Publish/Subscribe Notification Middleware for Vehicular networks

Ilias Leontiadis
Department of Computer Science
University College London
Gower Street, London, WC1E 6BT
United Kingdom
I.Leontiadis@cs.ucl.ac.uk

ABSTRACT

There is a large number of interesting applications for vehicular networks: traffic information dissemination, warnings, free parking spots finders, fuel prices advertising, etc. The Publish/ Subscribe (P/S) communication paradigm enables the application developers to easily design flexible notification systems. Furthermore, it enables the drivers/vehicles to indicate their interests about certain types of notifications (e.g. receive warnings concerning traffic jams only within 1km from the vehicle's route). And finally, P/S is an asynchronous communication protocol (spatial and temporal decoupling) that is suitable for the delay-tolerant network conditions.

Our main goal is to design a P/S Middleware for vehicular networks that considers location and time in its primitives. We would like to enable the application developers to easily publish notification in specific location by treating location as context. We will use subscriptions and the navigation system to automatically express interests on the affected vehicles and to filter incoming notifications. Additionally, our middleware incorporates the appropriate communication mechanisms that implement the tasks instructed by the middleware primitives.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Middleware, Wireless Ad-Hoc networks, Vehicular Networks, Publish-Subscribe Systems

1. INTRODUCTION

One major application field for delay tolerant, wireless ad-hoc networks is vehicular communication. Vehicles can be equipped with short range radio to either connect with fixed *infrastructure* (e.g. road-side basestations or WiFi hotspots) or connect with each other to perform *Vehicle-to-Vehicle (V2V) communication*. There are many possible deployments of such networks: 1) Wired backbone with infostation and with wireless last hops, 2) a pure Vehicle-to-Vehicle (V2V) wireless ad-hoc network, and 3) hybrid approaches where V2V communication is used where there is no infrastructure available.

The hybrid approach is the most promising since it combines the advantages of both ad-hoc and infrastructure communication. Infostations can be fixed access points that are potentially connected to the Internet that provide fast and secure service to the network. They may act as dissemination points, from where information to/from the backbone network flows from/towards the vehicles. Vehicles communicate with each other, disseminating messages further. In the same way, vehicles can exchange local information realtime (e.g. parking spots) without consuming resources of local infostations.

These networks can also be considered as intermittently connected networks with some unique characteristics: the network topology is constantly changing (as in mobile ad hoc networks) but in a somewhat predictable way (e.g., cars move on roads). Furthermore, cars tend to move in clusters in a specific direction, the density and the connectivity times vary greatly (e.g. cars moving in opposite directions or cars moving at the same direction). Consequently, *asynchronous* mechanisms are better suited to route and disseminate information in such networks.

Vehicular ad-hoc networks (VANETs) may have a large variety of interesting applications: First of all, road safety applications (accident warnings, red-light warnings, leading vehicle emergency break, platoon of vehicles). Furthermore, information dissemination applications (parking spots, traffic warnings/conditions, fuel prices, local information, landmarks, bus times). Additionally, vehicles can be considered as distributed sensors that collect various observations and report them to local basestations (average speeds, potholes, temperature, pollution etc). Finally, entertainment applications (file sharing, advertisements, voice communication with nearby vehicles, gaming etc).

There are currently a lot of ongoing research projects that concern routing and disseminating information in VANETs. For instance, CarTel [2] is a mobile sensor computing system designed to collect, process, deliver, and visualise data from sensors located on mobile units such as automobiles using opportunistic wireless connectivity (e.g., Wi-Fi, Bluetooth) to the Internet, or to "data mules" such as other CarTel nodes. In DieselNet [1], that currently consists of 40 buses, they use DTN policies to route information from/to passengers and the Internet. The authors of Drive Through Internet [8] present a "session" protocol that offers persistent end-to-end communications even in the presence of interruptions. Fleet-

MDS'07, November 26-November 30, 2007, Newport Beach, CA, USA

Figure 1: Information is gathered by vehicles, geographically routed to nearest infostation. Information can be disseminated in publish area for a certain time interval.

Net [12] also employs a message ferrying approach for data delivery in sparse mobile ad hoc networks. In [10, 13, 9] they employ mechanisms such as statistics of previous encounters, or precise mobility schedules to dispatch information.

With respect to these works, currently there is no middleware platform that enables the application developers to publish notifications to a group of affected vehicles. We believe that it is very important to disseminate information only to vehicles that actually need it, in order to minimise the communication overhead and to avoid disrupting drivers that shouldn't be involved. In the same way, there is no framework that enables the drivers/vehicles to express interests about certain notifications. But most importantly, current mechanisms do not take into account a key tool that is available in most modern vehicles: the *satellite Navigation System (NS)*. As we are going to see, the navigation system provides very valuable information to our notification middleware.

