An Overlay Architecture for Vehicular Networks*

Luigi Liquori[1], Diego Borsetti[2], Claudio Casetti[2], and Carla-Fabiana Chiasserini[2]

[1] INRIA Sophia Antipolis Méditerranée, France
Luigi.Liquori@inria.fr
[2] Dipartimento di Elettronica, Politecnico di Torino, Italy
lastname@tlc.polito.it

Abstract. We propose and discuss an overlay architecture relying on a mobile *ad hoc* network, called *Arigatoni on wheels* (Ariwheels for short). More specifically, Ariwheels is a virtual network organization that is designed for a vehicular network underlay environment. It provides efficient and transparent service advertising and retrieves services carried by on-board and roadside nodes. The paper outlines application scenarios for Ariwheels and evaluates them through simulation in a realistic vehicular environment.

1 Introduction

The explosive growth of the information technology fosters the design of large programmable overlay networks connecting virtual organizations of computers. These are capable of providing a rich spectrum of services through the use of aggregated computational power, storage and services. The main idea of programmable overlay networks is to realize computation, storage and information retrieval via a seamless, geographically distributed, open-ended network of bounded services owned by "Agents" and "Brokers", acting with partial knowledge and no central coordination. To make these services accessible to all Agents, an efficient and scalable communication protocol among them needs to be devised.

Arigatoni [1] is a structured multi-layer overlay network which provides service discovery with variable guarantees in a virtual organization, where peers can dynamically appear, disappear, and self-organize. To the best of our knowledge, Arigatoni is the first fully–programmable overlay architecture. It dictates how and where services are declared and discovered in the overlay, allowing peers to make secure use of global services. Thanks to such features, Arigatoni appears to be the natural choice for information delivery and sharing in a urban vehicular environment.

In the context of mobile *ad hoc* networks, solutions previously proposed in the literature have addressed service discovery [2] and subscription for mobile users. In particular, the works in [3,4] present service discovery protocols that are based on the deployment of a virtual backbone of directories within an infrastructure-less network. Each node composing the backbone acts as a service Broker by performing service

* This work is supported by AEOLUS FP6-IST-FET Proactive, INRIA Sophia Méditerranée through the project Colors-Ariwheels, University of Nice Sophia Antipolis, and by the *Regione Piemonte* through the project VICSUM.

A. Das et al. (Eds.): NETWORKING 2008, LNCS 4982, pp. 60–71, 2008.

discovery in its proximity, while global service discovery is provided by the cooperative action of the directories. In [5], service publishing and subscribing, i.e., the way clients and servers are matched together, is addressed: a cross-layer approach is applied, which leverages some routing-specific metrics, such as hop count or node traffic load. Also, a general discussion on the use of Brokers for publish/subscribe services in vehicular networks is presented in [6], while in [7] Broker entities are exploited in a publish/subscribe protocol combined with a content-based routing scheme.

In this paper, we design a new network architecture called Ariwheels which is a smooth integration of the Arigatoni programmable overlay network and a mobile *ad hoc* network. We consider both vehicle-to-vehicle and vehicle-to-infrastructure communication, since the number of vehicles properly equipped will realistically remain limited in the not too distant future, thus hampering a vehicular network relying only on mobile nodes. Pedestrian users, equipped with small wireless devices, are naturally considered as mobile. We define the semantics and the interaction among logical entities in Ariwheels, and investigate some performance metrics that validate our design.

Therefore, the main contributions of this paper are:

- to define the Ariwheels overlay architecture units;
- to provide insight of the overlay architecture over a vehicular underlay network;
- to set up and simulate various scenarios, via the Omnet++ [8] event-based simulator.

The rest of the paper is structured as follows: after Section 2 describing an introduction of the Arigatoni programmable overlay network, Section 3 introduces the Ariwheels's architecture and its logical units. Section 4 shows our simulation results, while Section 5 provides some concluding remarks.

2 Arigatoni in a Nutshell

What follows is a short overview of the activity of the main entities and of the protocols involved (the interested reader can refer to [9,10,1]).

