Architecture of a Communication Middleware for VANET Applications

Luca Caviglione[1], Giuseppe Ciaccio[2], and Vittoria Gianuzzi[2]

[1]ISSIA, U.o.s. Genova, Consiglio Nazionale delle Ricerche. Via de Marini 6, I-16149 Genova, Italy
[2]DISI, Università di Genova. Via Dodecaneso 35, I-16146 Genova, Italy

***Abstract*—As one of the main tasks of a project for developing an Advanced Cooperative Infomobility System (ACIS), we have devised and partly implemented a communication middleware for infomobility applications on vehicular ad-hoc networks (VANETs). In this paper we illustrate and motivate some of the architectural choices underlying the ACIS middleware.**

I. Introduction

A Vehicular Ad-Hoc Networks (VANET) is a kind of ad-hoc network where the networking/computing devices are mostly installed on cars, with possibly a small number of them installed at fixed locations called Road Side Units (RSU) that improve connectivity by acting as hubs. Designing the communication middleware to be deployed within a VANET was one of the main tasks in an industrial project for developing an Advanced Cooperative Infomobility System (ACIS) where we have been involved as academic partners. One of the main goals of the project was to turn each moving car into a sort of mobile sensor, able to observe and report about various phenomena, possibly but not exclusively related to traffic. The VANET may also play additional roles; for instance, it may allow information to flow back from the data center to the cars (e.g. delivery of traffic alerts or route prescriptions to individual users or groups) as well as from car to car (e.g. direct dissemination of traffic data, forward collision warnings, coordination or entertainment within platoons of vehicles). In this paper we report about the architecture of the ACIS communication middleware, called ACME (ACis Middleware for Emerging vanet applications), which is broadly based on the publish-subscribe and the peer-to-peer (p2p) paradigms.

In what follows, we use the term "node" to denote any of the computing/networking devices participating to the system, regardless of it being deployed on a moving vehicle or at a fixed RSU.

II. Basics of the communication middleware

A. Main goals and challenges

VANET platforms pose serious challenges about the network addressing scheme, the expected kind of interaction, the expected quality of service, and the desired degree of security. Device-based addressing is highly challenging when each node constantly changes position in the network, as this requires a huge overhead for updating the routing tables. Synchronous communication is unfeasible, given the unpredictable communication delays that may occur due to the multi-hop and transient interconnections. Broadcast trees involving preselected individual nodes cannot be feasibly maintained due to mobility. All the communication security is challenged by the transient nature of interaction and the general lack of an online public key infrastructure, that imposes the use of some sort of onboard tamper-proof device securing the electronic identity and the GPS receiver [9]. Privacy of individual users may be at risk if a node is required to provide own identity information along with own location/route data, but a lack of identification is in constrast with some safety-related uses (e.g. for ascertaining individual liabilities in case of car accidents or system misuses).

B. Below the ACIS middleware: node addressing, positioning, and networking

Below the ACIS middleware, VANET connectivity is ensured by the emerging IEEE 802.11p specification [5]. Node addressing is based on unique node identifiers (MAC addresses) plus geographical coordinates. To this end, the networking layer at each node is supposed to make use of one or more *positioning devices* (based on, e.g., GPS receiver, GSM cell id., or information diffused by RSUs), and a *street map* of the surrounding region. In the street map, each road is dissected into fixed *cells*, each univocally identified by the geographic coordinates of its middle point. The size of each cell is such that direct 802.11p communications can take place between any two nodes within adjacent cells. All vehicles share the same map, so that each vehicle can convert its current position into the identifier of the local cell [4]. *Short-range* communication, occurring within a single cell or between adjacent cells, use 802.11p broadcast addresses or MAC addresses of individual nodes. *Long-range* communications, instead, use cell identifiers as destination address and thus reach all vehicles in a cell, and are propagated by means of a *geographical routing* algorithm. The ACIS networking layer makes use of a variant of the algorithm proposed by Leontiadis and Mascolo [7]. RSUs attached to a wired network infrastructure can enhance the efficiency of long-range routing. As a possible future improvement, it would be possible to enhance the routing by taking advantage of the very same traffic information gathered at higher level, since crowded roads are known to be better carriers for vehicle-to-vehicle propagation [8], [3].