Therefore, we strongly believe that there is a need for middleware architecture that takes into account the unique characteristics of vehicular networks and provides asynchronous communication primitives to support VANET applications. In this paper we examine how we to use the *Publish/Subscribe* communication paradigm [3], with the aid of the navigation system, in order to implement a notification service middleware for vehicular networks. Our aim is to design a suitable Publish/Subscribe primitives that equip the publishers with a flexible tool in order to dispatch notifications to affected areas/vehicles for a certain time interval. The subscriptions indicate the driver's/vehicle's interests about certain types of notifications and are used to 1) *filter* and 2) *disseminate* information to them. Our approach utilises both the centralised infrastructure and the decentralised car to car communication technologies. At the same time, the interaction with the navigation system is a key aspect of our middleware: we exploit the information that can be extracted to perform 1) *geographical routing*, 2) *subscription generation and matching* and 3) *information dissemination*.

2. SCENARIO

In this section we present a scenario motivating our work and our approach at a glance. We consider vehicles as mobile sensors that collect information about traffic conditions, accidents, etc (e.g., like in CarTel [2]). Vehicles report this information to the closest known infostation or WiFi hotspot (Figure 1) using geographic V2V routing or connect directly with the infostation.

A centralised system can combine the information gathered from various sources and generate traffic warnings concerning the affected areas. Initially, warnings are sent to the nearest infostation. From there, they are routed to the affected road segments using V2V communication. Upon reaching the area, we use dissemination techniques to spread the information to *affected vehicles* From the vehicles' point of view, drivers require to receive only related notifications: For example, a vehicle can require to receive only traffic warnings that may affect their route to their final destination (Figure 1). Therefore, some filtering of such notifications is required.

Once the notification time starts, the notification is then continuously propagated inside the area until its expiration time, so that new vehicles entering the zone are informed. The navigation system of vehicles that receive such a warning can evaluate the information provided (if interested or not) and automatically recalculate a route avoiding the affected areas.

It is worth noting here that this scenario can be easily expanded to support numerous applications: accident warnings, road works, free parking spots, fuel prices, advertisements, etc.

The specific design goals, and advantages, of our middleware are:

- *Use of available infrastructure*. Our middleware assumes a hybrid setting where infostations and V2V communication can be employed. V2V communication is only used to extend the coverage of infostations/ hotspots and to disseminate local information (i.e., spreading the information inside the affected area).

- *Asynchronous Communication*. We need to deliver the notifications to multiple vehicles. We need a middleware that decouples not only the locations of the publishers and subscribers, but also decouple them temporally.

- *Inform only affected vehicles.* We need mechanisms to filter incoming notifications that are only relevant to the driver/ vehicle (e.g. that affects its route). Furthermore, We would like to avoid disseminating the notification in areas where there are no interested vehicles (to avoid unnecessary communication overhead).

- *Enable the vehicles/developers to express interests/affected areas*. The middleware should provide flexible primitives enabling the application developers and drivers to easily describe interests. Ideally, the driver only needs to define some general policies (e.g., "receive warnings concerning route" or "receive fuel prices from stations on my route when fuel is lower than X") and their middleware can match all the appropriate subscriptions automatically. Similarly, publishers need the same level of flexibility to generate the appropriate notifications: they should be able to define the dissemination areas and the areas affected.

- *Time-stable notifications*. Vehicular notifications may have to be delivered over a time interval. Since there is mobility, the information needs to be delivered to vehicles that will enter the publication area anytime during that time interval.

- *Geographical-aware dissemination*. Geographic delay tolerant routing protocols should be utilised to route the information *from* and *to* the vehicles from an infostation. Furthermore, we need the publisher to be able to geographically constrain the dissemination.

3. CONTRIBUTION

To address these goals we designed a Publish/Subsribe (P/S) notification middleware for vehicular networks. In this section we describe an overview of the three key components of our middleware design: 1) P/S primitives to enable the publisher to design and dispatch notifications and subscribers to describe interests. 2) Interaction with the satellite Navigation System (NS) to aid the P/S and communication components. 3) Communication mechanisms to route and disseminate the notification as instructed by the primitives.

We envisage that our middleware will be used by notification applications to route and spread information into the vehicular environment. Examples of possible applications could range from traffic/accident warning applications to advertisement dissemination.