2.1 Functional Units

Two main logical entities (the Agent and the Broker) and two basic protocols (a registration and a service discovery protocol) are the core of the Arigatoni architecture.

The Agent is a computing device with wired/wireless connectivity capabilities. It does not necessarily need to be a high-end device, such as a supercomputer; on the contrary, it may have limited storage and computation capabilities, and few basic installed applications (a simple editor, one or two compilers, an email client, a mini browser).

Agents are organized in *colonies*, lead by a Broker. Unlike the Agent, though, the Broker is required to be a mid- to high-end device, equipped with a high–speed wired/wireless connection and a service table, crucial to perform the publish/subscribe content-based routing. Given the hierarchical architecture, colonies may recursively be embedded into super-colonies, each lead by a super-Broker.

The Agent should be able to work in *local mode* for all the tasks that it can manage locally or in *colony mode*, by first registering itself to one or many colonies of the overlay, and then by asking and serving colony-originated requests via the Brokers. The tasks of an Agent can thus be summarized as:

- discovering the address of one or more Brokers, acting as colony leaders, upon its arrival in a "connected area";
- registering to one or many Brokers, thus entering the virtual organization;
- requesting and offering services to other Agents, through its own Broker;
- connecting directly to other Agents in a peer-to-peer fashion, and exchanging services between each others. Note that an Agent can also be a service provider. This symmetry is one of the key features of Arigatoni.

The Broker requires higher capabilities than an Agent to store and manage the content-based routing table of the colony it leads. Such table is essential to route queries, and the Broker must also efficiently match and filter the routing table against a received query. The tasks of a Broker are:

- discovering the address of another Broker, and possibly embedding its colony into the other Broker's;
- registering/unregistering Agents in its colony and updating the internal content-based routing table accordingly (who offers what within the colony, its address, and other geographical information);
- receiving Agents service requests, discovering the services that satisfy an Agent request in its local colony, according to its content-based routing table, or delegating the request to its direct super-Broker;
- in case the Agent request can be satisfied, forwarding, in a service response, all the information necessary to allow the requesting agent to communicate directly with the agent offering the service;
- in case the agent request cannot be satisfied, notifying the requesting agent, after a fixed timeout period, that its service request could not be served.

2.2 Arigatoni Service Discovery and Intermittent Protocols

There are mostly two mechanisms of service discovery, namely:

- the process of a Broker finding and negotiating services to serve an Agent request in its own colony;
- the process of an Agent discovering a Broker, upon physical/logical insertion in a colony.

The first discovery is processed by Arigatoni service discovery protocol, while the second is processed out of the Arigatoni overlay, using well-known network protocols like DHCP, SLP in Bluetooth, or Active/Passive Scanning in Wi-Fi.

The Service Discovery Protocol (SDP). It is used by a Broker to find and negotiate services to serve agent requests in its own colony. SDP allows the request for multiple

services and service conjunctions (i.e., each agent may offer several services at the same time). The SDP protocol allows Agents to:

- ask to a Broker a request for a service set S;
- reply to a Broker the availability to offer a service set S'.

The colony's Broker handles the service request received through SDP and looks up the service set in its routing table, filtering S against the offered set S'. If a match is found, the Broker returns to the requesting agent the address of the agent matching, partly or fully, the request. From then on, the two agents interact in a peer-to-peer fashion, without further intervention of the Broker.

Each Broker maintains a content-based *routing table* locating the *services* that are registered in its colony. The table carries one entry for each member matching the ID of the Agent with the set of services it can offer.

The Virtual Intermittent Protocol (VIP). It manages peers' participation in Arigatoni colonies. The protocol deals with the *dynamic topology* of the overlay, by allowing individuals to login/logout to/from a colony. Registration is the act through which an Agent becomes member of a colony. An Agent is allowed to unregister when it has no pending service requests or offers. Agents that abruptly unregister or behave as "free riders" (using other Agents' services without offering or giving theirs in return) are tagged as unfair and may be forcefully unregistered from a colony by its Broker.

