The Trap Coverage Area Protocol for Scalable Vehicular Target Tracking

Paolo Bellavista, *Senior, IEEE*, Azzedine Boukerche, *Fellow, IEEE*, Tommaso Campanella, and Luca Foschini, *Member, IEEE*

***Abstract*— Vehicle target tracking is a sub-field of increasing and increasing interest in the vehicular networking research area, in particular for its potential application in dense urban areas with low associated costs, e.g., by exploiting existing monitoring infrastructures and cooperative collaboration of regular vehicles. Inspired by the concept of trap coverage area, we have originally designed and implemented an original protocol for vehicle tracking in wide-scale urban scenarios, called TCAP. TCAP is capable of achieving the needed performance while exploiting a limited number of inexpensive sensors (e.g., public-authority cameras already installed at intersections for traffic monitoring), and opportunistic vehicle collaboration, with high scalability and low overhead if compared with state-of-the-art literature.In particular, the wide set of reported results show i) the suitability of our TCAP tracking in the challenging urban conditions of high density of vehicles, ii) the very weak dependency of TCAP performance from topology changes/constraints (e.g., street lengths and speed limits), iii) the TCAP capability of self-adapting to differentiated runtime conditions.**

***Index Terms*— Scalability, Target Tracking, Vehicular Ad Hoc Networks, Vehicular Applications, Vehicular Communication Protocols, Vehicular Protocol Tuning.**

I. INTRODUCTION

IN the last years, Vehicular Ad hoc Networks (VANets) have received relevant interest from the research community [1] [2]: they have become a very active area of research, standardization, and development thanks to their tremendous potential to improve vehicle and road safety, traffic efficiency, as well as comfort to both drivers and passengers. VANet applications can be primarily divided into two categories: *safety*- and *comfort-related* (with the latter sometimes indicated as *commercial*) [3]. The former includes any type of applications that can increase vehicles' and drivers' safety while on road; the latter focuses on enriching the overall driving experience by providing services with more limited reliability/latency requirements to drivers and passengers, e.g., "infotainment" applications.

P. Bellavista, T. Campanella, and L. Foschini are with the Dipartimento di Informatica-Scienza e Ingegneria (DISI), University of Bologna, 40122 Bologna, Italy (e-mail: {paolo.bellavista, luca.foschini}@unibo.it, tommaso.campanella@studio.unibo.it).

A. Boukerche is with the School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: boukerch@site.uottawa.ca).

Target tracking, defined as the capability to detect and continuously trace the state of one (or more) vehicles [4], has nowadays started to play a key role in vehicular networking, since a growing variety of envisioned applications is actually relying on such ability. Although deeply explored in Wireless Sensor Networks (WSNs) [5], at the moment only few works have attempted to tackle this challenging topic in vehicular environments. In [4], Ramos et al. addressed the main aspects and challenges for cooperatively tracking one or multiple targets in VANets. They argued that Cooperative Target Tracking systems might augment driver's perception of the surrounding context and increase the comfort and safety of the driving experience. In [6] and [7] instead, Reeza et al. proposed a tracking system to trace an on-the-run vehicle in a metropolitan area, by means of cameras mounted on cars. Although promising, these papers present seminal work not yet ready for wide-scale and industrial applicability, with support infrastructures that have proved to exhibit limited scalability whenever large deployment scenarios are involved. Moreover, both proposals heavily rely on the participation of non-regular vehicles equipped with specific On-Board Units (OBUs) in order to contribute to the collaborative tracking goals.

Considered the above limitations of the state-of-the-art literature in the field, we have decided to work on the proposal of an innovative, efficient, and scalable lightweight target tracking protocol, called Trap Coverage Area Protocol (TCAP), originally presented in this paper. The primary goal of TCAP is to let authorities keep tracking of any desired vehicle in a metropolitan area by means of (typically publicly controlled and already available) cameras deployed at intersections, under the simple assumption that they are capable of detecting a car by its License Plate Number (LPN). The central part of our TCAP proposal is a novel target tracking technique, which focus on finding a suitable trade-off between the desired tracking performance and the cost of the infrastructure needed to trace the target. Our technique is based on the abstract concept of Trap Coverage, introduced by Balister et al. in [8] for other deployment environments and other application purposes (typically in WSN coverage problems and WSN applications), for scenarios where more traditional blanket coverage approaches cannot work and therefore coverage holes are present [9]. The basic idea is that the presence of coverage holes might not be necessarily a problem, since any target could move at most a known

displacement inside it before being detected by a WSN, *thus making the target trapped inside the hole*. We will call TCA an area demarcated by two intersecting roads, inside which a vehicle is trapped by means of cameras deployed at the end of each enabled road segment. In the paper we originally claim the suitability of addressing the problem of vehicle tracking as the issue of sequentially creating new TCAs each time a target moves from a detected trap area to a new one.

By following the above concepts and guidelines, we have designed, implemented, and validated the novel TCAP for scalable vehicle target tracking. TCAP can use a relatively limited number of inexpensive sensors (e.g., public-authority cameras already installed at intersections for traffic monitoring) with no additional ad-hoc infrastructure, by exhibiting good scalability in large scale deployment scenarios, if compared with state-of-the-art literature, such as [4] where instead complex and expensive target tracking infrastructure is needed. To enable lightweight camera coordination and communication, TCAP employs Roadside Routers (RRs), deployed at intersections and responsible for controlling the cameras in their TCAs: there is no assumption about the possibility for RRs to communicate each other directly; on the contrary, the idea is that RR communications are opportunistically enabled via passing-on vehicles equipped with OBUs supporting TCAP. In other words, this allows TCAP to exploit the existing sensor infrastructure, already deployed for other motivations, without requiring any dedicated communication infrastructure, but only single-hop wireless connectivity capabilities of RRs and OBUs.

In addition, the paper presents how we have modularly designed and implemented our TCAP solution in an incremental way in order to show the different contributions of the different TCAP features, which can be dis/enabled to experiment different properties, behaviors, and protocol performance tradeoffs, up to its most complete and finalized version. In particular, we first describe how TCAP exploits a modified version of GPSR [10] as the basic underlying routing protocol. Then, we show how higher reliability can be achieved via proper ad-hoc extensions while maintaining high efficiency and dynamicity in creating/destroying TCAs of interest. Based on that, we present how it is possible to introduce a controlled level of redundancy by concurrently exploiting multiple paths to route TCAP packets, as well as an additional opportunistic routing extension is appropriate when multiple intersections can benefit from being reached at the same time. Finally, a cross-layer optimization is introduced, with relevant advantages in terms of overall TCAP reliability, efficiency, and dynamicity.

The remainder of the paper is organized as follows. Section II presents the problem formulation and adopted system model. Section III describes our original TCAP proposal, while Section IV gives insights about its efficient and modular implementation, in terms of layered features that can be dis/enabled to experiment different characteristics and performance tradeoffs of the protocol. Section V reports extensive experimental results, related to TCAP with its different layered features, in order to give quantitative insights about the advantages and costs associated with the different characteristics of the protocol. Related work, conclusive remarks, and ongoing directions of research work end the paper..

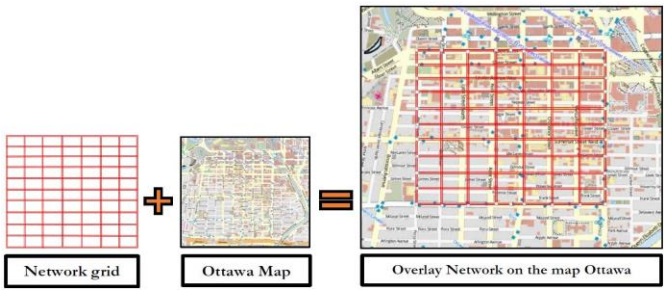


Fig. 1. TCAP topology and overlay network.

II. Problem Formulation and System Model

It is widely recognized that there is a growing interest for police authorities to have the opportunity to trace an unaware car in a specific area of interest within an urban scenario. The vehicle tracking solution should be able to monitor and collect information about target movements and behaviors, without that the observed vehicles realize of being traced, in order to let the authorities either access their final destination (in the area of interest) or to catch them, whenever desired, by means of roadblocks. It is evident that the tracking solution can benefit from involving minimal police resources (e.g., no police cars), while cameras already deployed throughout the city for other purposes could be instead a relevant and limited-cost resource to try to involve, by exploiting available, low cost, and consolidated algorithms for LPN recognition.

Usually, once detected the target for the first time, vehicle tracking systems are asked to keep tracking the target vehicle until it is out of the area of interest. In this phase, target information must be regularly forwarded to either a specific server or, in alternative, to police vehicles nearby the latest detection. Typically the tracking solution cannot make any strong simplifying assumption on the target behavior (e.g., future movements or expected destination), whilst it is usual that the unaware target tends to respect the local driving regulations (e.g., speed limits) in order not to attract the police attention.

A. System Model

In our system model, we focus on considering VANet solutions specifically optimized for a downtown scenario, which is usually the most relevant and challenging one for vehicle tracking applications. In particular, TCAP exploits an overlay network (see Fig. 1), organized based on a simple 2-D grid plan, so that TCAP might be easily applicable to other cities with similar characteristics (we have decided to evaluate our proposal on the realistically simulated environment of the city of Ottawa, with real road topologies and realistic mobility traces, as detailed in Section V). The considered Manhattan-like road topology is motivated by several aspects: first, at this stage of the TCAP proposal our primary objective is showing the feasibility and effectiveness of our novel target tracking

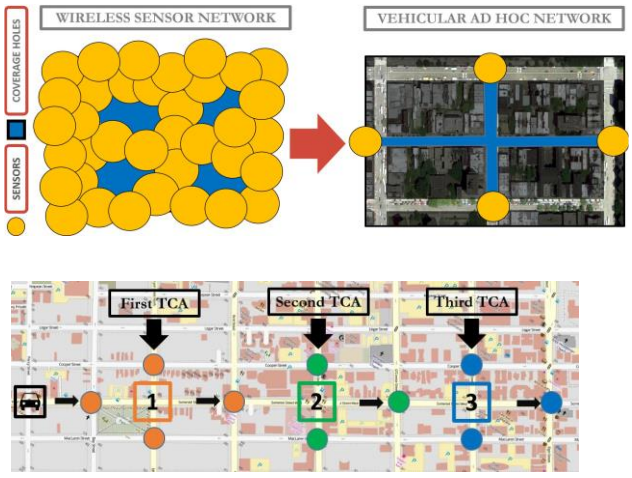


Fig. 2. Trap coverage concept applied to VANets.

technique in a realistic scenario; second, this kind of topology is realistic and easily adaptable to many urban scenarios of practical relevance, e.g., in several North-American cities; third, a realistic but quite simple deployment scenario simplifies our research work of clearly understanding how the different aspects and properties of TCAP affect its behavior and performance, with no masking effects due to topology dependence.

In particular, we consider a grid composed by two-way streets and intersections (Fig. 1), where a vehicle can choose up to three different directions, plus the one it is coming from. We call *zone* the area including any street between two intersections; a zone is uniquely identified by its central coordinates on a map (if different maps are used, our TCAP support admits a configurable precision tolerance). Intersections follow the same specification about coordinates. We suppose that any target can move in the scenario with no simplifying assumptions and restrictions, except the respect of traffic regulations: a vehicle is allowed to perform a U-turn where permitted, and we do not consider the possibility that it might pull over while being tracked (this is currently under consideration as future work).

Before delving into the detailed description of the primary entities of our system model and to fully understand them, let us introduce first our modification to the known concept of Trap Coverage [8], which is at the basis of our original tracking solution and TCAP.

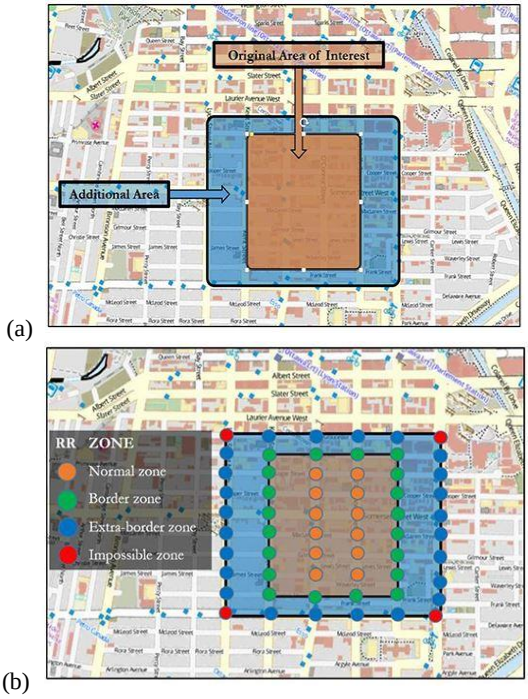
B. *Extending the Trap Coverage Concept for Vehicle Target Tracking*

In the existing literature the Trap Coverage model generalizes full coverage by allowing coverage holes of bounded diameter, so that, even if not enough devices were deployed in an observation area, a target can be in any case "trapped" inside a hole (the target is detected at the moment of leaving the uncovered area). We claim that this approach can be fruitfully adapted to VANets: as Fig. 2(a) shows, if compared with WSNs where a target can move arbitrarily inside an area, given the road topology constraints a vehicle can only move inside existing streets; therefore, a much lower number of sensors is needed to cover the same hole, i.e., 4 in our considered topology, by making trap coverage techniques eligible to be applied efficiently in real-life scenarios. Therefore, we define Trap Coverage Area (TCA) as the area defined by two intersecting roads, inside which a moving target vehicle is trapped by means of sensors deployed at the end of each road (intersection). This means that, for example, from the moment a sensor at an intersection has detected the target, such vehicle can be guaranteed of being inside the TCA, until it will not be detected again by one of the four sensors serving the TCA. In addition, once a target has moved out of a TCA, one of the 4 sensors has detected such event and can work to determine a new TCA, connected with the old one by the intersection where the detection has last occurred, as illustrated in Fig. 2(b). This is just a simple example of scenario to show the basic approach idea of continuously tracking a target vehicle by means of sequentially creating new TCAs each time the target moves out of an old one, one after the other.