3.1 Publish/Subscribe Middleware

We selected the Publish/Subscribe paradigm due to the fact that it enables 1) *multicast delivery*, 2) *asynchronous connectivity* (spatial and temporal decoupling), 3) ways to *express interests* (topics), and 4) *matching engines* (filtering). All these features will provide a powerful and flexible middleware to perform notification dissemination in the vehicular environment.

Our primitives include *location* and *time* aspects in order to enable the application developers define the area of interest, the affected areas and the time period during which the notification should be disseminated. The publishers can be either centralised authorities (e.g. transport agencies), or even individual vehicles (e.g. warning about an accident) . Additionally, subscriptions can be used to 1)*filter* incoming notifications that concern the vehicle and its driver, 2)efficiently *disseminate* information in areas where there is interest. We will go into details in the next section.

3.2 Navigation System

Navigation systems provide valuable information that is currently not used in existing middleware architectures. The most important information is the *suggested route* of the vehicle. The suggested routes indicate *where the vehicles are likely to be in the future*. Therefore, *our first aim* is to examine how our middleware can use the suggested routes to efficiently perform geographical routing.

A part from navigation suggestions, navigation systems provide valuable information. For example, the map database of existing navigation systems contains various types of information (e.g. street/city names, points of interests, zip codes, km-ranges) that can be exploited by notification applications to describe affected areas/sub-areas etc.

Therefore, our *second aim* is to use the map database and the suggested routes to 1) *define location as context*, 2) *generate interests and publications*, and 3)*match subscriptions to publications*. For example, a vehicle can subscribe to receive notifications about "MyRoute", or concerning "Motorway M25", and a notification about an accident may concern "Junction A51" that might be a part of M25 or a part of the vehicle's route, and thus matched.

Therefore, our middleware architecture should interact with the vehicle's navigation system to enable the application developers and drivers to flexibly express interests according to the nearby map topology and to the current route of the vehicle.

3.3 Communication protocols

Our middleware primitives require the appropriate mechanisms to perform the instructed tasks. Therefore, we need to design the appropriate 1) routing and 2) dissemination algorithms.

Firstly, we need *delay-tolerant geographic routing algorithm* to deliver the information to/from infostations (the publisher). And secondly, we need a *notification dissemination protocols* that provide the functionality needed by the Publish/Subscribe communication primitives. By using the subscriptions and the navigation system we may further optimise the delivery ratio and the communication overhead of these protocols.

4. INITIAL MIDDLEWARE DESIGN

In this section we present an overview of our current design of the system.

4.1 Publish/Subscribe Primitives

From a publisher point of view, the information should be time and location stable. Therefore, the publisher needs to specify in which areas the notification should be disseminated and the time interval in which the notification is relevant. The publisher also creates a topic that contains the application-dependent keywords and the location of the areas that this notification concerns (we call them Points of Interest).

In order to clarify how application developers will interface with our protocol we define our communication primitives as:

notify(message,topic,publicationArea,T_{start},T_{end})

For example: the notification primitive which should be invoked by the publishing application is: [1]

notify(Works at Junction A51,
Type = Warning:Road works| PointOfInterest =
M25:Junction A51, 2km around Junction A50, 2pm,
5pm)

In this example, drivers that intend to use *Junction A51* of motorway M25 will be informed that there are road works, when driving close to *Junction A50* between 2pm and 5pm (so that they can exit).

Similarly, the navigation system can invoke the **subscribe**(topic) primitive

Subscription can be invoked automatically from the application (by the navigation system) or manually (by the user). For instance, the navigation system of a vehicle could subscribe to receive warnings about road segments in its calculated route:

subscribe(Type=Warning|PointOfInterest=MyRoute)

As before, the topic contains the type of the notification (any warning) and the area of interest (vehicle's route). The vehicle will later use it's navigation information (if available) to determine if it is interested or not.

Finally, the application can invoke the **unsubscribe**(topic) primitive to unsubscribe altogether.

4.2 Topic Context

The topic should accurately describe the context of the notification: when a notification is received our middleware has to determine if it is interested in it or not, according to its subscriptions and the information from the navigation system. And similarly, the navigation system applications and the drivers need flexible ways of describing the areas of interest.

The majority of notifications concern only a specific geographical area. However, it is not trivial to define a notification's point of interest. For example, a traffic jam warning can concern only a junction, a whole highway part (kilometre range), a city area, etc. A fuel-price advertisement may affect only vehicles driving through

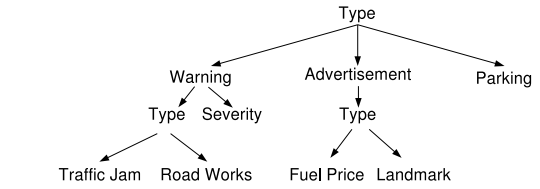
[1]For the sake of simplicity we use here a simplified version of the topic definition. More details of definition of topics will be given later in this section.

the shop's location. A parking spot may affect vehicles in a certain range from their destination.