An Agent registers to a colony with a list of services. If a Broker accepts an individual in its colony, then it sends a service update to its direct super-Broker in order to communicate the availability of the new services in its colony. This message is then propagated from Broker to Broker until the root (if any) of the multi-layer overlay is reached. This means a high node churn forces routing tables to be *faulty* until all service updates are properly propagated. As such, service registration in an overlay network computer is an activity that must be taken seriously into account.

3 The Ariwheels Vehicular Overlay Network

Ariwheels is an overlay architecture for vehicular networks, based on Arigatoni. Overlay networks in a vehicular setting are problematic because of the highly dynamic nature of mobile nodes, and the limited number of vehicles that, at first, will be equipped with the necessary communication devices. For this reason, we envision an architecture encompassing both mobile users and infrastructure nodes.

The most challenging problem in Ariwheels design is an efficient mapping between physical devices in the vehicular underlay network and virtual entities in the Arigatoni overlay network.

3.1 Ariwheels Architectural Overview

We consider an urban area in which a Mobile Ad hoc Network (MANET) is deployed. Such MANET is populated by both mobile users, e.g., pedestrians with hand-held devices, cars equipped with browsing/computational capabilities, public-transportation

vehicles and roadside infrastructures such as bus stops. All devices are supposed to have a wireless interface. Depending on their mobility, they may also be equipped with a wired interface. Such is the case of wireless Access Points (APs), which are installed at a bus stop, in order to provide connectivity either to users waiting for a bus or to the bus itself (hence to its passengers). In such settings, devices carried by cars and pedestrians play the role of Mobile Agents; roadside infrastructures (APs) and public transportation vehicles (buses, trams, cabs...) act as Brokers, although some distinctive behaviors have to be introduced.

A Broker in Ariwheels is logistically represented by a bus stop, and its colony is composed by Mobile Agents that have registered to it when they were within radio range of the AP installed at the bus stop. Therefore, a colony is a logical entity, whose members may be physically localized anywhere within the area where Ariwheels is deployed (i.e., they have to be within the Broker radio range only while registering to it).

However, to take into account the high mobility of the scenario and enhance its performance in terms of load balancing and service response time, we introduce an additional, Ariwheels-specific entity, the *Mobile Broker* (mB). This unit is a public transport vehicle equipped with a scaled-down Broker-like wireless device. Every Mobile Broker is *associated* to (i.e., it has the same identity of) a single Broker. Such association exists at the overlay level and holds throughout its bus route. Clearly, at the underlay level, connectivity between the Mobile Broker and the associated Broker may at times be severed.

The main aim of the Mobile Broker is to introduce the novel concept of *colony–room*: a small subset of Mobile Agents with a wireless connection to the Mobile Broker (pedestrian users on the bus, or pedestrian/vehicles around the bus or travelling along the same bus direction during a traffic jam...). In addition, thanks to its mobility, the Mobile Broker can collect registrations from Mobile Agents that were too far from the AP of the associated Broker, and, therefore, might have never had the chance to register to it. As such, every Mobile Broker associated with a Broker may enrich the colony with new members, and with the contents they own, that are not within the Broker's radio range.

The Mobile Broker collects (un)registrations, service requests and service offers from the Agents within the colony–room. When a wireless connection has been established between the Mobile Broker and a roadside AP (not necessarily corresponding to the associated Broker), the data path to the associated Broker is again available and an information exchange takes place resulting in the updating of each other's data. Specifically, the following actions occur. Firstly, the associated Broker merges the Mobile Broker's routing table with the one it currently carries. Then, the associated Broker handles the registration/discovery information and generates the appropriate responses. Finally, depending on the response time, the responses are returned to the Mobile Broker before it leaves the wireless AP coverage, or the next time it connects to an AP: this will normally happen at the next bus stop.

