$), while the second is mapped on to the input variables SpeedUp and SlowDown (denoting an increasing or decreasing future speed). The relative membership functions are depicted in the left plots of Fig. 10.

The output variables used to determine TF are AccUp and AccDown, which refer to the acceleration needed to catch up with the current desired Carrier position. They are controlled through the following rules:

- if Fast is true or SpeedUp is true Then AccUp is true
- if Slow is true or SlowDown is true Then AccDown is true.

The output membership functions that return TF are shown in the right plot of Fig. 10.

The fuzzy rules, the shape and the thresholds of input and output membership functions are taken in this case from [30,32]. The resulting bidimensional function from the speed error/acceleration space to the TF value is represented in Fig. 11. Similarly to the SF, the throttle factor is also zero when the vehicle moves at the desired cruise speed, while it takes values close to −1 and to 1 when it is too slow and too fast, respectively. Also, given a slow (fast) vehicle, the higher its acceleration (deceleration), the closer to zero its TF.

4.2.4. Rounding up the contributions

The overall index controlling the ARM-c handover is obtained by composing the three factors, as:

$$i = DF + 1_{[DF=0]}(2 - (|SF| + |TF|))$$

where $1_{[DF=0]}$ takes the unitary value when $DF = 0$, and it is zero otherwise.
This aggregation of the DF, SF, and TF factors, although simple in its formulation, satisfies the control rules behind each contribution, as also shown by the simulation results presented next. Note that the ARM-c index ranges in the interval \([-1, 2]\), and an election procedure to select a new suitable Carrier has to be triggered whenever the index of the current Carrier drops below 1. Indeed, when negative values of DF (i.e., \(i = DF\)) indicate that the vehicle is outside the road pipe or at a distance from \(P_e(t)\) along the ideal trajectory that is greater than \(w_p\); hence, a current Carrier with index \(i < 0\) should start a procedure for selecting a new Carrier. If instead \(DF = 0\) (i.e., the vehicle is within the tolerance area) but the vehicle direction or speed are unsuitable, \(|SF|\) and \(|TF|\) take values close to 1 and \(i \leq 1\); again, an election procedure has to be triggered. Finally, when \(DF = 0\) and both \(|SF|\) and \(|TF|\) take small values, it means that the Carrier is suitable and we have that \(i\) ranges in \([1, 2]\).

5. Communication protocol

ARM is a reactive, application-layer protocol, which assumes that a contention-based MAC (e.g., belonging to the 802.11 family) is used.

Let us consider that a source generates an ARM instance and starts a search procedure aimed at identifying a candidate node to either disseminate or collect an information object. Note that a similar procedure has been used in [33], with the aim of identifying a backbone of nodes in a vehicular ad hoc network.

The source computes its own index and sends a POLL message as a one-hop broadcast message, which, as such, will be heard by all one-hop neighbors. The POLL carries the following information: (i) the current Carrier index \(i_e\), (ii) a sequence number, (iii) the sender address, (iv) the identifier of the ARM instance and (v) the instructions related to the ARM instance. The latter are, in the case of ARM-d, the target geographical coordinates and the target area radius \(R\), while, in the case of ARM-c, the route vector, the current target point, the road pipe width \(w_p\) and the desired speed. A neighboring node receiving the POLL computes its own neighbor index \(i_n\) and compares it with the one broadcast in the POLL message. If the difference between the two indices \(\Delta = i_n - i_e > 0\), the neighboring node replies with a BID message to the Carrier. The BID message is sent as a unicast message so that, in case of failure, retransmissions can be performed at the MAC layer and higher reliability is provided. The BID message includes (i) the identifier of the ARM instance, (ii) the POLL sequence number and (iii) the BID sender address.

Upon overhearing the transmission of the BID, candidate nodes operating in promiscuous mode will refrain from issuing their scheduled reply, thus curbing the overhead. A flood of BID messages is also avoided by letting neighboring nodes reply after a lag time\(^4\) \(t_l\), which decreases with \(\Delta\). The lag time is computed as

\[
t_l = \begin{cases} 
T_l \left(1 - \frac{\Delta}{\Delta_{\text{max}}} \right), & \text{if } \Delta \leq \Delta_{\text{max}} \\
0, & \text{if } \Delta > \Delta_{\text{max}}
\end{cases}
\]

where \(T_l\) is the maximum lag time, and \(\Delta_{\text{max}}\) represents the maximum difference between Carrier and neighbor indices. For \(\Delta > \Delta_{\text{max}}\), the lag time is always set to 0. We also introduce a small jitter to avoid synchronization among BID transmissions.