This tracking technique, at the basis of our proposed TCAP, defines a new concept of "target tracking", together with the associated goal of finding a dynamic and proper trade-off between tracking performance and infrastructural costs. In fact, as we will demonstrate later, TCAP deployment can be easily settled to decrease location accuracy (TCA with coarser grain) by maintaining the same capability of continuously tracking a vehicle of other solutions in the literature [6] [7], and additionally with the exploitation of a limited number of camera sensors.

In addition to cameras (indicated in the following as Roadside Cameras – RCs), our system model includes two other entities, i.e., Roadside Routers (RRs) and On-Board Units (OBUs):

- **RCs** are fixed cameras deployed at intersections. We assume that a specific number of RCs is already present in the area of interest for other purposes, e.g., for road traffic control. Since no full coverage can be guaranteed when exploiting existing sensor infrastructure, as we would like to do with TCAP for imposing only loose and lazy constraints on sensing infrastructure availability, in the following we assume, for the sake of simplicity, that one intersection every two hosts RCs capable of detecting a target based on its LPN. We also suppose the availability of a "detection service" at any RC-enabled intersection, simply capable of receiving requests for detection of a vehicle with a given LPN and of returning replies with target information once one of the cameras detects it. Reply information includes position, speed, time, and direction at detection time (RC connectivity is provided by the following RRs);
- **RRs** are devices with wireless communication capabilities, deployed at each intersection, and logically assigned to the local detection service at the same junction. Each RR is the only responsible for communicating with the local camera sub-system; IEEE 802.11p is used for single-hop communication with other routers, OBUs, and RCs when and if they are in the RR radio coverage area; each RR



(a)

(b)

Fig.3. TCAP tracking area.

knows its own geographical position. In other words and shortly, TCAP RRs have the role of extending camera control over a target detection among different intersections, thus by introducing a new utilization (target tracking) of such sensors with no additional infrastructure costs. In addition, as better detailed in Section III, RRs are the only entities in charge of managing TCAs created by our tracking solutions. Moreover, since TCAP interacts with the local detection service uniquely through RRs, adding a new intersection to the current TCAP participants only implies the "hot deploy" of an RR at that intersection, with no need of further explicit management operations. To support also the case when an RR has to communicate the detected target data to an Internet-accessible remote server, some RRs may have also Internet connectivity, but this is the only case where TCAP exploits traditional Internet access for a very limited fraction of RRs;

- **OBUs** are devices with routing capabilities and capable of single-hop wireless connectivity (i.e., IEEE 802.11p) deployed on participating vehicles. As usual for VANets, they have no strict constraints on energy consumption. The primary role of TCAP OBUs is to serve as intermediate nodes among RRs, given they are rarely in direct radio coverage the one of the other. As it is more realistic, we do not make any assumption on the fact that all vehicles should be equipped with OBUs: the idea is that, even if with a limited penetration rate of OBUs in the vehicle population, TCAP can properly work; the performance results will show which is the minimum number of vehicles to comply with common and regular tracking requirements, and how the TCAP behavior depends on OBU penetration rate. In addition, each vehicle is supposed to be GPS-assisted, thus allowing the OBU to determine vehicle position and driving direction with a given degree of approximation; the same mobility data can be obtained, with different precision/accuracy levels, also via other motion sensors (e.g., Android-hosted accelerators and gyroscopes of smartphones that jointly move with the vehicle), which are anyway not integrated in the implemented OBU prototype at the moment.

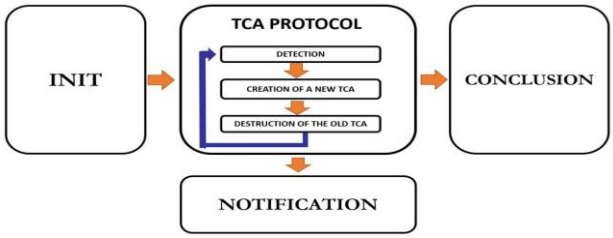


Fig. 4. TCAP protocol phases.

III. An Innovative Vehicle Tracking Solution: General Overview

In our proposal, we define a Tracking Area (TA) as the region that includes the area of interest and all the intersections one junction outside such area, to let avoid losing the target before it leaves the region, as shown in Fig. 3(a).

A TA consists of four zones, each of them defining the behavior that an RR will assume to contribute to TCAP (see Fig. 3(b)):

1) *Normal* zone - this region corresponds to the area of interest, without its border. An RR in this zone will behave independently from the current direction of the target vehicle;
2) *Border* zone - this region, instead, consists of the first intersections outside the previous region. Depending on either the vehicle is moving out or in the zone, TCAP either concludes or keeps tracking the target;
3) *Extra-Border* zone - this third region consists of those remaining intersections within the enlarged TCA; once one of the cameras here has detected the target, the associated RR will end TCAP involvement for that TA;
4) *Impossible* zone - this last zone is represented by the intersections at the four edges of the *extra-border* zone; a RR deployed in such region does not participate to TCAP operations.

Fig. 4 shows how our TCAP solution is the central element that can be employed in a larger tracking solution consisting of different phases: *Init*, *TCAP*, *Notification*, and *Conclusion*. To fully understand the details of our proposed TCAP, we now quickly introduce each of these phases, even though only TCAP remains the original subject focus of this paper:

1) *Init* represents the initial phase when a pair <TA, LPN> are given as input to our tracking solution. At this stage, the RRs in the TA are notified of a (service) request for tracking the LPN vehicle, and all the corresponding cameras start looking for the target. Upon detecting the car for the first time, RRs are notified and this phase concludes, with the overall system moving to the following *TCAP* phase;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access

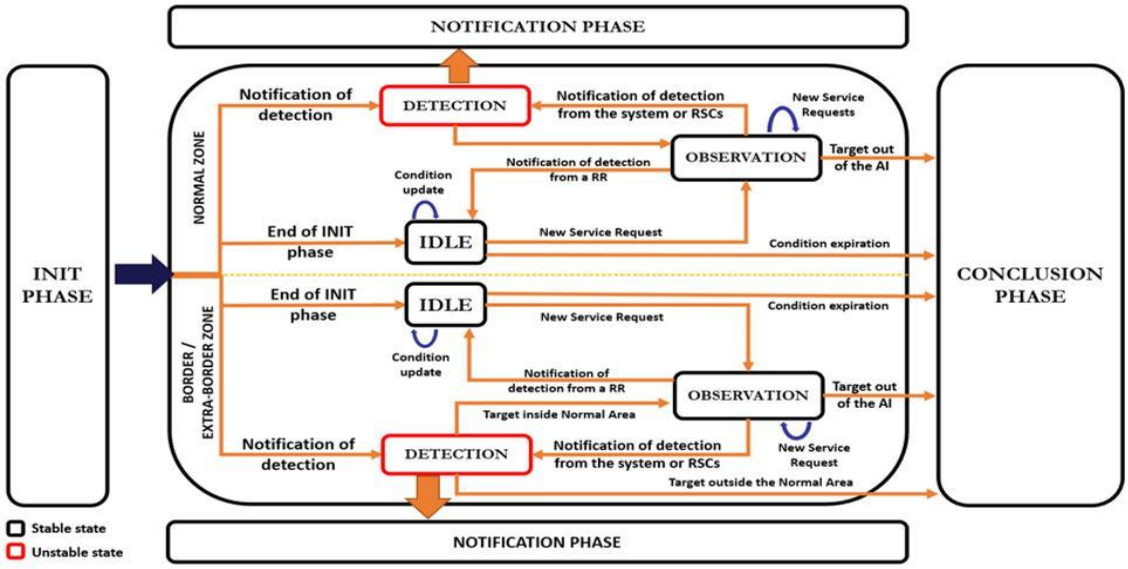


Fig.5. RR logical architecture.

2) The *TCAP* phase, fully described later and original proposal of this paper, begin once any RR in the TA receives a detection notification from its RCs. The corresponding TCA creation is commanded and three steps are then executed:
   a) the RR where the detection occurred sends a service request to three designated intersections (the other 3 RRs in the TA) to "create" a new TCA;
   b) the same RR forwards the detection information (direction, speed, position, and time of detection) to the successive Notification phase;
   c) the same RR notifies the intersections that are involved in the previous TCA to command the removal of it.

   These three steps are executed upon every detection until the target has moved out of the TCA and TCAP is therefore concluded in the area;
3) *Notification* is the third phase in which the RR (where the last detection occurred) is responsible for dispatching the target information to a specific server on the Internet (sink of the target tracking info). In case the router has not direct access to the Internet, a dedicated protocol, e.g., for multi-hop VANet routing, is executed. Let us note that TCAP is completely independent from this phase, which could be completed by integrating with widely available solutions for VANet node connectivity to the Internet;
4) the *Conclusion* phase determines the end of the tracking service. Every RR stops accepting any tracing request for the same LPN, until a new Init phase is issued. Since we do not want TCAP to make any centralized decision (as it might limit its applicability to a large set of future real-life VANet applications), we have currently implemented this phase by employing timers with lease mechanisms, to let RRs autonomously choose when the protocol has terminated.

As graphically depicted in Fig. 5, each RR can assume different states while executing TCAP depending on various factors:

- *Idle* - an RR is in this state when no requests are being served by its RC sub-system. It can be reached either as the result of concluding the Init phase (target detected at another RR) or due to the localization of the vehicle at another intersection of the TCA (coming from the Observation state below). An RR remains in this state until a given condition is fulfilled (e.g., through a lease mechanism) or when it ends the tracking protocol by moving to the conclusion phase (independently from the actual zone);
- *Observation* - the second state is reached by an RR when either a new service request is received (from *Idle*) or the vehicle is detected at its intersection while remaining in the *Normal* zone (from *Detection*). One or more requests might be served at the same time, as an RR could be awaiting the confirmation of destruction of a previous TCA. At the reception of a detection notification, the RR moves to the *Detection* state below or, in the case the notification was received from another RR, it either passes to *Idle* or ends its active role in TCAP (moving to the *Conclusion* phase), depending on the target is either outside the *Border* zone or not;
- *Detection* - an RR moves to the *Detection* state either from *Init* (when the target is detected at its intersection) or from *Observation* (as the result of a notification from RCs, used also in the new TCA). This state is more temporary than the previous ones, as an RR, upon finishing its detection work, will either move back to *Observation* or conclude the protocol depending on its location in the TA. Finally, this state is responsible for forwarding the detected target data to the Notification phase.

When either creating or destroying a TCA, four border and one central RRs are involved. The central RR is located at the junction where the two roads of a TCA are intersecting, which makes it an intersection far away from each of the other four

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access

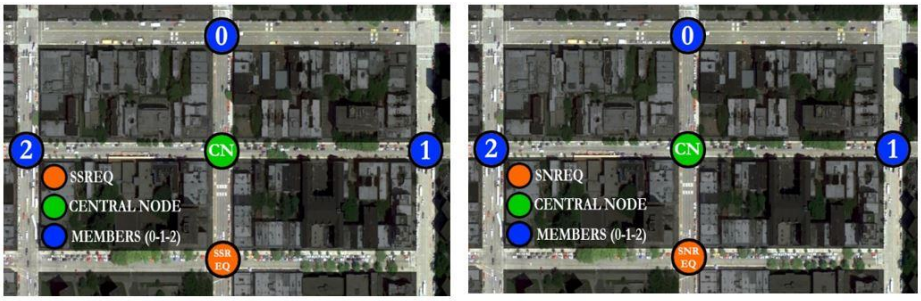


Fig. 6. Member definition in case of creation (left) and destruction (right) of a TCA.

RRs. Due to its central position, it is beneficial that the central RR is responsible for distributing each TCAP message within the TCA in both phases. As Fig. 6 shows, during the creation phase, an RR assumes the role of Source of Service Request (SSREQ) if it is located at the intersection where the last detection has occurred; this entity is responsible for dispatching a service request (and awaiting a notification of reception) to the other three RRs, which in turn assume the role of "members" (each of them is distinguished by a number, e.g. Member0). In the case of TCA destruction, the SSREQ RR of the previous phase assumes also the role of Source of Notification Request (SNREQ), as the creation of a new coverage area passes necessarily through the destruction of the previous one. Regarding the RR members, Member1 and Member2 during TCA creation (the TCA now going to be destroyed) keep their role of members (with swapped numbers), while the RR member serving as SSREQ assumes the role of Member0. In this phase, SNREQ is responsible for dispatching detection notifications to let the three other RR members stop their active TCAP role. Finally, any RR member stores a list of neighbor RRs at most two intersections far away, organized by groups of four and identified by their direction from the list owner. This allows a RR to know immediately which intersections to contact when either creating or destroying a TCA.