Figure 1 illustrates the relationships among overlay and underlay entities in Ariwheels. A central coordination entity is located at a headquarter (HQ), in our case corresponding to the local transportation authority building. The coordination entity plays

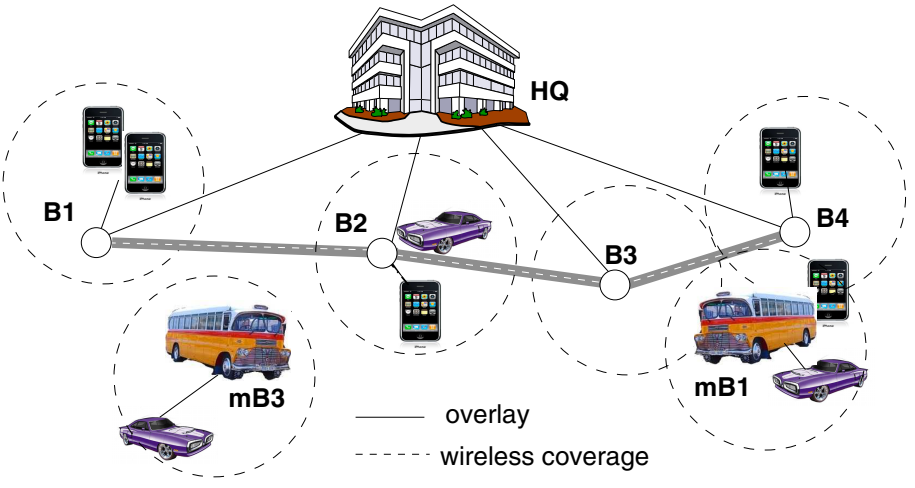

Fig. 1. An Ariwheels scenario

the role of a super-Broker and it is provided with a wired connection to each of the 4 roadside AP at bus stops (B1 to B4). Mobile Brokers (mB1 and mB3) shuttle between bus stops, each carrying a different Broker association (to B1 and B3), while Mobile Agents (portable devices or on-board car devices in the figure) are either connected to Brokers or Mobile Brokers, depending on their mobility.

3.2 Behaviors of Ariwheels Entities

We can now summarize the different activities of the three main entities in Ariwheels, i.e., Mobile Agents, Brokers and Mobile Brokers.

Mobile Agents. Their activity is carried out through the following main operations.

- *Broker Discovery*: the reception of a HELLO message from one or more Brokers (and/or Mobile Brokers) by an Agent that is not already registered to a Broker; the choice of the Broker to which to register is performed using information stored in the HELLO message.
- *(Un)Registration*: the Agent sends a service registration (SREG) message of the VIP protocol trying to register to a new Broker or to unregister from the current Broker.
- *Service Request/Response*: these tasks require that the Agent is already registered to a Broker and that it is part of a colony. It can send a service request (SREQ) of the SDP protocol to its Broker or a service response (SRESP) offering some services. The service will be then exchanged in peer-to-peer fashion.

Brokers. Their activity is carried out through the following main operations.

- *Colony Health*: periodically broadcasting of HELLO messages in order to establish the liveliness property of Mobile Agents;

- *Colony (Un)Registration*: to and from a higher-level Broker;
- *Colony Management*: the Broker interacts with colony members, (un)registering and handling service requests through the VIP and SDP protocols.

Mobile Brokers. The activity of a Mobile Broker is carried out through the following three main operations.

- *Broker Association*: the process of associating to a specific Broker; the Broker for which the Mobile Broker acts as colony-room is chosen according to some policy (see below) and the association is held throughout the Mobile Broker's route.
- *Colony-Room Advertising*: periodical broadcasting of HELLO messages along its whole route; HELLO messages forward information about the associated Broker for which they are acting as colony-room.
- *Relaying*: relaying VIP and SDP messages from (to) Agents inside the colony-room to (from) the associated Broker. Relaying may occur at once if a wireless connection to an AP exists, or it may be deferred until the wireless connection is re-established.

3.3 Membership Policies

As a byproduct of the Ariwheels architecture, members of a colony will be geographically distributed, although this distribution should be carefully planned by a Broker (accepting or refusing registration requests) for load balancing purposes.