978-1-4577-0899-2/11/$26.00 ©2011 IEEE

C. A context-aware publish-subscribe middleware

When delivering information to nodes located in a given geographical area, it could be useful for the system to be able to spot only subsets of nodes as receivers. The only scalable (and privacy-compatible) way for selecting specific receivers in an area is by attribute or tag (e.g. “rescue vehicles”, or ”road side units”) rather than by name (e.g. licence plate or MAC address), because tags can be evaluated by each potential receiver based on locally determined interests or filters. Another advantage of tagged messages is that each receiver node could differentiate its reaction depending on the received tag and the driver’s current needs, or, more generally, the current *context* of the node. This leads to considering a one-to-many asynchronous communication model with anonymous receivers, a paradigm commonly referred to as *publish-subscribe (P/S)*, suitable to *event-driven* applications and not new in the VANET literature [6].

Context-awareness in P/S [2] has long been recognized as a generalization of traditional content-based P/S, and a key ingredient of ubiquitous computing platforms [10]. With ACIS, context-awareness means the ability, for the local P/S broker, to manage subscriptions that take into account the message content but also the physical context as currently sensed by the node. In ACIS, these physical data may include the current location and speeed, the local weather, any signals reported by the car’s internal sensors, the driver’s preferences, the surrounding vehicular traffic (nearby traffic observed locally, as well as remote traffic reported by long-range communications), context information shared by surrounding nodes via short-range communications, and any other information already notified to the node by previous events. Context information can also be of direct use for applications; for this reason, the middleware also supports a *query-based API* in addition to the P/S API.

A context information of key importance for the communication middleware of each node is the updated *list of neighbours* in the cell. To support this, each vehicle periodically sends short-range broadcast *heartbeats* that are received by each node within the cell. An heartbeat message reports the network address and geographical coordinates of the sender, the identifier of the cell it is currently traversing, the current speed and direction, plus room for other small sized information up to the MTU size. One of the neighbours is labelled as being the *cell leader*. The role and importance of neighbour list and cell leader will become clear when discussing the sensing and collection of traffic information (Section II-D) and the optimizations of short-range content distribution (Section II-E). Cell leadership is advertised as a flag in the leader’s heartbeat, along with the remaining leadership lifetime. In order to designate a statistically unique cell leader, each node periodically and autonomously evaluates its own distance from the center of its current cell and, if such distance falls below a system-wide threshold and the lifetime of the current leader approaches zero (or no leader seems to be left in the cell) and no other neighbour is candidate for leadership, advertises itself as a candidate for leadership by raising a flag in its own heartbeat messages; if no other candidates emerge for a while, the node turns the candidate flag into a leadership flag and subsequent heartbeats will advertise this to all neighbours; otherwise, the process is repeated until convergence.

Context information is better viewed as a continuous flow of data entering the node from various sources, both internal (e.g. vehicle sensors) and external (e.g. the heartbeats). This flow drives a continuous update of the middleware’s state representing the context. It could be useful for applications to be allowed to express interest in specific context conditions and be notified as soon as such conditions occur. Thus, it makes sense for the P/S broker to also manage subscriptions on *context predicates*. Subscribing to context predicates opens to a lot of possible uses; for instance, an application could be notified of nearby filling stations but only if the fuel level falls below a threshold decided by the application itself; a speed in excess of the local speed limit might be notified to the driver but only if a vehicle of the police is sensed in the vicinity, or the road pavement is slippery; a form of cooperative adaptive cruise control could be obtained by putting the vehicle under control of an application that subscribes to conditions about the speed and distance of the vehicles located in the forward path. In principle, subscriptions to context predicates could be implemented in the classical P/S paradigm by generating an event with suitable keywords or tags on each single context change and letting this event be notified to every application with compatible event filters, then letting each application update and evaluate its own copy of the context state upon notification. This however would be highly resource-consuming. A much better way to implement subscriptions to context predicates is to move predicate evaluation inside the P/S broker (thus in the middleware) so that notifications are only issued upon verification of subscribed conditions.