IV. The TCAP Modular Design and Implementation

This section describes the TCAP design principles and some implementation insights toward its effective, efficient, and scalable implementation. Starting from the requirements, there are two of them necessary to determine when a TCA has been successfully created or destroyed:

1) Every RR member of the associated TCA must receive at least one request (SREQ/NREQ) from its corresponding SSREQ/SNREQ;
2) A SSREQ/SNREQ must receive a request notification from each RR member of the same TCA, within given time constraints.

Based on these specifications, our TCAP adopts the following design principles:

- *Reliability* – the application domain constraints of the targeted vehicle tracking impose high levels of reliability. In particular, every TCA creation and destruction operation should be designed to be reliable to avoid significant decrease in relevant TCAP performance indicators, as detailed in the following section;
- *Efficiency* – while not sacrificing reliability, TCAP should be efficient, with limited overhead for enabling the envisioned deployment over wide-scale environments and with performance results satisfying the challenging requirements of vehicle tracking;
- *Dynamicity* – TCAP should be capable of dynamically adapting to different deployment conditions and scenarios (for example, when no cars are temporarily moving on a road), by modifying reliability and efficiency strategies in order to continue to achieve the targeted performance goals in changing and unforeseen execution environments.

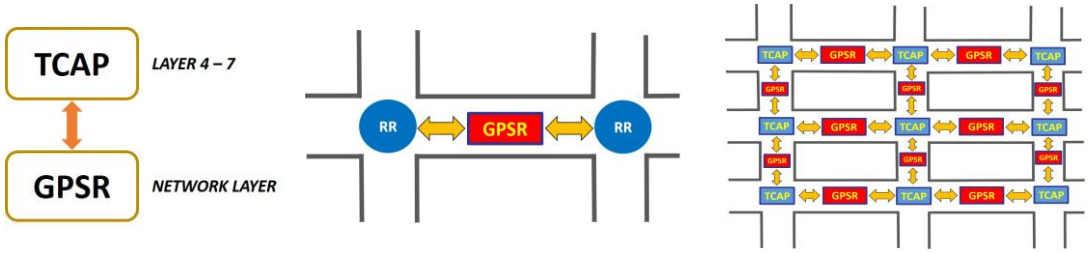
The pseudo code of Algorithms I and II in the annex provide a full description of the fully fledged version of the implemented TCAP. To facilitate the full understanding of the proposed protocol, let us start by noting that TCAP packets are of 9 different types, efficiently identified by a specific field in the header (see Fig. 7). Header and packet format is not reported here for the sake of briefness; however, it is worth mentioning that storing the least necessary information required by the protocol was a major focus during TCAP development (a TCAP packet is at most as large as the minimum MTU for IPv4, as reasonably suggested in IETF RFC791).

A. TCAP Routing

About TCAP routing decisions, we identify two primary logical parts: one dedicated to exchanging packets between neighbor intersections and the other to routing messages among the four TCA RRs. The TCAP solution is traditionally ISO OSI layered, with each packet delivered from one RR to another by means of a routing protocol at the network layer,

TCAP packets
(1) SREQ
(2) NREQ
(3) NOT_SREQ
(4) NOT_NREQ
(5) PB_FWD_RQST (PBRQ)
(6) PB_FWD_REPLY (PBRY)
(7) UNABLE_TO_FORWARD (UTF)
(8) UNABLE_TO_FORWARD_PB_RQST (UTF_PBRQ)
(9) UNABLE_TO_FORWARD_PB_REPLY (UTF_PBRY)

Fig. 7. TCAP packet types.



(A) Layering structure *(B) Packet delivery between RRs* *(C) Message routing within a TCA*

Fig. 8. Routing in TCA.

while the upper levels are responsible for message routing within a TCA (see Fig. 8). Therefore, the integrated network-layer protocol (i.e., GPSR in the following) can be easily replaced with another one, as its role in TCAP is limited to delivering packets between neighbor intersections (loose coupling between TCAP and the underlying network-layer routing).

GPSR Modifications

In the current TCAP implementation we have decided to integrate GPSR because of its simplicity and limited overhead, together with its capability of finding valid routes quickly as network topology changes. The GPSR role in TCAP is to deliver packets between neighbor intersections, by using OBU-equipped vehicles as intermediate hops. In particular, at the moment of packet dispatching, GPSR behaves as follows:

1. First, the application-layer TCAP support forwards the packet downwards to GPSR at the network layer;
2. Then, GPSR routes the packet to its destination, i.e., a neighbor intersection;
3. Finally, upon receiving the packet at destination, GPSR eventually forwards it upwards to the application-layer TCAP, where further routing decisions might be taken.

As GPSR suffers from known issues when trivially applied to VANets [11], in TCAP we have modified the standard GPSR protocol as follows:

1. Only the GPSR "greedy" part is used as routing technique (perimeter mode is deactivated). This has no negative consequences because GPSR utilization here is restricted to a single connected area with no concave holes. Once reached the RR destination, the next routing decision is taken by TCAP. Consequently, network disconnections do not occur due to the presence of obstacles, and GPSR has higher routing efficiency as multiple routing alternatives do not exist;
2. Given the high dynamicity of the considered VANets, every OBU and RR receive coordinate updates and beacon messages in any exchanged TCAP packet (i.e., in some sense GPSR is used in both reactive and proactive ways in the TCAP exploitation).

In addition, GPSR has been further revised to increase packet delivery ratio and, consequently, the overall TCAP reliability. In particular, we have specialized GPSR by exploiting the fact that we are willing to use it only on top of the IEEE 802.11p MAC layer (*cross-layer coupling*), by adopting specialized settings to reduce sources of unreliability, such as collisions, presence of obstacles, or dynamic movements of destinations too far away from sources. In particular, after extensive simulations and in-the-field experimental results, we have chosen to configure 100m as our TCAP "reference" distance (around 80% chances of successfully delivering a packet in single-hop communications between two moving vehicles); a forwarding decision is taken only if the single-hop destination neighbor is closer than this reference distance.

Based on these considerations, GPSR has been modified to operate in a new modality, called *safe mode* in the following, with this behavior:

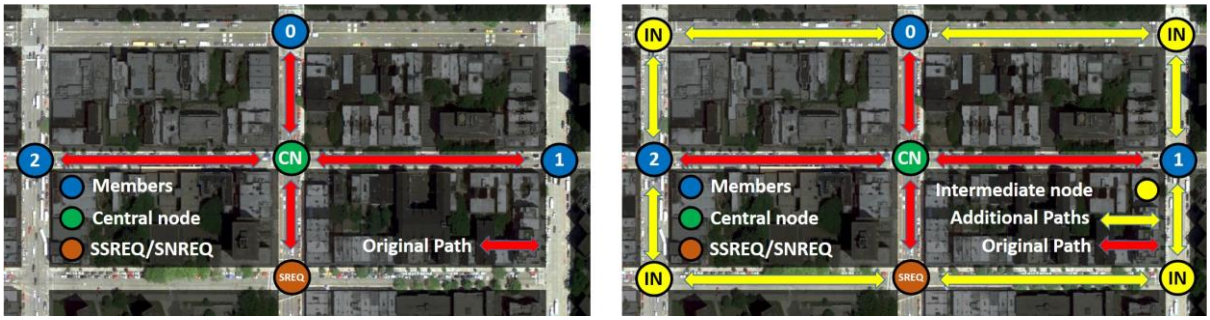
1. Whenever a packet has to be dispatched, a node picks the closest candidate to packet destination. Our modified GPSR chooses the two best candidates within a range of *(100+gap)*m from the actual node. The arbitrary gap gives the possibility of changing the applied range whenever needed, for instance in an application-specific way or when affected by the current conditions of the deployment scenario (e.g., the range could be widened when the vehicle is driving in a street where there are no obstacles and the number of other cars is negligible);
2. GPSR forwards the packet to the first best node only in case its own information (coordinates and time of reception) does not meet at least one of the following criteria:

   $\boldsymbol{i}$) ((**distance_from_source** $\geq$ **70% threshold_distance**) && ((**GPSR_CURRENT** $-$ **ts_next_hop**) $\geq$ **75% beaconing period**) && (**nexthop_second** ! $=$ $-$**1**))

   ii) ((**distance_from_source** $\geq$ **85% threshold_distance**) && ((**GPSR_CURRENT** $-$ **ts_next_hop**) $\geq$ **75% beaconing_period**))

   iii) ((**distance_from_source** $\geq$ **85% threshold_distance**) && ((**GPSR_CURRENT** $-$ **ts_next_hop**) $\geq$ **45% beaconing_period**) && (**nexthop_second** ! $=$ $-$**1**))

   This will tie the choice of the next hop to the time passed since its last received information and the estimated distance from the actual hop. Moreover, GPSR will try to forward a message in case the best node has not met the criteria by a short interval of tolerance (around 5%) and a second best next hop is unavailable, in which case the



Original TCA (a) *Modified TCA (b)*

Fig. 9. TCAP modifications to add redundancy.

packet should be dropped as the delivery confidence is too low. In this way, a higher delivery ratio can be achieved compared to its original implementation because a controlled level of redundancy is inserted (see the detailed discussion later and the reported experimental results). As a result of these modifications, GPSR will deliberately use a higher number of hops to deliver a message: on the one hand, this increases packet latency but, on the other hand, makes higher the probability that a packet is delivered to its destination within a specific time frame (strong and application-specific *reliability* constraints).

RR Role in Routing Decisions

Every RR, whose work spans from layer 4 to layer 7 of the ISO OSI stack, is responsible for determining the next intersection to which messages have to be forwarded (see Fig. 8(c)). For instance, when SSREQ has to send a message to Member0, RR specifies the type of message and its logical destination (Member0), without mentioning the exact intersection to be delivered at. The only junction that SSREQ will eventually specify is the one toward which GPSR has to route the packet. In particular, the next RR is chosen based only on the information carried within the received packet and on the list of the neighbors of the node where the routing decision has been taken. In this way, every RR is capable of taking autonomous routing decisions based only on local information. As a result, our RRs assume a role depending only on the content of the received packets, without global knowledge of TCAP, and routing decisions are taken at every intersection, regardless of the content or type of the message (such decisions are independent of GPSR, as they are taken only at the upper layers, where TCAP is located).

B. Redundancy

Considering how a TCA is defined, to reach every Member, a SSREQ/SNREQ actually exploits only one street as its routing path (Fig. 9(a)). Since there is no guarantee of finding OBUs in any of those streets, more paths are required to increase the chances of reaching every Member and consequently the overall reliability. From this observation, to increase reliability via redundancy we have decided to exploit different roads simultaneously, while trying to reach a Member or when a Request source has to be notified of a message reception. As shown in Fig. 9(b), in TCAP a message source (SSREQ/SNREQ or Member) sends a single packet in at least two directions, increasing the chances of reliably delivering the message to its final destination. Even if this may introduce duplicate reception at destination, dedicated efficiency and dynamicity solutions can easily mitigate such effect (as discussed later in this section). As a result of the introduction of these additional paths, four additional intersections (RRs) were added to the phase of both creation and destruction of a TCA. These routers have similar behaviors as the main RRs, but will be referred to as "intermediate nodes" in the rest of the work.

C. Broadcast Exploitation

As already discussed, a TCAP RR might have to send a message in multiple directions at the same time, by exploiting the intrinsic nature of broadcast transmissions to efficiently deliver a single data packet to multiple nodes - guaranteeing that every RR would eventually receive the message. By specification, GPSR cannot exploit broadcast communications; we have extended this in TCAP to include the broadcasting opportunity when needed and by managing it at the upper layers of the protocol stack. In particular, TCAP exploits broadcasting in the following way:

1) when SSREQ, SNREQ, CN, or a Member have to send a message in more than one direction at the same time (e.g., see A.1.4A, A.1.5.0B, and A.1.6 in the annexed algorithm), a new broadcast packet is created, with a stored list of <zone, neighbor RR> pairs where the packet has to be initially forwarded. This will let every OBU determine whether it has to forward the packet and, eventually, towards which neighbor intersection;
2) the RR forwards the message to the network layer and the packet is finally dispatched;
3) at broadcast reception, each OBU, once determined if it is inside one of the designated zones, autonomously makes a decision about message forwarding to the associated RR (see Case A.0.0. of the annexed Algorithm II). In particular, the vehicle forwards the message if and only if there are no other OBUs closer to the destination that are also within a specific threshold, as detailed in the following pseudo code:

```
if (current vehicle is located in one of the requested zones)
    if (no other vehicles are closer to the destination)
        forward_the_packet();
    else if (∃ a vehicle closer to the destination which is also closer
than a given threshold to the source of the packet)
        drop_the_packet();
    else forward_the_packet();
else drop_the_packet();
```

Practically speaking, by adopting this application-layer broadcasting management solution, each OBU in a selected zone decides independently to forward the packet based on the presence of other vehicles closer to packet destination (which might have in turn received the same message). This reduces the broadcast-associated redundancy but might cause a node dropping a message in the case where the closest node available is too far away from the broadcast source. In such a case, an OBU can decide to drop the packet based on either the availability of closer vehicles or the distance of other nodes from the message source (shorter than a predefined threshold) or both. In other words, the level of redundancy generated by our TCAP broadcast is highly controllable because, by reducing this threshold, a higher number of vehicles might be induced to relay the packet. As further discussed in the related work section, this mechanism has been inspired by CBF [12] and LAOR [13]: however, differently from them and from the more traditional goal of network efficiency, the objective here is to focus on exploiting the redundancy (not necessarily reducing it) generated by a broadcast transmission, so as to increase the overall TCAP reliability.