As a consequence, due to the contention-based MAC, the first BID message received by the Carrier will likely be sent by the best available candidate in the neighborhood.\(^5\)

If no BID is received, the procedure is repeated at the next position update until a BID arrives. Upon receiving the first BID, the source replies with one or more unicast messages transferring the application-level role and the ARM data to the new Carrier. When the new Carrier acknowledges the reception, the search procedure ends.

The current Carrier keeps track of its position at the update frequency of its positioning system. Upon every position update, it recomputes its own index \(i_e\). If it is below the threshold (i.e., 0 for ARM-d and 1 for ARM-c), it searches for a better candidate to carry the information, following the procedure outlined above.

Finally, we point out that, during the role transfer, the Carrier drops the ARM data only if it receives an acknowledgment, thus preventing information loss and protocol disruption.

For the scenario depicted in Fig. 12, we provide an example message exchange generated by the ARM communication protocol in Fig. 13. There, nodes \(A, B\) and \(C\) are in radio proximity of each other, as are \(A, D\) and \(E\). Node \(A\) is a Carrier that is...

---

\(^4\) A similar mechanism is used, for instance, in DSR to prevent route reply storms, or in TCP delayed acknowledgment to exploit cumulative acks.

\(^5\) Due to message collisions or high buffer load, the first received BID may be from a second-choice candidate; however, we elected not to add any further robustness to this procedure as a tradeoff between latency and efficiency. Furthermore, setting \(T_l\) to a reasonable value (e.g., 200 ms) ensures that, if an inversion occurs in the reception of BID messages, with very high probability it takes place between two messages carrying similar index values.
moving away from the target point and, thus, starts a new election process among the nodes within its transmission range. Node A broadcasts a POLL message to its neighbors B, C, D, and E. Each of these nodes checks whether its local index is higher than that advertised by A in the POLL message, \( i_A \); in the example, nodes B, C, and D have a higher local index, and therefore compute a bidding delay \( t^B_i \), \( t^C_i \), and \( t^D_i \), respectively. Node E computes a lower local index and leaves the election process.

As for the remaining competing nodes, we assume \( i_B > i_C > i_D \), implying that \( t^B_i < t^C_i < t^D_i \). Node B thus waits for the shortest time and then broadcasts its BID message. The message is overheard by node C, which, as a consequence, stops its bidding timer before expiration and leaves the election process as well. The BID issued by Node B is also received by A, which replies with a unicast transmission containing the application-level role and the ARM data; then, node B concludes the data transfer by acknowledging the information reception to A.

The election process is now concluded, and B is the new Carrier, but node D is still waiting for its bidding delay to expire, since it could not hear the BID by B (an out-of-range transmission) or the reply by A (a unicast message).\(^6\) Therefore, after a time \( t^D_i \), node D broadcasts its own BID, that is however discarded by A as an out-of-date message.

### 6. Performance evaluation

Here, we study the performance of our framework in the two role mobility scenarios outlined before. We first present the vehicular simulation environment employed in our tests, and then show how the index-based framework successfully manages to hand over application-level roles between nodes, so as to achieve the different goals.

#### 6.1. Scenarios

To evaluate the performance of our framework, we employed the ns-2 network simulator, with the support of the MatLab Fuzzy Logic Toolbox to implement the ARM-c index. Vehicular mobility is generated using VanetMobiSim\(^{[34]}\), with car-to-car and car-to-road signaling interactions mimicked through the IDM-LC car-following model.

For both the data dissemination and collection applications, we focused on highway and urban scenarios, in the presence of different vehicular densities. More precisely, the highway scenario features a 2 km straight road, with two lanes in each direction, over which vehicles travel with desired maximum speed ranging between 70 and 130 km/h. The urban scenario is instead a 4 km\(^2\) regular city section, with intersections 500 m apart from each other and regulated by traffic lights. Roads include one lane in each direction. Each vehicle starts its itinerary from an entry/exit point at the border of the topology, and chooses its destination among a subset of border points that it reaches trying to maintain a desired speed between 35 and 50 km/h (although, due to the presence of traffic lights and slower vehicles the actual average speed is significantly lower than the target one). The highway and urban scenarios are portrayed in Figs. 14 and 15, respectively.