VIP registration policies are usually not specified in the protocol itself; thus every Broker is free to choose its acceptance policy. Different self-organization policies may be used to address issues such as load-balancing and colony specialization. Possible policies therefore are: (i) *mono-thematic*: a Broker accepts an Agent in its colony only if it offers services that the colony already owns in large quantities, so as to increase its specialization; (ii) *balanced*: a Broker accepts an Agent in its colony only if it offers services that the colony lacks, with the aim of evening out its service offer; (iii) *unbalanced*: a Broker unconditionally accepts all Agent registration. Membership to a colony is also affected by the Mobile Broker association. The choice of which Broker is associated to a Mobile Broker can be performed by a specific load balancing algorithm that is run periodically, i.e., when the public transportation vehicle leaves the deposit and sets off on its route. One possible aim of the load balancing algorithm is to let the Mobile Broker collect Agents for a Broker with a scarcely populated colony at the time of the Mobile Broker departure; other aims, such as specializing the colony population, can be also envisaged.

3.4 Service Discovery in Ariwheels

Service discovery over a MANET, hence in Ariwheels, plays a crucial role in the successful retrieval of information. There are two main concerns regarding the issue of service discovery. The first one is to *expedite* the selection of the Mobile Agent providing the service, given that Ariwheels Agents are nominally free to roam in and out of their own Broker's underlay reach. Therefore, if a service is advertized by more than one Agent, and a request for that service is pending, a Broker should be given the opportunity to hand the service request over to the Agent that is more likely to be within

its reach. The second concern is the *suitability* of the match that the Broker is about to create. Indeed, finding a "good" Agent carrying the requested service must also account for its ability to establish an effective communication channel with the Agent requesting the service. The mobility of both Agents (the requesting one and the potential provider) must be accounted for, e.g., by selecting Agents that are either within radio range of each other, or that are likely to remain within some AP coverage for enough time.

From a practical standpoint, service discovery at an Arigatoni Broker is carried out through a table that mainly records colony member IDs and their service lists. In Ariwheels, however, the table information for each Agent is integrated by a *liveliness* field, indicating the time elapsed since the last contact from that Agent, and by a *mobility* field, that can be used to pinpoint the position of the Agent and to infer the direction of its movement (the latter information is provided by the Agent in its last message sent to its Broker).

In order for these additional table parameters to be maintained up-to-date, Agents are required to interact with their own Broker on a regular basis. Such interaction, in the form of a refresh SREG, should not be limited to the period when the Agent is within its own Broker radio range. Rather, it should also be promoted when the Agent is within *any* Broker range (whence it will be relayed to the Agent's Broker). The refresh SREG will therefore be issued by the Agent upon hearing a HELLO message from a Broker[1]. The rationale is to let the Agent's Broker know that the Agent is within coverage of whatever wireless technology is used by any Ariwheels Brokers, hence it is readily accessible if a service is requested.

For each Ariwheels Broker, the content-based routing table has the form:

Agent ID	Services	Liveliness	Mobility
A	$\{\mathsf{S}_i\}^{i=1\ldots n}$	t	(x, y, θ, v)
...	...	...	...

where $\{\mathsf{S}_i\}^{i=1\ldots n}$ is the set of services it can offer, t is the time passed since the Agent has sent an Ariwheels message, and (x, y, θ, v) is a quadruple denoting physical position, direction and speed (all those information easily retrievable by a GPS module, or computed through it). The table is updated according to the dynamic registration and unregistration of Agents in the overlay. When an Agent asks for a service, then the query is *filtered* against the routing tables of its own Broker; in case of a *filter-failure*, the Broker forwards the query to its direct super-Broker.

4 Simulation Scenario

We implemented Ariwheels in the Omnet++ [8] simulator, coding the overlay part and exploiting the existing wireless underlay network modules. In the underlay we used IEEE 802.11 at the MAC layer and the DYMO routing protocol (an AODV-like reactive routing protocol).