D. Cross Layering

Due to our GPSR modifications, upon receiving a packet that has to be forwarded, a node is now forced to drop the message if no candidates are found. To implement this, the layered model has been redesigned to add a new public interface to the network layer, accessible only from the upper levels of the protocol stack. This allows our application-layer tracking accessing specific read-only data otherwise available only at the network layer. In addition, to let TCAP exploit such interface, GPSR forwards each received TCAP message upwards, by giving the protocol the possibility to control every single packet (as shown in the annexed algorithms for Case A). In this way, TCAP knows in advance when a next-hop is available or not, by simply accessing the same data used by GPSR to make routing decisions (with no additional costs), thereby providing for the case when a packet is going to be dropped by the network layer. In addition to this new characteristic, the TCAP cross-layer interface introduces the following features:

1) a RR can detect the unavailability of vehicles in a specific zone (direction) before dispatching a message;
2) as well as TCAP message forwarding, TCAP is capable of explicitly requesting the network layer to forward also GPSR hello messages upwards (as shown in the annexed algorithms - Case B). TCAP uses the cross-layer interface to set a Boolean variable (at the network layer), checked by GPSR at the reception of each hello packet, to eventually forward the message upwards to TCAP. This new modality takes the name of Listening Mode (LM) because of the capability that the upper layers have thus acquired to listen to routing messages between the node and its neighbors;
3) TCAP can extend its control over OBUs and consequently to every routing decision, thereby expanding the message exchange reliability to vehicle communications.

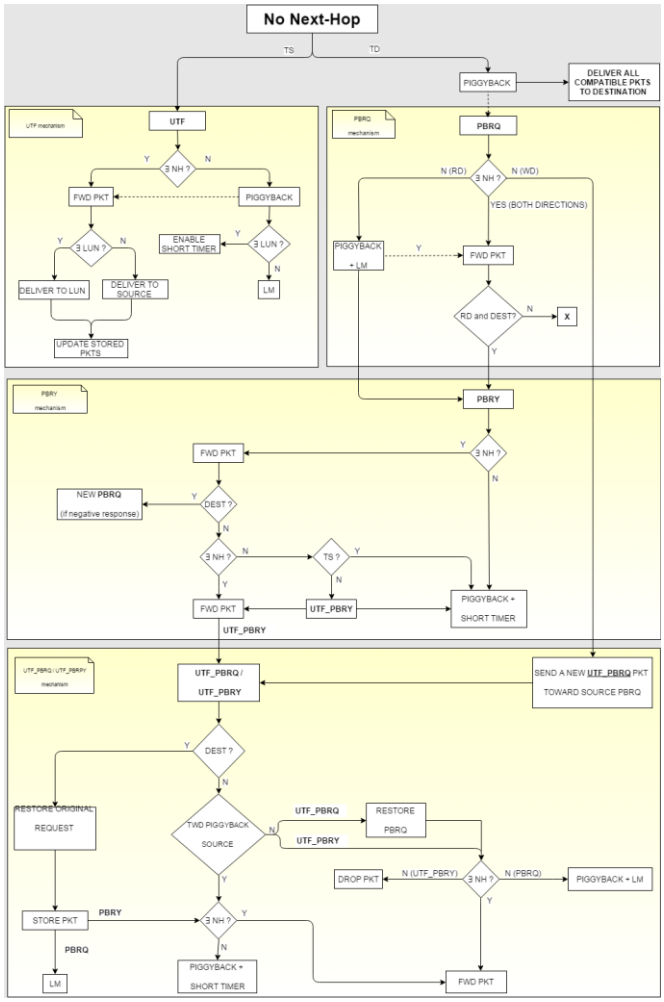


Fig. 10. TCAP protocol on OBUs when next-hop is not available.

E. TCAP OBUs

By restricting the modifications to a specific public interface implementing the cross-layer features (which limits the access to read-only information), a good level of decoupling is anyway maintained, thus simplifying any further development of the protocol.

Regardless of its current role, a RR needs to make use of intermediate nodes (i.e., vehicles equipped with OBUs) to reach one of its neighbor intersections. Algorithm II describes the behavior of an OBU upon receiving either a TCAP or a GPSR hello packet (Case A and B, respectively). Every vehicle, in addition to the modifications introduced for cross layering described earlier, has access to its own data about geographic coordinates (located either inside a zone or at an intersection) and driving direction (in terms of the intersection where a car is moving to). Fig. 10 shows a detailed view of

how TCAP works on vehicles to support critical case when the next-hop is not available.

An additional TCAP mechanism, called Piggyback in the following, has been introduced to prevent vehicles from dropping packets (see step A.0.0.0.0. of Algorithm II): the basic idea is that packets are carried when the car is moving towards the destination until either a new vehicle (which in turn can either piggyback again packets or forward them) is found or, in the worst case, the RR is reached. Whenever a vehicle can neither piggyback (driving in the opposite direction) nor forward a message, it is in charge of delivering a packet of type UNABLE_TO_FORWARD (UTF) to either a specific OBU (Last Useful Node – LUN) or the source of the request (RR), see Fig. 10 - UTF mechanism. Such node, defined in a dedicated field of the TCAP header, is the last hop (in the greedy forwarding mechanism) driving toward the final destination of the message. Upon creating a UTF packet, the original message is also stored for possible later restoring.

As Fig. 10 shows, OBUs have two options, based on the availability of a next-hop: i) forward the message or ii) piggyback it. In the latter case, either a short timer or the LM is enabled depending on whether LUN was previously set. Once forwarded, the UTF packet will be received by another vehicle (Case A.4 – step A.4.1-2): if this vehicle is driving towards the destination of the original request/notification, the OBU will restore the packet (LUN is also set) and look for a next-hop, which would bring the protocol back to the case when the message might be either forwarded or piggybacked. In the latter case, the following steps are executed, as depicted in Fig. 10 (packet is stored and LM enabled):

1) upon receiving a GPSR Hello message, every "pending" request/notification is checked: if the packet source is among the destinations of any stored message, each of them is sequentially dispatched and removed from the storage, or else, in the case it is moving towards the destination, a new PB_FWD_RQST packet is sent (having the current node as source and containing the original message);
2) at the reception of a PBRQ-type packet, the OBU checks the availability of a next hop (see PBRQ mechanism in Fig. 10): in case a node is found, the packet is forwarded. Then, if the same vehicle is also the destination of such request, a new PB_FWD_REPLY packet is created, to inform the source of PBRQ that the message has been successfully delivered to its destination (if a next-hop does not exist, the packet is stored and a short timer is set to attempt message delivery another time). Instead, if no vehicles are found, two different behaviors are executed depending on the direction of the current ego car: if it is moving toward the final destination, the OBU sets LUN, enables LM, stores the request, and then dispatches a new PBRY message (like in the case when a next hop is found, which is also PBRQ destination); otherwise, a new message of type UNABLE_TO_FORWARD_ PB_RQST is created, to inform the source of the piggyback request of the inability of forwarding such request (if a next-hop does not exist, the message is stored and a short timer is set to re-attempt message delivery);
3) at this stage, upon receiving a packet of type PBRY, an OBU behaves differently depending on whether the node is the destination (PBRQ source) or not (see PBRY in Fig. 10). In the former case, if PBRY is positive, the request is just removed, or else the packet is discarded and a new PBRQ is dispatched. In the latter case (step A.3.1), the availability of a next hop is checked: if a car is found, the message is just forwarded, otherwise, the OBU either stores the packet (and enables a short timer) or dispatches a new packet of type UNABLE_TO_FORWARD_ PB_REPLY (once again, if no vehicles are available, the message is stored and a short timer is enabled);
4) as the last case, an OBU might receive a UTF_PBRQ/UTF_PBRY message: if the vehicle coincides with the packet destination, the original request (independently from the type of message) is restored and saved. Furthermore, if the message is of type PBRQ, LM is enabled; otherwise, if the message is a PBRY one, it is either forwarded or piggybacked (a short timer is also scheduled), depending on the availability of a next hop towards the PBRY destination. Instead, if the vehicle is an intermediate node, the OBU checks whether it is moving toward the source of the packet indicated by the UTF message: in that case, the packet is either forwarded or piggybacked (a short timer is also scheduled), depending on the availability of a next hop; otherwise, if the OBU is moving in the opposite direction, two different behaviors might be assumed, based on the type of packet. When the message is UTF_PBRQ, the node is moving toward the original destination. Therefore, the original PBRQ is restored, LUN is set, and a PBRY is sent back to the request source. Then, similarly to when the packet is of UTF_PBRY type, the protocol tries to dispatch the message. If no cars are available, the packet is either piggybacked (LM enabled) or dropped, depending on whether the message is of type UTF_PBRQ or UTF_PBRY. It is important to remark that, if the original PBRQ source does not receive a positive reply, it will try to forward a new piggyback request as soon as it receives a GPSR Hello message from a compatible next-hop. Consequently, TCAP can forward a PBRQ message even if a packet has been either dropped or lost in the meantime.

To conclude the description of TCAP at OBUs, each piggybacked message of type PBRY, UTF_PBRQST, or UTF_PBRY (see Algorithm II - Step B.1) is forwarded as soon as a compatible next-hop is found, and the associated timer stopped. If no more stored packets are left, the listening mode is disabled (Step B.3).

F. Timers

Timers enable a RR to attempt either to resend a packet (at the reception of a UTF message) or to generate a new request to create/destroy a TCA, if all notifications have not been received on time. There are two main categories of timers: one used to try to dispatch a packet every N seconds up to K attempts and the other used to keep LM enabled for at most P seconds. The former is divided into two types, depending on

ACTION	SSREQ/SNREQ	MEMBERS	CENTRAL NODE	ADDITIONAL NODES	OBU
Forward a request	---	**TYPE 3**	**TYPE 2**	**TYPE 3**	---
Forward a notification	---	**TYPE 3**	**TYPE 3**	**TYPE 3**	---
Generate a notification	---	**TYPE 2**	---	---	---
Generate a request	**TYPE 1 & 2**	---	---	---	---
Forward **PBRPY** packet	---	---	---	---	**TYPE 2**

Fig. 11. Timers distribution in TCAP.

the attempt is to deliver a message either to one/multiple destinations or to one/multiple zones; they are identified in TCAP respectively as type1 and type2. The latter will prevent a RR from waiting before attempting to send a packet; this timer is referred to as type3. Fig. 11 summarizes the deployment of these timers in a TCAP RR.

In particular, type1 and type2 timers are used by both SSREQ and SNREQ; type2 is also employed when a CN has to forward a request or a Member has to generate a new reception notification. As reported in Algorithm I Case C.0, the type1 timer is used to send the same request with an increased packet number - a field in the TCAP header to distinguish between two identical requests sent at different moments. Once a request is dispatched, the timer is scheduled again, together with a type2 timer in the case that the RR detects that one or more zones are without a next-hop at the moment of broadcasting. Let us point out that, at timer expiration, the packet is sent only in the case a next-hop is available in at least one of the missing zones; otherwise, it is rescheduled until either the maximum number of attempts is reached or every zone has received the packet (no UTF message has been received from a zone where a next-hop has been found). Both timers are associated with the same maximum number of attempts (default=3) but different expiration times, depending on the dynamically determined factors described in the next section. Three times the interval between two consecutive expirations of type2 has to be smaller than a single interval of type1, so that the former will always fall within two consecutive occurrences of the latter. The end of the third and last interval of type1 coincides with the time constraint associated to the creation or destruction of a TCA, so that, if a notification has not been received from each member, this implies that TCAP could not complete the tracking and the target was lost. In addition, every time a RR has to forward a reception notification for which it is not the source, a type3 timer is used at UTF reception (Algorithm I – step A.7.0) or if no next-hop is available (e.g., Algorithm I - Step A.1.4B). This is also the case when a request has to be forwarded, with the CN exception. The type3 timer is the simplest one, only scheduled once to disable LM.

As part of the first category of timers earlier mentioned, the timer shown in algorithm II – Case C, is specifically used when there is the need for a new attempt to send a single packet. Its peculiarity consists in the fact that an interval between two consecutive expirations is much shorter than the other three timers. As the packet destination is another moving vehicle, its geographical coordinates (used also by GPSR to determine the next-hop) change during time and, so, proper interval setting is influenced by vehicle speed (or speed limits in the targeted urban environment).

G. TCAP Efficiency and Dynamicity

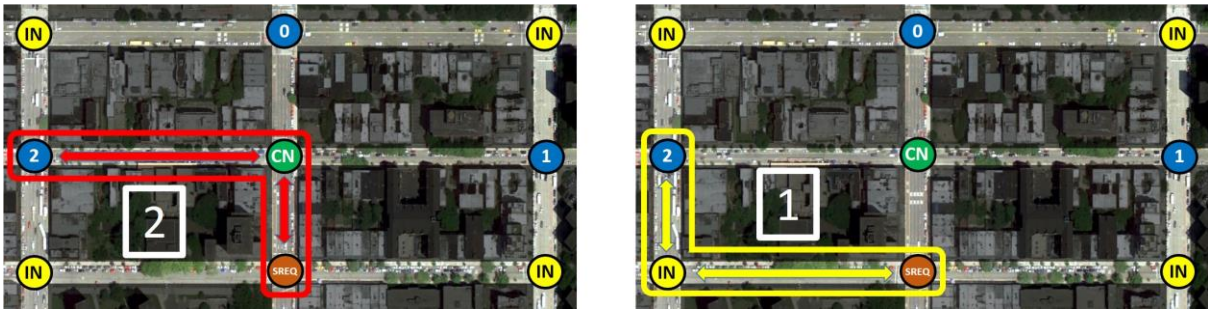
As the last addition to TCAP protocol features, we have introduced specific mechanisms to counterbalance the potentially negative effects of our reliability solutions and at the same time to improve overall speed and efficiency. These mechanisms could be split into two main groups: the first about redundancy reduction, the second about TCAP optimizations during TCA creation/destruction. About the former, we define a criterion based on which RRs and OBUs determine when a packet is a duplicate (and can be consequently dropped): every packet is stored only if either the timestamp of the request/notification is different or the same but with a higher packet number if compared with the one already stored. This simple mechanism makes every RR destination of an initial broadcast act as a sink, thus strongly reducing the potential negative effects of our broadcasting.