\(^6\) Note that no promiscuous listening is used.
We have tested four different average lane densities, namely 5, 10, 20 and 30 vehicles/km, representing very sparse to dense traffic conditions, and yielding different levels of network connectivity. In the highway scenario, a lane density of 5 vehicles/km implies, on average, a node degree (i.e., number of neighbors) equal to 2.74 and a network topology with 6.7 clusters, each of which is disconnected from the others and has mean size equal to 4.3 nodes. In the case of 20 vehicles/km, the average node degree is 11.5, the mean number of disconnected clusters is 1.5, and each cluster has an average size of 80 nodes. In the urban environment, on average, 5 vehicles/km implies a node degree of 3.3 and 34 clusters, each including on average 3 nodes, while for a density of 20 vehicles/km we have a node degree of 13.7 and 20 clusters in the network, each of mean size equal to 21 vehicles.

All vehicles are assumed to communicate using the IEEE 802.11 technology, with a fixed data rate of 11 Mb/s. Wireless signal propagation is simulated according to the two-ray ground reflection model, resulting in a radio range of 100 m; note that, in the urban scenario, this radio range prevents unrealistic communication between vehicles traveling on adjacent parallel roads. We also consider the update frequency of the vehicles positioning system to be equal to 1 Hz. All simulations are run for 10,000 s, and results are obtained so that 95% confidence intervals lay within 5% of the resulting mean value.

6.2. Data dissemination

When the framework is employed for the dissemination of an information object, i.e., in its ARM-d version, the target is located at the center of the simulation area, whereas a fixed information source is deployed at one end of the highway and at one of the entry points in the urban street layout. The former configuration is shown in Fig. 14(a), while the latter is depicted in Fig. 15(a).

Unless otherwise specified, the target area radius is set to \( R = 200 \) m, while \( T_i \) is set to 200 ms and \( \Delta_{\text{max}} \) is set to 1. The piece of information to be disseminated has a size of 1 kB, so as to fit one IP packet.\(^7\) We also set the size of the POLL and BID message, respectively, to 32 bytes and 16 bytes, consistent with the fields they contain.

The first collection of results shows the performance while steering the information to the target. Note that in all of our simulations the information object always successfully reaches the target area; what changes is the time needed to enter this area.

Table 1 presents the average and the standard deviation, observed over 20 simulation runs, of the time elapsed between the issuing of the information object at the source and the crossing of the border into the target area. Three different values of lane density are considered: 5 vehicles/km (low), 20 vehicles/km (medium) and 30 vehicles/km (high). We observe that the time to reach the target area is shorter in the highway scenario than in the urban environment, due to the higher vehicle density.
speed and the road topology that constrains movements along the trajectory where target points lie. Also, under low density conditions, the time to reach the target area may be significant; indeed, the same vehicle may have to carry the information all the way, for want of a better candidate. As the lane density increases from 5 to 20 vehicles/km, such an occurrence becomes unlikely thanks to multiple handovers, and the target area is reached in a much shorter time, for both the highway and urban cases.

When considering the dissemination role, the ARM-d framework objective is to confine the information in the target area with minimal overhead, thus we would like to maximize the time a Carrier holds the information (hereinafter called caching time) so as to reduce the number of role handovers. We compare the ARM-d implementation discussed in Section 4.1 against two benchmarks: a handover strategy based on a simpler index, and an oracle-based heuristic.

Distance-based: The rationale of comparing ARM-d with a simpler strategy is to gauge the payoff of a definition of the ARM-d index based on three separate contributions, versus a simplistic index that only carries distance information. Thus, we define a framework identical to ARM-d, but for the choice of the ARM-d index: \( i'(d) = f(d) \), with \( f(d) \) defined as in (1).

Heuristic: We would like to compare the choice of the sequence of Carriers performed by ARM-d to an optimal choice that maximizes the caching time while keeping the information within the target area. However, finding the optimal sequence of Carriers would require an exceedingly high computation time due to the lengthy duration of a meaningful mobility trace. We therefore resort to a heuristic, defined as follows. We start at the time instant at which the first Carrier \( B_1 \) crosses into the target area. We introduce the following definitions:

- \( \epsilon \)-rim: the outer rim of the target area comprised between distances \((1 - \epsilon)R\) and \(R\) from the target;
- residual presence time: the time since a Carrier first crosses into the target area, or hands over the information object, whichever occurs later, and the time it gets out of the target area.