We tested the performance of Ariwheels in a vehicular environment. We used a realistic mobility model generated by VanetMobiSim [11], whose output (mobility traces)

[1] Broadcast storms are prevented by forcing a latency period between consecutive SREG from the same Agent.

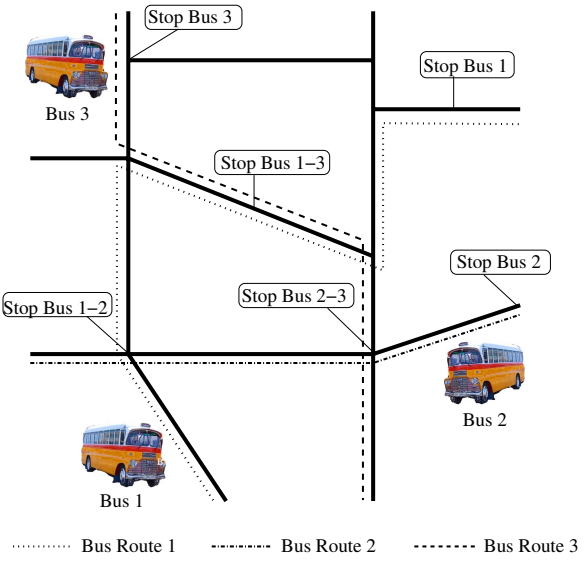

Fig. 2. The simulated city topology

was fed to the Omnet++ simulator. Vehicles travel in a 1 km^2 city section over a set of urban roads, which include several road intersections regulated by traffic lights or stop signs. In particular, we adopted the IDM-IM microscopic car-following model [12], which allows us to reproduce real-world traffic dynamics as queues of vehicles decelerating and/or coming to a full stop near crowded intersections.

We assumed that 60 vehicles enter the city section from one of the border entry/exit points, randomly choose another border entry/exit point as their destination, compute the fastest path to it and then cross the city section accordingly. A vehicle entering the topology is assigned a top speed of v m/s, that it tries to reach and maintain, as long as traffic conditions and road signs allow it to. When a vehicle reaches its destination, it stops for a random amount of time, uniformly distributed between 0 and 60 s, then it re-enters the city section. In our simulations, we tested two different top speeds v: 9 m/s (approx. 32 km/s) and 15 m/s (approx. 54 km/s).

Upon entering the topology, a vehicle acting as Mobile Agent owns a set of 12 unitary services (e.g., files, traffic informations, point of interests) randomly chosen from a set of 20 services. A Mobile Agent issues a (SREQ) for a service it is missing and the inter-request time is supposed to be exponentially distributed with parameter $\lambda = 0.05$ [req./s]. As typical in the publish/subscribe paradigm, where peers are not slaves, upon receiving a SREQ for a service it owns, a Mobile Agent sends back a positive response with a certain probability, which is set to 0.9 in our simulations.

The simulated city topology, shown in Figure 2, features 6 bus stops with APs, each corresponding to a Broker. Furthermore, 3 buses acting as Mobile Brokers weave their own routes across the topology, among a population of as many as 60 vehicles acting as Mobile Agents. Each bus carries 10 passengers equipped with Mobile Agent capabilities, and it associates to the Broker with the smallest colony at the time of departure

from the bus station. Brokers apply the unbalanced acceptance policy and filter the routing table against a received query by using the liveliness information only.

5 Performance Evaluation

Although the number of parameters that can potentially be analyzed is quite high, we focus on few selected sets of results for the sake of brevity. Therefore, we compare results for different radio ranges, different vehicle speeds, scenarios with and without Mobile Brokers (i.e., buses), and we evaluate the impact of a liveliness-based filtering.