This mechanism also applies when an OBU has to piggyback a message (regardless of the type): similarly, it acts as a sink that ensures that the vehicle piggybacks only the original message. The latter takes advantage of the stored messages, in order to behave as follows:

1. (Algorithm I – step A.1.3) for any received request, RR controls if a reception notification has been previously stored (and eventually not received by SSREQ/SNREQ since a new request has arrived). There is no need to compare the request number; the check is based on packet content;
2. (Algorithm I – step A.1.3) if a match is found, the notification is set as “already sent” into the TCAP header (so that any following RR avoids doing the same) and the received packet is processed and forwarded to its destination;
3. (Algorithm I – step A.1.9) RR sends back the eventually matched notification of reception on behalf of the source of such notification. Once a Member has received a request, RR might also avoid sending it back through the intersection, which has already notified the reception on its behalf (see Algorithm I – Step A.1.5).

Together with the above mechanism, another optimization has been introduced to improve the TCAP performance when either a Member (1-2) or a CN receives a reception notification for a TCA for which it has not received any request. Specifically, both Members (1-2) and the CN look into their locally stored messages to see if they have already dispatched either a notification (Members) or a request (CN)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access



Original path (a) *Alternative path (a)*

Fig. 12. Dynamic forwarding of reception notification through alternative path in TCAP.

regarding the same TCA. Otherwise; they emulate the reception of a corresponding request, generating either a new notification of reception (Members – Algorithm I – Step A.4.1) or a new request (CN – Algorithm I – Step A.3.2).

In addition, we have also optimized TCAP at OBUs in the following way. For every received packet of type SREQ, NREQ, NOT_SREQ or NOT_NREQ:

1. (Algorithm II – Case A.0.3.0) if any piggybacked request exists, OBU checks if the received message is either a reception notification or a NREQ (and the corresponding request exists), by removing the one already stored. On the contrary, if the received message is a request and a corresponding packet is locally stored, the packet is attempted to be forwarded;
2. (Algorithm II – Case A.0.3.1) if any piggybacked UTF or reply exists, OBU checks if its source can be used to forward one or more of the stored packets. In that case, the vehicle first forwards every matched message and removes them from local storage (any associated timer is cancelled);
3. finally, if no stored packets are left, LM is disabled.

Consequently, an OBU can drop piggybacked requests that are likely to be discarded at their final destination, thereby working to increase TCAP efficiency and reliability at the OBU side. Let us note that packets are automatically discarded when the vehicle reaches an intersection, to avoid accumulating them over TCAP execution and because they will be out of scope once the car has moved into a different zone. Regarding RRs, every router has a timer dedicated to discard the locally stored packets that are older than a configurable threshold, again to avoid accumulating too many packets during target tracking.

Finally, an additional mechanism has been implemented to enable Member1 and Member2 dynamically choosing between two alternative directions depending on next-hop availability (at the moment of forwarding a reception notification generated by Member0). In particular, RR first forwards the notification to the zone between the current node and the closest intermediate RR to SSREQ/SNREQ (Fig.12 (a)); if a next-hop is unavailable, the other zone (between RR and CN) is then attempted (Fig. 12(b)); if again there are no cars available, the packet is stored and LM enabled (see Algorithm I – Step A.4.2-3).

V. Performance Evaluation

This section first describes our simulation testbed and, then, reports the TCAP performance results with realistic traffic patterns, by comparing the quantitative performance indicators achieved by enabling the different incremental modules and functionality previously described. In fact, also because there are no similar solutions already available in the literature that can be compared quantitatively with the TCAP vehicle tracking results, we have decided to report the results of the different incremental modules to highlight the effects of each introduced mechanism and technique, up to the finalized implementation of the full TCAP. As detailed in the following, our experiments show i) the suitability of our proposed target tracking technique in the challenging conditions of high density of vehicles, ii) the very weak dependency of TCAP performance from topology changes/constraints (e.g., street lengths and speed limits), iii) the TCAP capability of dynamically adapting to differentiated runtime conditions (e.g., the temporary lack of vehicles in a street).

As already stated, we identified 5 different and incremental TCAP modules, from basic features to the complete protocol. In the first, only the basic TCAP idea is included. In the second module, we have included all our original extensions to GPSR routing (e.g., safe mode). Then, in the third variant, our TCAP implementation considers increased reliability via the exploitation of our original mechanisms for TCA redundancy increase and broadcasting; moreover, type1 timers are employed (not the other timer types given that they relate to the capability of detecting next hops). In the fourth "evolution" stage, we have considered the TCAP variant with our cross-layering optimizations: these include active exploitation of TCAP at OBUs and the other two timer types. Finally, the full and finalized TCAP implementation is taken into consideration, with the introduction of our additional efficiency and dynamicity mechanisms.

We have conducted an extensive set of simulation experiments by using the widespread ns-2 [14] and realistic mobility traces corresponding to a Manhattan traffic pattern over a grid of 7x7 intersections and streets of the same length (see Fig. 13(a) for additional details about our simulation environment, specifically targeted to mimic urban behaviors in common downtown scenarios around the world, such as in Ottawa, which was of our central interest).

PARAMETER NAME	PARAMETER VALUE
Wireless medium	802.11p
Propagation model	Shadowing (beta = 2.7)
Data transmission rate	3Mbps
Transmission range (meters)	200 (originally 300m – 1km)
Received signal strength threshold (meters)	200 (originally 300m – 1km)
Vehicle's speed (meters/second)	[0..13.9] (13.9 m/s == (50 km/h)
Simulation time(seconds)	1000
Simulation area (meters^2)	1640 x 1640 = 2.689.600
Street area (meters^2)	250 x 20 = 5000
Intersection area (meters^2)	20 x 20 = 400
Number of RR	7 x 7 = 49
Beacon message interval (seconds)	2.0
GPSR beacon packet dispatch delay	0.005 - 0.015
GPSR TCAP packet dispatch delay	0.010 - 0.020
Number of attempts timer 1 & 2 (creation phase)	3
Number of attempts timer 1 (destruction phase)	4
Timers (1 – 2 – 3) (seconds)	12.0 – 3.0 – 12.0
Failure time creation (seconds)	36.0
Failure time destruction (seconds)	48.0

(a)

SPEED LIMIT (km/h)	Creation time constraint (s)	Destruction time constraint (s)
30	60	80
40	45	60
50	36	48
60	30	40
STREETS LENGTH (m)	Creation time constraint (s)	Destruction time constraint (s)
200	28.8	38.4
250	36	48
300	43.2	57.6

(b)

Fig. 13. Dynamic forwarding of reception notification through alternative path in TCAP.

The employed mobility traces were generated with the widely accepted SUMO simulator [15]. Each simulation run starts by including 2000 vehicles, for an average inter-vehicle distance of 12-13m and with an average speed limit of 13.9 m/s; the simulation duration is 1000s, with a warm up period of 280s for avoiding border effects, thus simplifying the full understanding of the reported results and how different TCAP modules affect performance indicators in different situations. Fig. 13(b) also reports the time constraints regarding the creation/destruction of each TCA during our simulations, which determine when TCAP fails at tracking the target vehicle for the whole duration of the simulation: such value is set equal to the time a target would take to drive, at the speed limit, between two intersections of a TCA (no decelerations, all semaphores with green light - worst case scenario). In our experiments, we use IEEE 802.11p with a data transmission rate of 3Mbps and 200m communication range; the received signal strength threshold is 200m; the propagation model is Shadowing, with β=2.7 and σ=6.0 (for outdoor shadowed urban scenarios), which in our experience and related literature is considered to provide a reasonably good modelling of practical scenarios of interest [16]. The resulting probability of successfully receiving a packet at 150m and 100m is respectively of 88% and 79%.

A. Simulation Results

To quantitatively evaluate TCAP suitability and efficiency, we defined the following metrics:

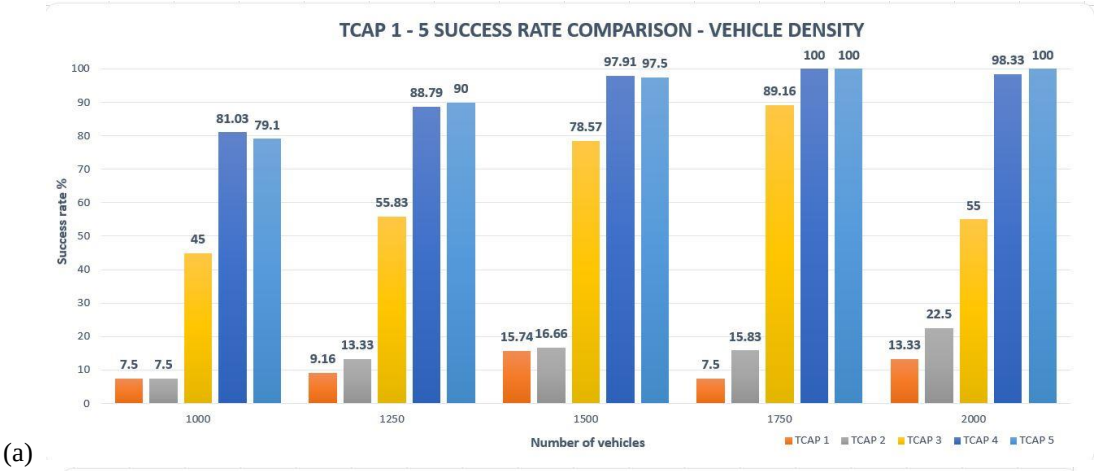
1) **Success Rate (SR)** indicates the success in tracking the target in terms of percentage of TCAs effectively created to successfully track the vehicle during the entire simulation and until the target leaves the simulation area (i.e., all necessary creation notifications from each member of both the new and the old TCAs have been correctly received);
2) **Creation/Destruction** time defines the time needed to create/destroy each single TCA. This metric gives additional insights and reinforces the results indicated by SR, because relatively high creation/destruction latencies show that TCAP, even if successful, was probabilistically close to an event of target loss;
3) **Overhead** measures the number of sent/received packets within a simulation. It is an indicator about the effective capability of our efficiency and dynamicity features to counterbalance the overall redundancy in TCAP;
4) **Packet Delivery Ratio** is directly correlated with the overhead metric and indicates the percentage of correctly received packets.

In our experiments we evaluated SR by varying vehicle density (between 1000 and 2000 vehicles in the simulation area), speed limit (between 30 and 60km/h) and street length between neighbor intersections (between 200 and 300m).

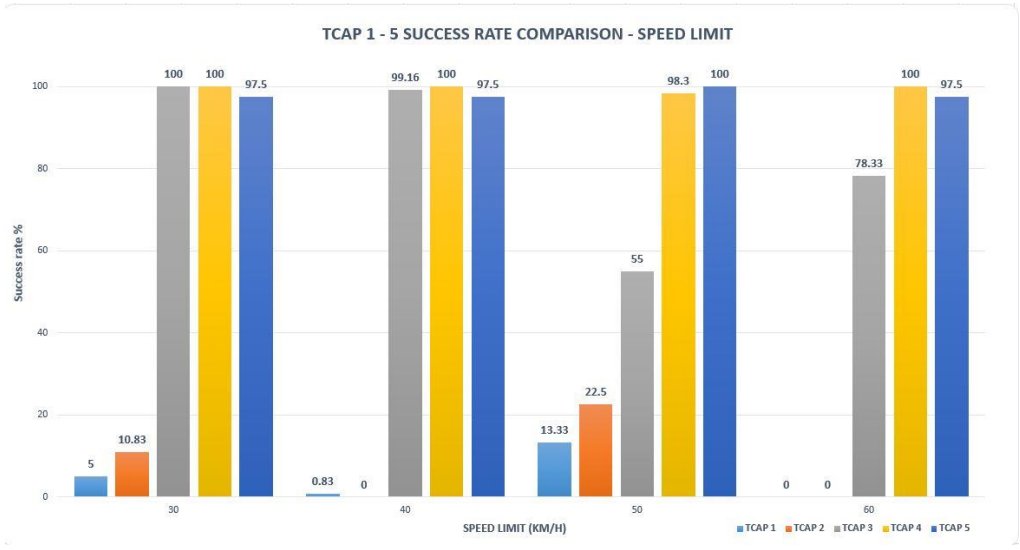
For what relates to the other three metrics, here, for the sake of briefness, we report simulation results only when varying vehicle density, which has demonstrated to be the most influential factor. Let us note that the different speeds and street lengths have influenced also the associated simulation time constraints, as summarized in Fig. 13(b). The initial tracking time is randomly generated after 280s and each simulation is concluded the moment the target is detected moving out of the simulated scenario (with a timeout equal to 600s). For any scenario, 30 simulation runs were executed, each of them with a different seed. The collected results are summarized in Fig. 14, Fig. 15, Fig. 16, and Fig. 17.