In a sense, the set of all outstanding subscriptions posted to the middleware by all the applications running on a vehicle represents a generalization of that vehicle’s *electronic horizon*. One key challenge for the middleware is to be able to react to sudden changes of the electronic horizon. For instance, a driver might suddenly decide to change her trip and, as a consequence, an entire geographical region that was considered as outside the electronic horizon becomes of interest for that vehicle’s middleware. Clearly, in order to accomodate such changes, at least to some extent, the middleware has to store events and context information belonging to a superset of the current horizon.

D. Collecting and reporting traffic information

The traffic conditions as observed locally by each vehicle concur to that vehicle’s current context, but are also delivered through the VANET in order to feed the traffic prediction model running at the data center, as one of the main goals of ACIS. Collection of traffic data towards a data center poses different challenges compared to purely decentralized and self-organizing approaches for discovery and dissemination of

traffic data. In the ACIS middleware, cell leaders are appointed to collect traffic data of the cell they are leader of. This is done by simply sensing the heartbeats coming from vehicles in the cell. Each heartbeat message reports various data concerning the sender, including its current speed. The cell leader can thus derive aggregate statistics of neighbours within the cell. In order to collect the data from all cells, the algorithm proposed and evaluated by Jerbi *et al.* [4] is used.

E. Short-range content distribution

Small-sized data can be efficiently shared in the short range by appending them to the heartbeat messages. For sharing increasingly larger pieces of information, however, the uncoordinated and simultaneous use of broadcast by nodes in the cell can become increasingly impractical due to the consumption of the available bandwidth. For this reason, short-range sharing of larger data needs a coordination mechanism that optimizes the medium access while avoiding broadcast storms (that could otherwise disrupt the heartbeats). The main purpose of such a cooperative sharing is to support *infotainment* applications, e.g., distribution of multimedia content. A more complete description can be found in reference [1]. The *de-facto* standard approach for handling data to be exchanged in p2p file-sharing applications is to dissect the content into fixed-size chunks, enabling multiple and concurrent transfers according to a number of delivery strategies. A frame is the "unit" of content distribution in this delivery scheme, and is composed by a number of equally sized placeholders for as many chunks. These placeholders may be filled by chunks from possibly different sources. While the chunk size is fixed, the size of a frame can vary at runtime depending on the number of vehicles, their average speed, bandwidth capacity, and the available data link capabilities (e.g., the availability of multicast). To ensure the correct reconstruction and retrieval of possibly heterogeneous contents, a proper *hash set* describing how the frame is organized must be created and sent along with the frame itself. According to the BitTorrent jargon, we define as *swarm* the set of nodes that are cooperating in the sharing and distribution of a given content. Multiple swarms may be simultaneously active in a region, and a node may participate to one or more of them. Nodes belonging to a swarm will exchange individual chunks with one another, depicting a sort of overlay network superimposed on the VANET. The exchanges are aimed to replicate chunks at every node in the swarm until all vehicles own a complete copy of the desired content. Some kind of coordination is needed for the efficient exchange of chunks, with the goal of maximize the aggregate transfer rate in the face of connection instability. Such coordination task is appointed to the *cell leader*, which, again in a BitTorrent jargon, acts as a "tracker". The leader thus initiates the swarm, and orchestrates participants by instructing each node about what chunks to exchange with whom. In order to orchestrate for efficiency, the leader runs a chunk selection strategy that depends upon the state/features of the underlying physical communication medium. The chunk selection strategy strives for concurrent use of different channels to increase the transfer rate, if they are available; allows some chunks to be broadcasted if there is a sufficient number of potentially interested receivers; assigns the best partners to any given node based on relative distance and speeds (the lesser the distance and relative speed, the more stable the wireless contact is supposed to be). Different from BitTorrent, with ACME it is not mandatory to have a *seeder*, i.e., a node owning all the chunks of the content (i.e., a complete copy of the file), because the same cooperation scheme can be used for *content aggregation*, namely, sharing of a content that originates as union of parts independently contributed by various peers, none of whom had a complete copy of the whole.