By using an omniscient oracle, in the time span while a Carrier \( B_i \) is within the \( \epsilon \)-rim on its way out of the target area, we identify all of its one-hop candidate neighbor nodes. For each of these candidate nodes, we retrieve the residual presence time since its last contact instant with \( B_i \) (while \( B_i \) is within the \( \epsilon \)-rim). The candidate node with the highest residual presence time according to the mobility trace is selected as the next Carrier, \( B_{i+1} \). We repeat the process until the oracle identifies the last Carrier before the end of the mobility trace, \( B_L \). While deriving our results, we set \( \epsilon = 0.1 \).

The objective of the heuristic maximization, i.e., the caching time, is shown in Fig. 16. Although also our framework tries to maximize the information bearing time of a node, it is outperformed by the oracle-based heuristic thanks to its a priori knowledge of node movements. However, our framework, in its turn, performs significantly better than the distance-based strategy. Also, the smaller standard deviation exhibited by the framework with respect to the distance-based strategy belies a much more stable, precise behavior. It is to be noted that, with high vehicle density, there is a large choice of candidate Carriers with significantly different residual presence time, due to the presence of a traffic light at the target location. Hence, the heuristic outperforms ARM-d by knowing in advance the vehicles waiting time at the intersection. As a final remark, we can notice that, when comparing the highway and urban scenarios, the larger choice of vehicles offered by the latter allows the framework to close the gap between its performance and those of the heuristic.

Still, maximizing the caching time is just one of objectives of ARM-d: choosing Carriers within the target area is a fundamental task. Figs. 17 and 18 present the cumulative distribution function (CDF) of Carrier distances from the target: looking at the plots, it can be seen that the distance-based strategy is no match for ARM-d accuracy in using speed and direction in Carrier selection. Furthermore, in terms of capability to confine the information in the target area, the heuristic is matched in most cases, and even outperformed, regardless of the scenario or vehicular density considered.

We point out that the small step at the right end of CDF plots for the low density case (Fig. 17(b)) is due to the relative position of intersections and target: very few candidate nodes can be found between 100 and 500 m from the target, then the likelihood of finding a candidate suddenly surges upward thanks to the four surrounding intersections, 500 m away from the target.

The protocol overhead due to the traffic generated by POLL/BID messages and by the information handover, is shown in Fig. 19. Results are shown as functions of the vehicle density for ARM-d and the distance-based strategy (clearly, we do not have this metric for the heuristic). The overhead induced by ARM-d is negligible, and always lower than that of the simple distance-based strategy. Indeed, the use of the ARM-d index makes information handovers rarer than under the
simple, but less efficient, distance-based policy. Comparing the highway and urban scenarios, the lack of traffic lights in the former leads to a shorter permanence of nodes in the target area, hence more handovers. The effect of the different values of vehicle density is almost irrelevant. We can conclude that not only does ARM-d exploit to the fullest its more complex index computation by efficiently steering information toward the target, but it also does so with minimal overhead.

As far as the actual information dissemination performance of ARM-d is concerned, we evaluate it within a delay-tolerant broadcasting context. The information object thus contains a beacon message that has to be periodically broadcast by the Carrier, with a fair delay tolerance. The range of the beacon message is controlled through a time to live (TTL) field, decremented after every transmission. Nodes receiving the beacon rebroadcast it provided its TTL is not 0, in which case they discard it. The overhead of the rebroadcast process is reduced by means of the Preferred Group Broadcasting (PGB) technique [35], that exploits physical layer information to select relays that are far from the last broadcast source and whose rebroadcast areas do not overlap.

For our performance evaluation, we define an observation area, that is the (circular) region, centered on the target, over which the metrics are computed. Note that such an area serves the sole purpose of metric computation, and it is unrelated to the target area. The following metrics will be employed in the remainder of this section:

- **1-beacon reception probability**: the probability that a node driving through the observation area receives at least one beacon;
- **informed nodes ratio**: the ratio of the number of nodes receiving the beacon to the number of nodes that are within the observation area at the beacon transmission time;
- **first beacon reception delay**: the time elapsed between the instant a node enters the observation area and the instant at which it receives the first beacon broadcast from a Carrier.
For sake of brevity, we focus on the urban scenario with medium lane density, which proved to be quite a challenging environment for the framework in our previous tests. Indeed, simulations in the highway scenario or with higher vehicular densities all showed better results than those outlined below.