In particular, Fig. 14(a)-(c) show TCAP SRs (in the five incremental implementation steps, namely, TCAP1-TCAP5) for different vehicle densities (i.e., Fig. 14(a)), speed limits (i.e., Fig. 14(b)), and street lengths (i.e., Fig. 14(c)). TCAP5 achieves a promising 100% SR with highest densities (2000 and 1750), while decreasing down to 79.1% for 1000 deployed vehicles, with a standard deviation of 8.74% from the average, which demonstrated to be 93.81%. These figures show how TCAP can track a target for 80% of its trajectory even in the worst scenario, when the least number of vehicles is present. Moreover, we can see how, with high density, TCAP can successfully conclude all simulations until the number of cars decreases to 1500, where SR is 96.55%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access



(a)



(b)

(c)

Fig. 14. SRs for different (a) vehicle densities, (b) speed limits, and (c) street lengths.

If compared with the fourth completion step (TCAP4), where our efficiency and dynamicity features are not included, TCAP5 has obtained similar results on average, with a smaller success rate at the lowest density (79.16% comparing to 81.03% by TCAP4), motivated by the fact that TCAP4 is making use of a higher number of packets, as highlighted by the overhead metric. Similar considerations apply to the interpretation of the opposite side of the graph where, in this case, TCAP4 SR = 98.33% (compared to 100% of TCAP5) because of the higher number of packets and consequent lower delivery ratio (see also Fig.17). The figure also shows how the performance results significantly differ in the case of TCAP1

and TCAP2: the latter, in case of highest density, obtains only 22.5% for SR (7.62% standard deviation), while the former's SR is 13.33% (12.68% standard deviation).

This is even worse when the number of vehicles is low, with the lowest peak reached at 7.5% in case of 1000 participating vehicles. TCAP3, instead, shows how our redundancy solutions can increase the overall reliability with a deep impact on TCAP SR (89.16% with 1750 vehicles and 45% with 1000 cars), with a strong improvement (almost six times) if compared with TCAP2 (standard deviation is 20.28%, while it is 3.71% for TCAP1 and 5.44 for TCAP2). In addition, these indicators show how, with only our redundancy support, TCPA suffers when working in both high-density and very-low-density scenarios, due to respectively high duplication rate and scarce dynamic reactivity, especially on OBUs.

Fig. 15(a)-(c), instead, highlight a further comparison on the number of simulations successfully concluded for different numbers of vehicles (Fig. 15(a)), speed limits (Fig. 15(b)), and street lengths (Fig. 15(c)). Here, we focus only on TCAP3, TCAP4, and TCAP5 because they have shown to achieve better performance. In the case of 1000 vehicles, TCAP4 and TCAP5 have the same completion success rate, i.e., 17; the complete TCAP implementation, i.e., TCAP5, has always obtained better figures (with the widest difference reached for 1500 vehicles, with 23 and 27 successful runs each).

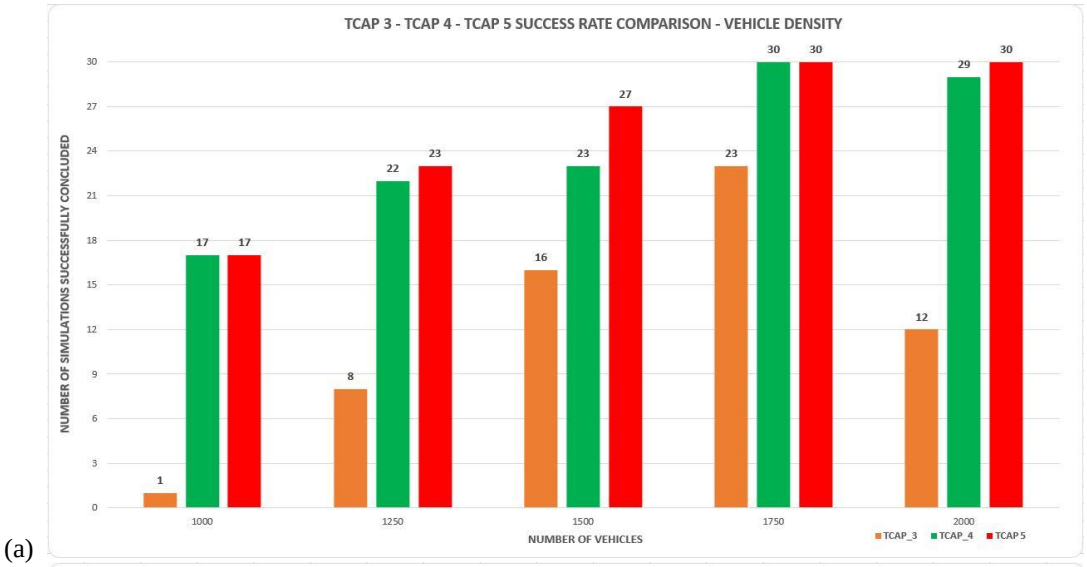
This graph also shows how the introduction of cross-layering optimizations has relevantly improved the TCAP ability to track a target in all cases, especially when the number of cars is scarce. In fact, TCAP3 can conclude only one simulation successfully with 1000 vehicles and 8 in case of 1250 vehicles, comparing to respectively 17 and 22/23 in the case of TCAP4/TCAP5. Fig. 14(b) and Fig. 15(b) present SR and the number of simulations successfully concluded with different speed limits respectively. On the one hand, we can see how TCAP5 performance does not depend on vehicle speed; also SR standard deviation over different speed limits equals to a small value of 1.25%, in line with our expectations. Fig. 14(b) also shows how TCAP4 can obtain better performance with lower speeds (100% for 30km/h while 97.5% for 40km/h) thanks to the usage of a larger number of packets, as we will comment in the following (overhead results). In addition, the same charts highlight how TCAP1 and TCAP2 behave similarly, with the performance dropping to around 0% when the speed limit is 40 or 60km/h. Instead, TCAP3 shows a high dependency on low speed limits, as can be further seen in Fig. 15(b), where the number of simulations successfully concluded are 30 for 30km/h, almost three times the ones at 50km/h. This is another proof of how our TCAP features, all together, can make the protocol independent from specific speed limits (and also from street lengths, see below).

Fig. 14(c) and Fig. 15(c) show results for SR and for the number of successfully concluded simulations with different street lengths. Again, TCAP5 has shown that it does not strongly depend on deployment topologies: in the worst case of Fig. 14(c), TCAP has obtained a remarkable SR = 95% when the distance between two neighbor intersections is 200m, in line with the results reported in Fig. 14(b); at that distance, 26 out of 30 simulations have been successfully concluded (Fig. 15(c)), which is in turn the same value obtained at 300m, where the measured SR is equal to 95.68%. Regarding TCAP4, once again, that protocol version presents better behavior when the distance between neighbor intersections is 200m, with SR = 98.33%, mainly due to high packet duplication. As expected, TCAP1 and TCAP2 show results in line with those obtained with other vehicle densities and speed limits, even though TCAP2 shows a better behavior when distance changes; this is motivated by the fact that shorter distances increase the GPSR chances to successfully delivery a packet, thanks also to our original modifications (e.g., safe mode).

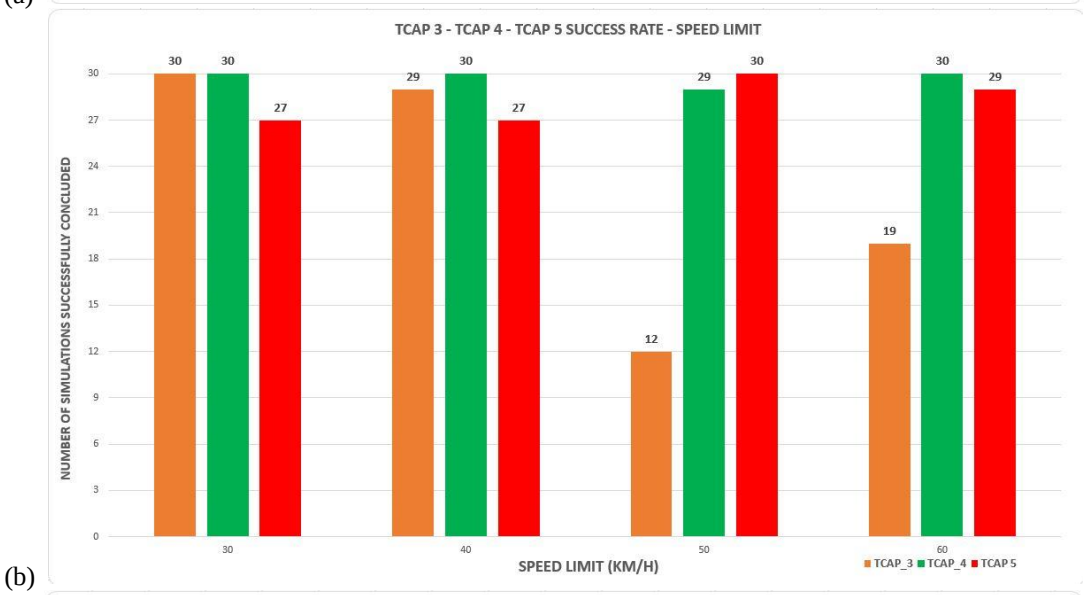
We also studied creation and destruction time of TCAP. In particular, Fig. 16(a) and (b) report obtained results for, respectively, the average creation and destruction time while changing vehicle density and with regards to the five implementation steps of our protocol. Let us recall that each TCA creation and destruction is defined as failed if SSREQ/SNREQ does not receive all the expected notifications within respectively 36 and 48s from the time of first request. In the case of failure, we have decided to calculate the average creation/destruction time by imposing a "indicator handicap" for each failure, by adding the worst possible time interval for each failure. In this way, we can anyway quantitatively compare all the cases, i.e., when all TCAs are successfully created/destroyed and when not all SSREQ/SNREQ received the expected notifications on time. The figure shows that, in the TCAP5 case, the average time needed to create/destroy a TCA is higher when the number of cars decreases, passing from 3.44/6.13s at 2000 to 15.05/14.30s at 1000. This behavior is expected because, with less dense vehicular scenarios, there is more often the need to either piggyback a message or to use timers, generated by the more probable chances of not finding a next-hop towards packet destination. The same effect is observed for TCAP4, with the peculiarity that the average time to create a TCA at low density is lower than for TCAP5; standard deviations of these creation/destruction results for TCAP4 and TCAP5 are 4.53/6.44s and 5.86/4.48s.

The other three less complete implementation steps, i.e., TCAP1, TCAP2, and TCAP3, have demonstrated considerably worse indicators. Starting from the first two, Fig. 16(b) highlights the inability of the protocol to destroy even one TCA on time. A similar behavior is shown in Fig. 16(a), where TCAP2 demonstrates to be unable to conclude even a single simulation successfully (best average at 2000 and equal to 23.85s). In addition, the lower number of vehicles shows how our GPSR modifications are scarcely usable if not supported by our original redundancy+cross-layering. We have experienced a slight improvement with TCAP3, even though, at the highest density (Fig. 16(b)), the protocol still takes more than five times the time needed in TCAP4 to destroy a TCA (29.74s against 5.08s).

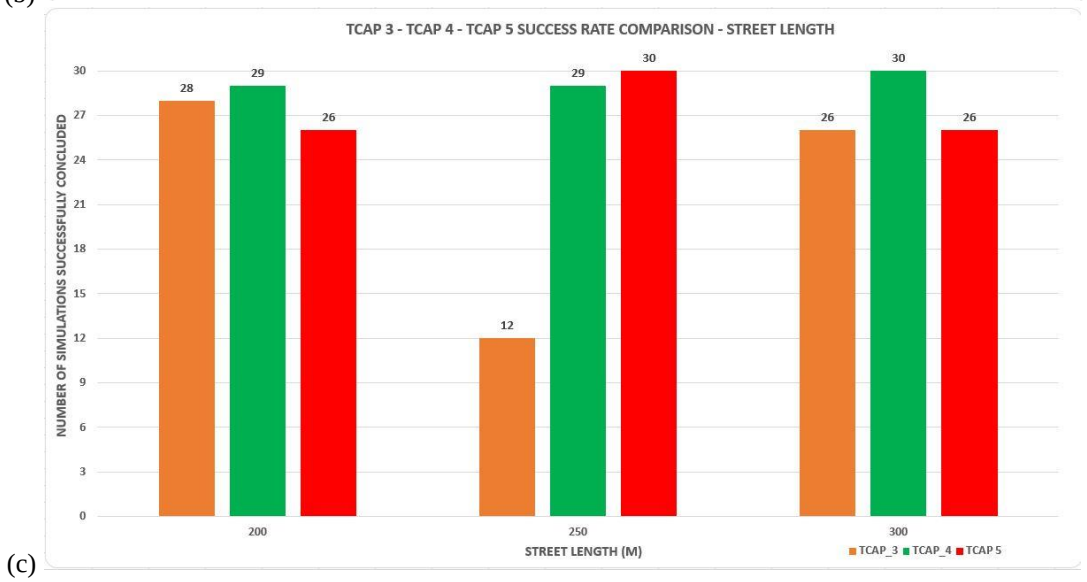
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access



(a)



(b)



(c)

Fig. 15. Number of simulations successfully concluded for different (a) vehicle densities, (b) speed limits, and (c) street lengths.