We initially address the success probability of a service request, i.e., the probability that a service request is positively answered by a Mobile Agent either in the requester's

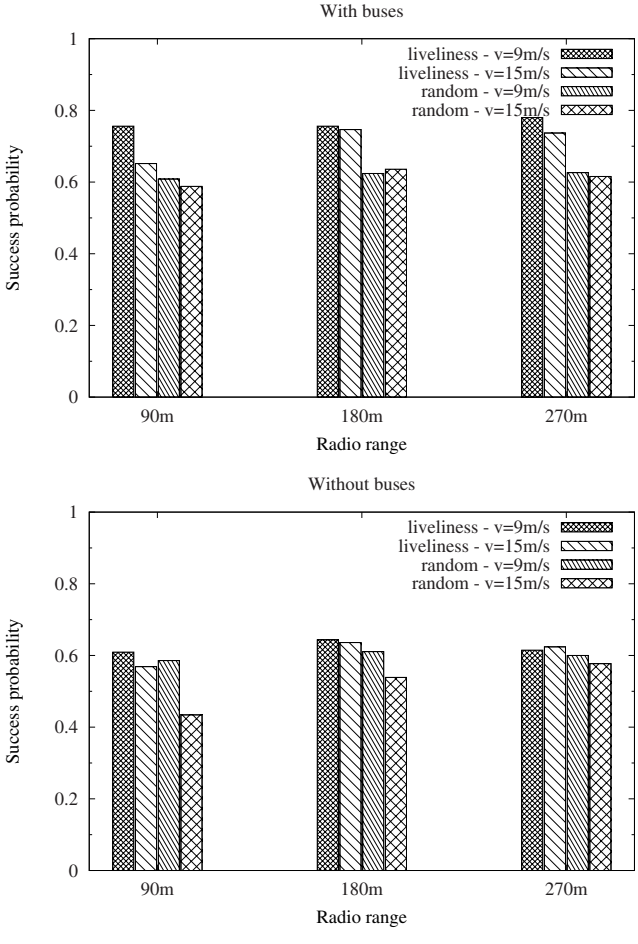

Fig. 3. Success probability of a service request for three different values of radio range (90, 180, 270 m). The results obtained under different mobility as well as routing table filtering are compared. The two histograms refer, respectively, to the cases with and without buses.

Table 1. Average service request response time (in seconds)

	with buses				without buses			
	$v = 9$ m/s		$v = 15$ m/s		$v = 9$ m/s		$v = 15$ m/s	
Range	Liveliness	Random	Liveliness	Random	Liveliness	Random	Liveliness	Random
90	3.42	8.38	3.91	6.75	7.12	7.38	5.30	8.33
180	2.73	5.11	4.81	5.20	4.85	7.30	4.46	7.23

Table 2. Super-Broker delegation probability

	with buses				without buses			
	$v = 9$ m/s		$v = 15$ m/s		$v = 9$ m/s		$v = 15$ m/s	
Range	Liveliness	Random	Liveliness	Random	Liveliness	Random	Liveliness	Random
90	0.13	0.22	0.12	0.20	0.31	0.38	0.32	0.45
180	0.13	0.19	0.23	0.21	0.33	0.44	0.33	0.39

colony or in another colony (after delegation to the super-Broker). Figure 3 carries a comparison of success probabilities for an Ariwheels system in which Brokers either use randomly-ordered routing tables, or list Agents in increasing order of time elapsed since their last contact, i.e., of their "liveliness". In the latter case, a service request is more likely to be routed toward an Agent that is still within radio coverage of a Broker. The top plot refers to the scenario with roaming buses acting as Mobile Brokers, while the bottom plot refers to the scenario without any buses. Different vehicles speeds are considered. It can be seen that the liveliness-based filtering yields the best results in almost all cases, and its prominence is especially clear when vehicles are slowly moving (hence lingering longer within radio coverage of a Broker). The gap between liveliness-based and random filtering is reduced when the radio coverage increases, bringing more vehicles within extended radio coverage. The price to pay, however, is the increase in 802.11 channel congestion, and consequently almost no gain as the coverage grows from 180 m to 270 m. This trend is confirmed by the scenario without buses, although the success probability is generally smaller due to the lack of the "colony-room" effect introduced by buses.