We compare the performance of the beaconing application when run over the proposed framework versus the performance of the same application when run by an infrastructure node, located at the target, that periodically broadcasts the beacon.

The plot in Fig. 20(a) shows the 1-beacon reception probability of an infrastructure-based and of a framework-based beaconing application, as a function of the beaconing interval. The TTL is 1 and the observation area radius is 200 m. For beacon intervals up to 10 s, no performance degradation is observed, while longer intervals uniformly worsen the performance. Remarkably, our solution with a small target area ($R = 50$ m) very closely matches the infrastructure result. For larger values of $R$, the 1-beacon reception probability remains quite close to that obtained with infrastructure, yet it is necessarily smaller: indeed, the Carrier and the target may be so far apart that the overlap of the Carrier radio range and the observation area is reduced, hence lowering the reception probability in the observation area.

Fig. 20(b), instead, provides the informed nodes ratio upon a beacon transmission for widening observation areas, for TTL = 1, 3, $R = 50$ m and beaconing interval equal to 5 s. The proposed distributed scheme again provides comparable performance to the infrastructure case no matter the width of the observation area. As expected, allowing the beacon to travel further increases the hit ratio for both the ARM and the infrastructure-based frameworks.

We conclude that the beaconing service using our framework could be provided at the same quality level as that of an infrastructure-based application, but without the constraints and costs of a fixed installation, and with greater flexibility in selecting the target area.

### 6.3. Data collection

We now evaluate the performance of the ARM-c index for the collection of sensor measurements through vehicles. We consider the highway and urban scenario configurations depicted in Figs. 14(b), 15(b), respectively. There, the slashed-line arrow represents the direction of movement of the Carrier role within the gray-shadowed road pipe of width $w_p$. The road pipe width is equal to $w_p = 20$ m; the maximum acceptable lateral and longitudinal errors are equal to $E_w = E_l = 1$ km.

The Carrier role is first assigned to a vehicle by a road-side unit placed at the central intersection and, then, is driven along the highlighted route, which is approximately 2 km long. The sensor network is composed of 65 nodes, which have a radio range of 20 m and are regularly placed along the route at a fixed distance equal to 30 m. We assume that, at each contact opportunity, each sensor transfers 10 bytes of data to the Carrier, and that the information stored by the Carrier is updated with the new data.

We start by evaluating the capability of the ARM-c index to maintain the Carrier along the desired path. Fig. 21 shows the CDF of the Carrier distance from the ideal trajectory in a worst case, i.e., when the desired cruise speed is 10 km/h, for different vehicle densities. We observe that in both the highway (left plot) and the urban (right plot) scenarios, our scheme manages to maintain the Carrier within the road pipe (i.e., within distance $w_p$ from the ideal trajectory) with very high probability, even when the vehicle lane density is low. Also, the Carrier distance from the ideal trajectory never exceeds 55 m, which is well below the value of $E_w = 1$ km; it follows that there is never the need to drop the information collected from the sensors or to give it away to a roadside unit.

Beside the good performance in terms of distance from the desired path over time, Fig. 22 shows the ability of the ARM-c index to follow the desired cruise speed. We note that our solution achieves excellent performance, even for very different values of the average vehicle velocity, until a certain value of desired velocity, beyond which it does not significantly increase. The saturation point depends on the scenario and on connectivity level of the network, thus on the vehicle lane density. For example, in the highway scenario where the vehicle speed is higher, the saturation point is about 80 km/h and 100 km/h for, respectively, low and medium/high density values; in the urban scenario, it is about 50 km/h for low density and about 80 km/h for the medium/high density value.
As can be observed in Figs. 23 and 24, the ARM-c handover rate (e.g., the number of handovers per second) and the control message rate (e.g., the number of bid/poll messages per second) are lower in the highway than in the urban scenario, and, for a fixed mobility environment, we have better performance when the desired cruise speed is close to the average vehicle speed. However, the obtained values are practically negligible in both environments and for any value of desired cruise speed and lane density.