Finally, the last two charts in Fig. 17(a) and (b) respectively report the average number of sent/received packets and the packet delivery ratio while executing TCAP3, TCAP4, and TCAP5. In particular, Fig. 17(a) highlights how TCAP5 uses a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2678107, IEEE Access

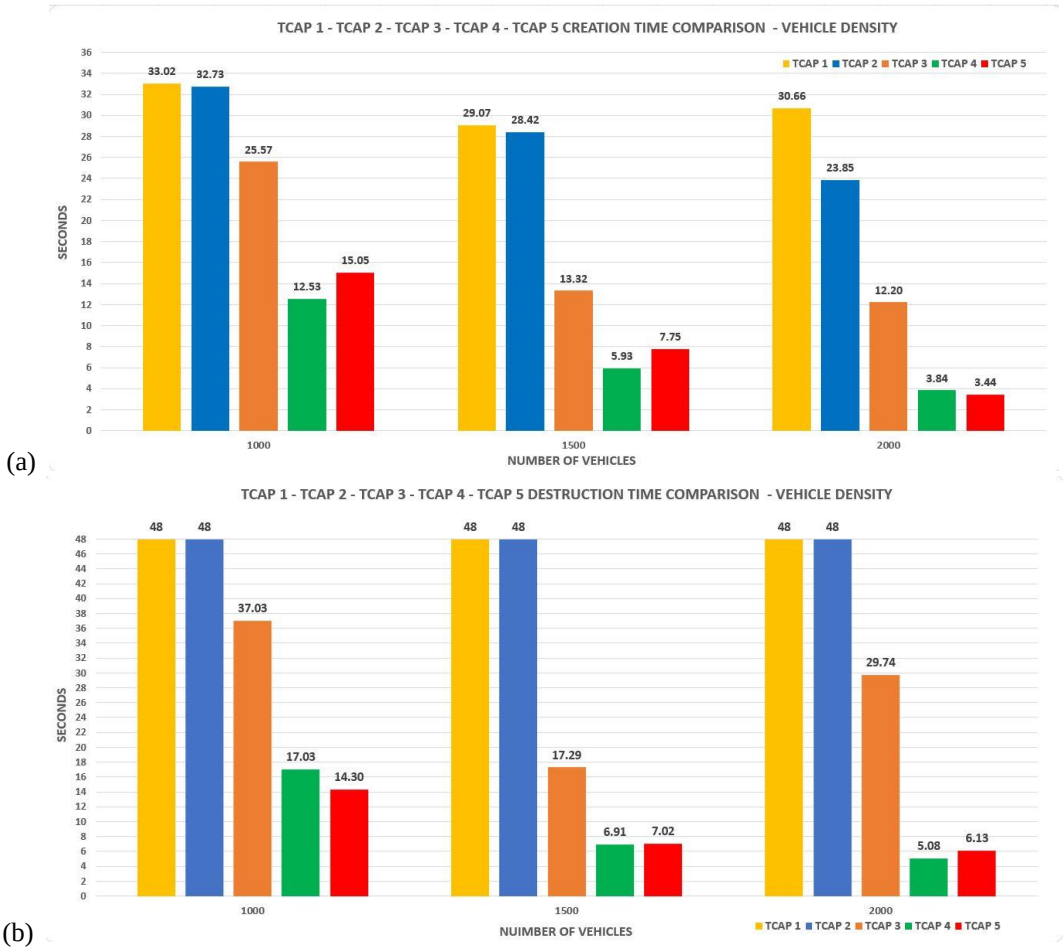


Fig. 16. (a) Creation and (b) destruction times for different vehicle densities.

significantly smaller number of packets if compared with the two previous TCAP evolution steps. In other words, with the introduction of our efficiency and dynamicity solutions, TCAP can counterbalance the effects given by the adoption of our redundancy mechanisms, without affecting the overall protocol performance, as demonstrated by comparing Fig. 14 and 15 with Fig. 16(a)-(b). On average, TCAP5 has sent/received respectively 5366 and 4667 packets with the highest density of vehicles; TCAP4, instead, with the same density, has sent/received respectively 28029 and 22211 packets; these last numbers are still considerably higher if compared with TCAP3, where our cross-layering mechanisms are not introduced yet. For example, with 1000 vehicles, TCAP3 has sent/received respectively 20630/16201 packets, which is around six thousand messages less than TCAP4. This proves, once again, how cross-layering is crucial for the development of our protocol, mostly due to the traffic generated at OBUs, which has a deep impact on the overall TCAP performance. Nonetheless, TCAP3 also shows how, without the mechanisms introduced in the further evolution steps, the overhead severely depends on the number of vehicles.

Moreover, Fig. 17(a) also shows how, in the case of lowest density (1000), the number of received packets in TCAP5 is higher than the sent ones (4500 and 3822 respectively); this gives also a measure of the occurrence of broadcasting situations. In the previous implementation steps, the broadcasting mechanism was less observable because the degree of redundancy of our protocol was still high (e.g., TCAP4 sends/receives respectively 26603/22388 packets); thanks to the introduction of our efficiency and dynamicity mechanisms, this effect is clearly visible in this case. The effect can be also observed when analyzing Fig. 17(b) where, with 1000 vehicles moving in the scenario, the percentage of effectively delivered TCAP5 packets becomes 100%. Nonetheless, the delivery ratio anyway decreases to 92.42 % in case of 1500 cars and to 85.92 % at 2000: these figures are direct consequence of the simultaneous presence of a considerably high number of cars in the deployment environment, which increases a lot the probability of collision. This also motivates why the TCAP3 delivery ratio is almost always higher than the TCAP4 one.

As a final summarizing consideration, let us note that, by introducing (step by step) each different additional feature to our protocol, TCAP has gradually obtained better results, until satisfying all our performance requirements in the complete TCAP5 version. In particular, without the introduction of multiple paths, the protocol was not able to sufficiently cover TCA creations/destructions; our cross-layering mechanisms have proved to be of fundamental importance when high

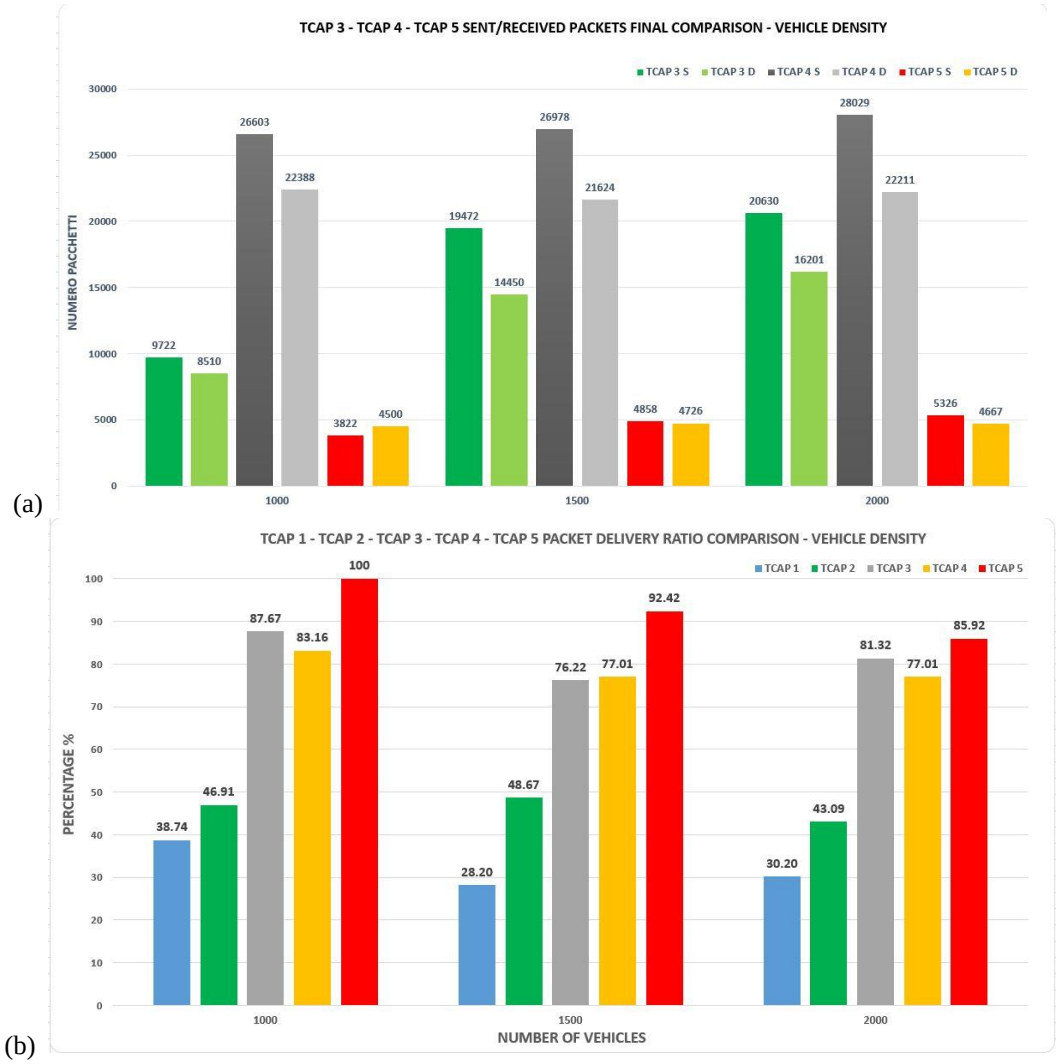


Fig. 17. (a) Average number of sent/received packets and (b) packet delivery ration for different vehicle densities.

success rate has to be obtained, as in many mission-critical scenarios; whenever the number of exchanged packets and the TCAP communication overhead can be considered not crucial, we believe that also TCAP4 has some practical opportunities to be exploited.

VI. Related Work

Here we first list the primary state-of-the-art target tracking solutions for VANets, followed by an overview of the actual coverage technologies in WSNs and a focus on the Trap Coverage [8] technique (at the basis of our tracking approach). Then, we rapidly discuss how VANet routing solutions can be categorized by introducing the three protocols related to our TCAP routing (GPSR [10], GPCR [17], and TO-GO [18]). The section is concluded with a focus on opportunistic routing and, in particular, CBF [12] and LAOR [13], which partially inspired some elements of our TCAP work.

Only few works on VANet target tracking have been presented in the literature so far. In [4], Ramos et al. address the main aspects and challenges for the cooperative tracking of one or multiple targets in a VANet scenario. They argue that cooperative target tracking solutions might augment driver's perception of the surrounding context and increase the comfort/safety of driving experience; they defined the main components at the base of any tracking system, i.e., a motion model, the measurement of target state, a data association algorithm, and a filtering component (e.g., the Kalman filter [19]), from which target tracking is thus defined as a sequence of (1) prediction of future target position and (2) correction of this estimation based on measurements. The closest work to our TCAP proposal is presented in [6]: Reza et al. define a system to trace an on the run vehicle by means of cameras mounted on cars in a metropolitan area (vehicles are identified by their LPNs and supposed to follow common traffic regulations). The authors described their tracking technique as consisting of three steps, i.e., (1) vehicle localization, (2) tracking data dissemination to neighbor OBUs/RSUs, and (3) future target position prediction in order to let RSUs broadcast the tracking request in the area where the vehicle is expected to be found. However, this tracking solution exhibits several non-negligible issues: first, the system relies on the strong assumption of the continuous availability of a reliable

backbone of intercommunicating RSUs, which makes the proposal hardly scalable and not applicable in several real scenarios; second, it assumes the presence of vehicles whenever needed and no alternative scenarios are considered (e.g., very limited traffic during the night); third, in predicting future location, a target is assumed to move at the average speed provided by a Traffic Information Center (TIC). In [7], instead, Reza et al. present a variant to the proposal in [6] in order to improve the movement estimation model, as they argued that, in [6], a vehicle at an intersection has always to choose between three possible zones and four directions (the choice sets are uniform in each trial). In particular, they based the model on condition logics to let a choice depend on the characteristics of each single alternative. However, this work still suffers from the above issues of [6], although it partly succeeded on improving the overall prediction model.

Coverage is one primary research interest in the WSN area [20]. Its main goal is to provide the best deployment setting given the available sensors, given that they often might be insufficient to cover the whole area of interest. As pointed out in [21], coverage usually involves two main challenging issues: (1) how to assess coverage performance and (2) how to improve such performance when the set of available sensors cannot satisfy the application requirements. However, [21] argues that several some open issues remain to be adequately addressed. For example, several solutions approximate communication ranges and sensing areas as perfect circles, without considering obstacles, collisions, sensor mobility, and other possible sources of communication faults. When designing TCAP, we based our approach on the seminal idea introduced by Balister et al. in [8], by the name of Trap Coverage, which generalizes the full coverage model by allowing holes (i.e., a set of uncovered points of a target region) of a given maximum diameter. In particular, such technique guarantees that any moving object can move at most a known displacement before being detected by the sensor network, for any trajectory and speed. This coverage model is demonstrating to scale well with large deployment regions, instead of focusing on a blanket coverage [9] (which might restrain real-life applications), thanks also to the fact that sensor density can be arranged depending on desired tracking quality, by enabling economizations on the number of sensors needed. In their work, Balister et al. also made the following contributions: (1) they provide a reliable estimation of the density needed to achieve trap coverage with a desired diameter in case of random deployments of the sensors; (2) they show how their introduced model explains the long existed gap between percolation threshold and critical density to achieve full coverage; (3) they provide an algorithm (with polynomial complexity) to determine the level of trap coverage achieved after an arbitrary sensor deployment.