III. Architecture of the Middleware

ACME is essentially a publish-subscribe middleware where data from sensors, heartbeats, and received messages, are recorded into a repository and continuous update of internal variables representing relevant context information takes place. As we shall see, application subscriptions may take the form of context predicates possibly involving these internal variables. Context predicates are allowed to take the form of expressions onto the "trend" of some variables, by letting past values of such variables to be referenced to some extent; this opens to what we call *stateful subscriptions*, as opposed to traditional, "stateless" subscriptions where only the current values of context variables may be referenced. ACME is built as a set of components implementing core services, as shown in Figure 1. All data in the middleware, including messages being exchanged among components, are represented as XML documents. ACME architecture draws up the design of Contory

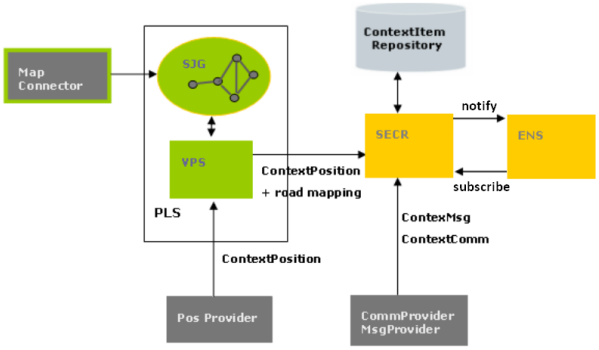

Fig. 1. ACME core services.

[10], a middleware for the management of sensor data, based on *ContextProvider* modules. A *ContextProvider* module converts raw input data to a common XML internal format. Each particular input channel requires a specific such module; this in turn guarantees independence of the core middleware services from the possibly changing data sources. Heartbeat messages, once entered, are received by the *PosProvider* component. This component encodes the heartbeat into an XML document called *ContextPosition*. The PosProvider periodically generates a fictious *ContextPosition* item with position and other

data from the ego-vehicle, to emulate the heartbeat message generated by the ego-vehicle itself. Other messages, with alert or advisory contents, are received by the *MsgProvider* component. Similar to a *ContextProvider*, the *MsgProvider* transcodes received messages to a common XML internal format.

ACME comprises three main service component (Figure 1):

- the *Positioning and Localization Service* (PLS) maps the geographic coordinates carried by the heartbeat messages on the street map, and keeps track of the electronic horizon of the ego-vehicle. It also receives prescriptions about the kind of vehicles to be considered of interest to the ego-vehicle, and immediately filters out all uninteresting *ContextPosition* items
- the *Spatial Environment Context Repository* (SECR) keeps track of all context data of interest for the vehicle, including the *ContextPosition* items.
- the *Event Notification Service* (ENS), supports the operations of `publish()`, `subscribe()`, and `unsubscribe()`, performs the event notifications both in stateful mode (using data recorded by the SECR) and in stateless mode, and incrementally evaluates aggregate functions over the flows.

The PLS builds an internal representation of the Segment/Junction Graph (SJG), recording the junctions and the roads with their angles. Since there exist different map representations, and no standard has been defined yet, road descriptions are obtained through connectors specific for each known map representation format and converted into a common internal representation. A road is composed by a list of linked edges, connecting two consecutive junctions. A road has an internal representation from the first to the second junction *R(J1, J2)*, so that the travel direction of a vehicle can be defined referring to such order. Moreover, the distances *D1(V)* and *D2(V)* of the vehicle from the 1st and the 2nd junction are added, to correctly position the vehicle on a road. A point *X* on a map can then be indicated as *P1(X, R) = (D1(X), R(J1, J2))*, or, respectively *P2(X, R) = (D2(X), R(J1, J2))*. The PLS also keeps an internal representation of the individual cells each road is composed of. Such a mapping of cells onto roads is provided by a separate document, independent of the specific format of street map.