Next, we evaluate the average contact time between the Carrier and the roadside sensor nodes. Looking at Fig. 25, we observe that, under both the highway and the urban scenarios, the ARM-c index allows the contact time interval to vary as needed, by changing just one system parameter, i.e., the desired cruise speed. We point out that, in the urban scenario and for low vehicle density, the contact time is larger than for medium/large densities, since, especially in proximity of road intersections, several handovers are needed before a suitable Carrier is found. These handovers take place between vehicles that are all close to the intersection (with either a traffic light or a stop sign) thus making the carrier linger longer within a sensor radio range.
Fig. 24. ARM-c: average poll/bid messages rate per time unit.

Fig. 25. ARM-c: average contact time between sensor and the Carrier.

Fig. 26. ARM-c: average time interval between two consecutive contact opportunities between the Carrier and a given sensor.

Fig. 26 shows the average inter-contact time, i.e., the time that elapses between two consecutive contact opportunities between the Carrier and a given sensor. Again, we see that, by simply changing the desired cruise speed, the inter-contact time can be significantly reduced or prolonged. Thus, fuzzy logic allows a control of the Carrier’s round trip time, which makes the collection data procedure adaptive to the sensor nodes buffer capability, the reliability level of the data transmissions, and the amount of data to be transferred. More importantly, through the ARM-c index, we can predict when the Carrier will be in visibility of each sensor. This allows the implementation of sleep/wake-up scheduling schemes, which let sensors save energy and wake up only when needed to send their data to the Carrier.
7. Conclusions

We introduced the original concept of Application-level Role Mobility (ARM), and showed how the ARM framework allows mobile nodes to be effectively used for information dissemination and data collection in vehicular networks. Through the definition of specific role handover indices, ARM correctly selects which node is best suited to perform the task demanded by the application. The result is the possibility of running such an application in a completely distributed way, with low overhead and without any need for a fixed infrastructure (in the information dissemination case) or dedicated nodes performing controllable movements (in data collection scenario).

Acknowledgements

This work was supported by the European Union through the “myMED” project, funded through the ALCOTRA action.

References

Diego Borsetti received his M.S. and Ph.D. degrees in Telecommunication Engineering from Politecnico di Torino, Italy, in 2005 and 2010 respectively. From 2005 to 2007 he worked at Motorola Technology Center in Torino as a software engineer. Currently, he is a researcher at the Uwicore Laboratory, University of Elche, Spain, working on the iTetris European project. His research interests include wireless and mobile networks with a special emphasis on vehicular computing systems.

Marco Fiore is an assistant professor at INSA Lyon. He obtained Master degrees in Computer Science from University of Illinois at Chicago, in 2003, and Politecnico di Torino, in 2004. He received his Ph.D. from Politecnico of Torino, in 2008. He has been a visiting researcher at Rice University, in 2006 and 2007, and Universitat Politécnica de Catalunya, in 2008. His current research focuses on the characterization and exploitation of mobility in wireless environments, and on the security and autoconfiguration of ad-hoc networks.

Claudio Casetti graduated in Electrical Engineering from Politecnico di Torino in 1992 and received his Ph.D. in Electronic Engineering from the same institution in 1997. He is an Assistant Professor at the Dipartimento di Elettronica e Telecomunicazioni of Politecnico di Torino. He has coauthored more than 120 journal and conference papers in the fields of networking and holds three patents. His interests focus on ad hoc wireless networks and vehicular networks.

Carla-Fabiana Chiasserini received the electrical engineering degree (summa cum laude) from the University of Florence, Florence, Italy, in 1996. She received the Ph.D. degree in 1999 from the Politecnico di Torino, Torino, Italy. Since then, she has been with the Department of Electrical Engineering, Politecnico di Torino, where she is currently an Associate Professor. From 1998 to 2003, she has worked as a visiting researcher with the University of California at San Diego. Her research interests include architectures, protocols, and performance analysis of wireless networks for integrated multimedia services. Dr. Chiasserini has published over 170 papers in prestigious journals and leading international conferences, and she is the co-author of three patents. She serves as Associated Editor of Ad Hoc Networks Journal (Elsevier), Computer Communications (Elsevier), IEEE Wireless Communications Magazine, IEEE Transactions on Wireless Communications and IEEE Communications Letters.