Since the advent of VANets, a large number of routing solutions has been proposed in the literature (the interested reader can find good taxonomies in [22-24]). In [22], Lee et al. distinguish between topology-based routing, which exploits information about existing network links, and geographic routing where neighbor locations are used. The authors argued that geographic solutions have higher chance of practical usability in real-life applications, as topology-based protocols suffer more from route breaks in highly dynamic environments like VANets. In addition, geographic solutions can be further classified into none delay tolerant (NONE-DTN), delay tolerant (DTN), or a hybrid of these two. A good example of NONE-DTN protocols is GPSR [10], a simple and lightweight min-delay protocol, well suited for highly dynamic scenarios, or GPSR+AGF [25], or CBF [12], which does not make use of any beaconing support.

As highlighted in [11], when used in urban scenarios, GPSR suffers from some critical issues. Since the planarization methods assume a free space model, obstacles might lead to network disconnection, and consequently loss of packets. Moreover, the resulting graphs, along streets, cause vehicles not to send packets to the node closest to destination, with a consequent higher number of intermediary nodes and longer latencies. Routing loops might indeed frequently happen, due to high dynamicity of VANet topologies. Finally, the right-hand rule might induce GPSR not to consider possibly faster alternative paths.

Topology-assist Geo-Opportunistic Routing (TO-GO) [18] is instead a hybrid routing protocol where the topology knowledge, acquired via 2-hop beaconing, is used to find the best forwarding target: this target is either the furthest node toward the destination or a junction node, located at an intersection, from where a packet can be forwarded in any direction (to specifically fit city topologies). Opportunistic routing is then exploited for message forwarding to minimize the negative effects related to hidden terminal issues. For beacon-assisted protocols, the authors in [22] distinguish between overlay routing, when a protocol operates on a set of representative nodes overlaid on top of the existing network, and non-overlay routing. A good example of overlay solutions are Greedy Perimeter Coordinator Routing (GPCR) [17] and GPSRJ+ [26]. GPCR is based on GPSR and greedily forwards messages (even in perimeter mode) to the next junction, where a node called coordinator is uniquely responsible for taking routing decisions. GPCR demonstrates to forward messages faster than GPSR, but at the expense of global knowledge of the city topology. Lochert et al. propose two alternative (local) approaches to decide if a node is a coordinator, both based on neighbors' position. The first one exploits additional beaconing mechanisms, while the second use a correlation coefficient (1 in a street, 0 at a junction) to induce different behaviors. The intuition that a city forms a natural planar graph, together with the idea of always greedily forwarding a packet inside a street, have influenced our proposed routing architecture and, in particular, the role an intersection plays in routing decisions in TCAP.

Opportunistic Routing (OR) is widely exploited nowadays in different kinds of wireless networks to increase transmission reliability and network throughput by taking advantage of the broadcast nature of the employed wireless medium [27, 33]. In OR solutions routing decisions are taken at each broadcast receiver and, since more than one packet copy may be relayed, one of the primary OR objectives is to

guarantee that the minimum number of copies (ideally one) is eventually forwarded to destination. In [28], Cabrera et al. present a good overview of OR common and basic operations. (1) Candidate Selection (CS) works at the beginning to select the set of candidate nodes for packet forwarding (a good classification based on CS determination metrics is found in [29]). GeRaf forwarding in [30] is a good example where each receiver decides independently whether being part of the CS or not. (2) Candidate Priority Assignment works after CS to assigns a priority to each forwarding candidate (see [31] for a good classification of priority metrics, generally based on local network information). (3) Then Candidate Coordination determines whether or not a candidate should forward the received packet; [31] provides a good classification of these solutions depending on the availability (or not) of control packets, used to inform the other candidates about packet reception (e.g., ExOR [32]). Among the wide spectrum of OR solutions in the literature, two have someway influenced the design of our TCAP: Contention-based Forwarding (CBF) [12] and Location-Aided Opportunistic Routing protocol (LAOR) [13]. In the former, Füßler et al. propose a greedy forwarding strategy for position-based routing algorithms where, at the moment of sending the initial broadcast in a forwarding area, the source includes both its actual position and the final destination in the transmitted packet; in this way, each receiver can independently decide being part or not of the CS by following a greedy forwarding approach. Upon receiving a packet, each candidate competes to obtain the right to forward it. Timer-based coordination is employed, together with a suppression mechanism to stop other nodes from uselessly relaying the same packet. In LAOR, instead, a CS is selected in advance by the broadcast source; hello messages are used, together with a Location Service, to let nodes access the destination location information; the broadcast source determines CS based on distance from destination; the same distance is used also for priority determination. Each receiver in the CS list sets a forwarding timer, proportional to its priority, to relay the message at its expiration if no node with higher priority has sent the packet in the meantime. LAOR considers three conditions to determine the forwarding set: i) the candidate distance to destination must be smaller than that of the source; ii) the candidate distance to a line segment L (source and destination are L's endpoints) must be within a given threshold (which prevents routes from diverging); iii) the distance between any pair of nodes of the forwarding set must be within another threshold (which ensures that candidates can hear each other, thus limiting duplicates). If no candidates receive the broadcast, LAOR uses a hop-by-hop network layer ACK mechanism, to let the source attempt to retransmit the message up to three times. Even if we claim that the need of a Location Service is a strong assumption, not always realistically feasible in practical deployment scenarios, we found relevant inspirations from LAOR in terms of CS creation and prioritization (see TCAP5).

VII. Conclusive Remarks and Directions of Future Work

Several recent research activities in the VANET area have focused on the benefits and open technical challenges stemming from direct vehicle-to-vehicle interaction to achieve cooperative goals in different domains. To the best of our knowledge, our TCAP solution is original in addressing the relevant and recognized vehicle target tracking problem by i) exploiting the concept of trap covering area, ii) introducing novel algorithms and protocols to maximize scalability while maintaining very high reliability, iii) being capable of dynamically exploiting existing cameras at some intersections and cooperative vehicles freely moving among them, and iv) requiring very limited infrastructure support.

TCAP has demonstrated to be able to achieve the challenging performance indicators needed for target tracking in urban scenarios, with a limited and scalable overhead. It outperforms existing general-purpose solutions for packet routing in VANets and provides a valuable starting basis for the development of industrially-usable solutions for urban tracking at sustainable costs. In addition, the paper provides a significant contribution to the literature in the field by quantitatively showing how the different mechanisms, progressively introduced in the different TCAP variants, affect reliability, communication overhead, latency, and redundancy in these deployment environments.

The encouraging results achieved so far are motivating us to go on with related research work. In particular, on the one hand, we are now collecting in-the-field experimental results from a set of TCAP-enabled cars in order to assess our simulation results. On the other hand, to validate the general applicability of the approach, we are significantly extending the simulation work by considering real traffic mobility traces collected in the city of Bologna, also thanks to the work accomplished within the framework of the EU FP7 COLOMBO project (www.colombo-fp7.eu/), and by considering less regular road topologies, which are more usual in historical EU cities such as Bologna.

References

[1] G. Dimitrakopoulos and P. Demestichas, "Intelligent Trasportation Systems," *IEEE Vehicular Technology Magazine*, pp. 77-84, March 2010.

[2] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin and A. Hassan, "Vehicular ad hoc networks (VANETS): status, results, and challenges," *Telecommunication Systems,* no. 50, pp. 217-241, 09 December 2010.

[3] Y. Wang and F. Li, "Vehicular Ad Hoc Networks," in *Guide to Wireless Ad Hoc Networks*, 2009, pp. 503-525.

[4] H. Ramos, A. Boukerche, R. Pazzi and A. Frey, "Coopeartive target tracking in vehicular sensor networks," *Wireless Communications, IEEE,* vol. 19, no. 5, pp. 66-73, October 2012.

[5] R. Brooks, P. Ramanathan and A. Sayeed, "Distributed target classification and tracking in sensor networks,"

Proceedings of the IEEE, vol. 91, no. 8, pp. 1163 - 1171, August 2003.

[6] T. Reeza, M. Barbeau and B. Alsubaihi, "Tracking an on the run vehicle in a metropolitan VANET," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, Gold Coast, 2013.

[7] T. Reeza, M. Barbeau, G. Lamothe and B. Alsubaihi, "Non-cooperating vehicle tracking in VANETs using the conditional logit model," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, The Hague, 2013.

[8] P. Balister, Z. Zizhan, S. Kumar and P. Sinha, "Trap Coverage: Allowing Coverage Holes of Bounded Diameter in Wireless Sensor Networks," in *INFOCOM 2009, IEEE*, Rio de Janeiro, 2009.

[9] S. Kumar, L. Ten H. and B. Jozef, "On k-coverage in a mostly sleeping sensor network," in *MobiCom '04 Proceedings of the 10th annual international conference on Mobile computing and networking* , New York, 2004.

[10] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MobiCom '00 Proceedings of the 6th annual international conference on Mobile computing and networking* , 2000.

[11] C. Lochert, H. Hartenstein, J. Tian and H. Fussler, "A routing Strategy for Vehicular Ad Hoc Networks in City Environments," in *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, 2003.

[12] H. Füßler, J. Widmer, M. Kasemann, M. Mauve and H. Hartenstein, "Contention-based forwarding for mobile ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 351-369, 2003.

[13] Y. Xudong , Y. Jiangtao and Y. Sunzheng, "Location-Aided Opportunistic Routing for Mobile Ad Hoc Networks," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, Beijing, 2009.

[14] K. Fall and K. Varadhan, "ns notes and documents," The VINT Project, a collaboration between UC Berkeley LBL, USC/ISI and Xerox PARC. Available at: http://www.isi.edu/nsnam/ns/ns-documentation.html, 2000.

[15] D. Krajzewicz, J. Erdmann, M. Behrisch and L. Bieker, "Recent Development and Applications of SUMO," *Int. J. Advances in Systems and Measurements*, vol. 5, no. 3, pp. 128-138, 2012.

[16] T. Abbas, K. Sjöberg, J. Karedal, and F. Tufvesson, "A Measurement Based Shadow Fading Model for Vehicle-to-Vehicle Network Simulations", *Int. Journal of Antennas and Propagation*, vol. 2015, 2015.

[17] C. Lochert, M. Mauve, H. Füßler and H. Hartenstein, "Geographic Routing in City Scenarios," *ACM SIGMOBILE Mobile Computing and Communications Review* , vol. 9, no. 1, pp. 69-72, January 2005.

[18] K. Lee, L. Uichin and M. Gerla, "TO-GO: TOpology-assist geo-opportunistic routing in urban vehicular grids," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on* , Snowbird, UT , 2009.

[19] G. Welch and G. Bishop, "An introduction to the Kalman filter," University of North Carolina at Chapel Hill, 2006.

[20] Z. Chuan, Z. Chunlin, S. Lei and H. Guangjje, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 619-632, Marc 2012.

[21] F. Gaojun and J. Shiyao, "Coverage Problem in Wireless Sensor Network: A Survey," *Journal of Networks*, vol. 5, no. 9, pp. 1033-1040, September 2010.

[22] K. C. Lee, U. Lee and M. Gerla, "Survey of routing protocols in vehicular ad hoc networks," in *Advances in Vehicular Ad-hoc Networks: Developments and Challanges*, IGI Global, 2010, pp. 149-170.

[23] L. Fan and W. Yu, "Routing in vehicular ad hoc networks: A survey," *Vehicular Technology Magazine, IEEE*, vol. 2, no. 2, pp. 12-22, 2007.

[24] L. Yun-Wei, C. Yuh-Shyan and L. Sing-Ling, "Routing Protocols in Vehicular Ad Hoc Networks: A Survey and Future Perspectives," *J. Information Science and Engineering*, vol. 26, pp. 913-932, 2010.

[25] V. Naumov, R. Baumann and T. Gross, "An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces," in *MobiHoc '06 Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing* , 2006.

[26] K. Lee, J. Haerri, L. Uichin and M. Gerla, "Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios," in *Globecom Workshops, 2007 IEEE* , Washington, DC , 2007.

[27] A. Darehshoorzadeh, . L. Cerdà-Alabern and V. Pla, "Opportunistic routing in wireless mesh networks," in *Routing in Opportunistic Networks*, Springer New York, 2013, pp. 289-330.

[28] A. Triviño-Cabrera and S. Cañadas-Hurtado, "Survey on Opportunistic Routing in Multihop Wireless Networks," *International Journal of Communication Networks and Information Security*, vol. 3, no. 2, pp. 170-177, 2011.

[29] L. Pelusi, A. Passerella and M. Conti, "Opportunistic Routing in Wireless Networks: Methods, Models and Classifications," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 134-141, Movember 2006.

[30] M. Zorzi and R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Network: Multihop Performance," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 4, pp. 337-348, 2003.

[31] A. Boukerche and A. Darehshoorzadeh, "Opportunistic routing in Wireless Network: Models, Algorithms and Classifications," *ACM Computing Survey*, 2014.

[32] S. Biswas and R. Morris, "ExOR: opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM*, 2005.

[33] L. Haitao , Z. Baoxian , H. Mouftah and X. Shen, "Opportunistic routing for wireless ad hoc and sensor networks: Present and future directions," *Communications Magazine, IEEE* , vol. 47, no. 12, pp. 103-109, 2009.