The SECR resides in the volatile memory of the system, and stores up to N most recent *ContextPosition* items from each individual vehicle (currently $N = 2$) unless their timestamp is older than a threshold (currently set to 3 seconds). The SECR also stores context data coming from other sources, like internal sensors. These kind of data, together with the *ContextPosition* items, are generically called *ContextItems*. All *ContextItems* arriving to the SECR (including *ContextPosition* items) are also forwarded to the ENS engine which scans all outstanding subscriptions for possible matches and raises notifications to applications.

Subscriptions are submitted to ACME by applications when these create their own event filters. Each subscription is an expression of the XQuery language [11] representing a context predicate (XQuery has been selected since it allows to rapidly develop a prototype middleware). XQuery is also the language for querying the data stored in the SECR as well as the XML documents sent by the SECR to the ENS. ACME exploits the feature of XQuery *external variables* to maintain relevant context information, like the ones representing the vehicle's physical state. An XQuery expression referencing those variables can assume the role of a context predicate, suitable for context-aware subscriptions. XQuery is a rather powerful language but is unable to capture the notion of "history" of a variable, that is, past values of a variable up to some extent. To partially circumvent this limitation of XQuery, each new *ContextPosition* item related to vehicle V is always forwarded to the ENS along with a copy of the previous item related to V itself. In the example above, the index [1] refers to the first item encountered, that is the older one, while $varDir, $varId, $varDistJ1 and $varDistJ2 assume the values of the direction, identifier, distance from the first and, respectively, second junction, of the ego-vehicle. This kind of *stateful subscription* is useful for application to be notified about trends, shown by context data, that might suggest the onset of potentially hazardous conditions, like, for instance, abrupt changes of position or speed of nearby vehicles.

ACKNOWLEDGMENTS

This work has been carried out within the SIIT (Sistemi Intelligenti Integrati e Tecnologie) consortium in the framework of the project ACIS, which is partly funded by the Italian Ministry of Education and Research.

REFERENCES

[1] L. Caviglione and C. Cervellera. Design and Performance Evaluation of an Optimized Peer-to-Peer Content Replication Scheme for Vehicular Networks. In *Proc. SPECTS'08*, 2008.

[2] D. Frey and G.-C. Roman. Context-Aware Publish Subscribe in Mobile Ad Hoc Networks. In *Proc. COORDINATION 2007, LNCS 4467*, 2007.

[3] F. Giudici and E. Pagani. Spatial and Traffic-Aware Routing (STAR) for Vehicular Systems. In *High Performance Computing and Communications, LNCS 4467*, 2005.

[4] M. Jerbi, S.-M. Senouci, T. Rasheed, and Y. Ghamri-Doudane. An Infrastructure-Free Traffic Information System for Vehicular Networks. In *Proc. 66th IEEE VTC Fall*, 2007.

[5] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments. In *Proc. IEEE VTC Spring*, 2008.

[6] I. Leontiadis. Publish/Subscribe Notification Middleware for Vehicular networks. In *Proc. ACM MDS'07*, 2007.

[7] I. Leontiadis and C. Mascolo. GeOpps: Geographical Opportunistic Routing for Vehicular Networks. In *Proc. IEEE workshop on Autonomic and Opportunistic Communication*, 2007.

[8] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea. VANET Routing on City Roads using Real-Time Vehicular Traffic Information. *IEEE Trans. on Vehicular Tech.*, 58(7), 2009.

[9] M. Raya and J.-P. Hubaux. The Security of Vehicular Ad Hoc Networks. In *Proc. ACM SASN'05*, 2005.

[10] O. Riva. Contory: A Middleware for the Provisioning of Context Information on Smart Phones. In *Proc. Middleware 2006, LNCS 4290*, 2006.

[11] XQuery 1.0 and XPath 2.0 Functions and Operators,W3C Recommendation. http://www.w3.org/tr/xpath-functions.