We point out that, with a widespread content distribution as the one we simulated (namely, each Agent owns 12 out of 20 services), a fully wired static scenario would gave more than 90% successful service requests. In a MANET, instead, several underlay-related events are responsible for missing responses: for example, a miscue from the routing protocol may lead an Agent to wrongly believing that it is within radio range of a Broker (or that a Broker may be reachable in multihop fashion through buses or other vehicles); an SREQ will therefore be issued but never delivered to the Broker, negatively affecting the success probability.

Another metric of interest is the average time after which a response to an SREQ is returned to the requesting Agent (Table 1). As can be expected, the trends already observed for the success probability are found in the response times as well. Finally,

Table 2 reports the probability that an SREQ cannot be solved within the requester's colony and is thus sent to the super-Broker, which is in charge of delegating it to another Broker. Again, the lower delegation probability exhibited by the liveliness-based filtering in the scenario featuring Mobile Brokers confirms that keeping tabs on which Agents are promptly available dramatically increases Ariwheels performance.

6 Conclusions and Future Work

This work presented Ariwheels, a structured overlay architecture for vehicular networks that aimed at facilitating the offering and retrieving of services by mobile users. Simulation results in a realistic environment have shown that Ariwheels provides good performance in service localization and have highlighted critical issues such as the need for proper content-routing table filtering and the role played by mobile Brokers (public transportation vehicles) in supporting and promoting the information exchange.

Future work will focus on further specializing the routing table filtering, including localization information as a means to select a good peer match to the requesting Agent, as well as reputation and trustfulness mechanisms to maximize the response rate and fire "free riders".

References

1. Liquori, L., Cosnard, M.: Logical Networks: Towards Foundations for Programmable Overlay Networks and Overlay Computing Systems. In: TGC, Trustworthy Global Computing. LNCS, Springer, Heidelberg (to appear, 2008)
2. Fiore, M., Casetti, C., Chiasserini, C.-F.: Efficient Retrieval of User Contents in MANETs. In: IEEE INFOCOM., pp. 10–18. IEEE, Los Alamitos (2007)
3. Kozat, U.C., Tassiulas, L.: Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In: IEEE INFOCOM., pp. 1965–1975. IEEE, Los Alamitos (2003)
4. Sailhan, F., Issarny, V.: Scalable Service Discovery for MANET. In: IEEE PerComm, International Conference on Pervasive Computing and Communications, pp. 235–244. IEEE, Los Alamitos (2005)
5. Varshavsky, A., Reid, B., de Lara, E.: A Cross-Layer Approach to Service Discovery and Selection in MANETs. In: IEEE MASS, International Conference on Mobile Ad-hoc and Sensor Systems, mettere numero pagina, pp. 8–12. IEEE, Los Alamitos (2005)
6. Iftode, L., Borcea, C., Ravi, N., Nadeem, T.: Exploring the Design and Implementation of Vehicular Networked Systems. Technical Report DCS-TR-585, Rutgers
7. Lundquist, D., Ouksel, A.: An Efficient Demand-driven and Density-controlled Publish/Subscribe Protocol for Mobile Environments. In: ACM International Conference on Distributed Event-based Systems, Toronto, Canada, pp. 26–37 (2007)
8. OMNET++ Discrete Events Simulator, `http://www.omnetpp.org`
9. Benza, D., Cosnard, M., Liquori, L., Vesin, M.: Arigatoni: Overlaying Internet via Low Level Network Protocols. In: IEEE JVA, John Vincent Atanasoff International Symposium on Modern Computing, pp. 82–91. IEEE, Los Alamitos (2006)
10. Chand, R., Cosnard, M., Liquori, L.: Powerful Resource Discovery for Arigatoni Overlay Network. Future Generation Computer Systems 24(1), 31–38 (2008)
11. VanetMobiSim. `http://vanet.eurecom.fr`
12. Fiore, M., Haerri, J., Filali, F., Bonnet, C.: Vehicular Mobility Simulation for VANETs. In: IEEE ANSS, Annual Simulation Symposium, pp. 301–309. IEEE, Los Alamitos (2007)