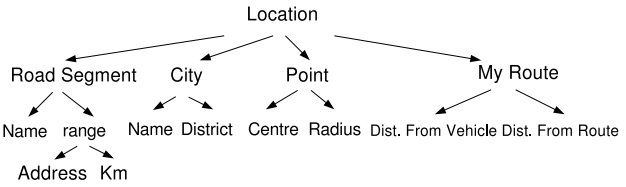
Our solution is to treat location as a context in order to enable developers to design flexible applications. In the next section we will examine how we can use the navigation system's map to map context in actual geographic locations.

Furthermore, vehicular applications may require additional custom keywords in the topic definition. Consequently, context needs to be organised and categorised. If we look at related work on context matching of traditional Publish/Subscribe systems, we observe that there are various ways of defining context. Some of them, employ a tree structure for the definition of the topic [7, 4]. Similar structures can be used to describe the notification topic and describe location.

Our middleware design provides the appropriate primitives to add/ remove attributes and to store them in appropriate XML files (e.g. `add (attribute, values, parent)`).



(a) Topic



(b) Area of interest

Figure 2: Examples of topic definition.

Figure 2 contains an example of the attributes that can be used to describe the topic and the location in our scenarios. In this example there are two trees with two root attributes (Type and Location). The applications can add more root attributes according to their needs. This is the minimum set of attributes to accurately describe a generic vehicular notification. Every attribute can have one ore more possible values. The topic can contain more than one values (or branches of the tree). Every value can have one or more sub-attributes, etc. Figure 2.a contains an example of the type attribute. A notification could be either a warning, an advertisement, or a free parking-spot etc. When the notification is, for example, a warning there are more sub-attributes available: type of warning, severity, etc and any of these attributes can have different values. For instance, a topic with Type={Warning(Type={Traffic Jam}, Severity ={High})} can be used to filter incoming notifications that concern warnings about severe traffic jams. Similarly a topic Type=Warning can be used to match any possible warning or Type={Warning(Severity = {High})} can only match severe warnings of any type.

By using these structures, we enable the application developers and users to define points of interests and areas of interest in a flexible manner. Figure 2.b illustrates an example where a location can be described as a kilometre range of a street, just a street name, city, or even just a point (x,y). As we are going to see, our middleware can match this information using the aid of the navigation system component.

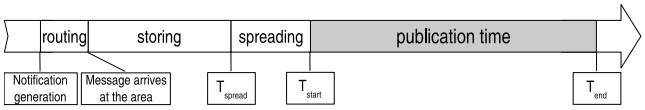


Figure 3: Timeline of events during publication of a notification.

4.3 Topic matching

Upon receiving a notification, our middleware has to decide if there is a subscription matching the notification's topic. More specifically, the vehicle has to respond as *1)Interested* or *2)Not Interested*. We can match simple Publish/Subscribe attributes (like the notification `type`) using existing matching protocols [7, 4].

Nevertheless, to perform location matching our middleware needs to consult the information from the navigation system component(e.g. street/city names, areas, points of interests, etc). This information enables us to treat location as context and makes matching a trivial process. For example, given a notification for a specific location (x,y), the navigation system can determine if this point is near the currently suggested route of the vehicle, a specific street or highway (given by name, name and address range, name and kilometre range), a city or a city area (given by name), a certain point of interest (train station, park, parking), or the current position of the vehicle. Similarly, the middleware can determine if a notification for a certain street (given by name) or highway segment (given by name and km-range), or a notification concerning the whole city, overlaps with its suggested route.

Therefore, a large number of subscriptions can be matched by our middleware according to the location and the destination of the vehicle, in order to determine if a received notification is relevant or not. This interaction with the navigation system provides the needed flexibility to perform context-based spatial matching and this is the main reason for including the navigation system to our middleware design.

4.4 Publish/Subscribe Semantics

In this section we will present an overview of the notification mechanisms that we are going to use in order to publish messages to subscribers in certain geographic areas (Figure 3):

Notification Generation: The notification is first generated by the publisher, the publisher defines the area of interest and the dissemination area. Furthermore, the publisher needs to generate a valid topic that, among others, includes *1)the type of the notification* and *2)the Point of Interest (POI) of the notification*. As we described before, our middleware provides flexible ways to define these parameters.

Notification Routing: The notification is then routed towards the dissemination area (as instructed by the middleware primitives) using both the infrastructure and the vehicle-to-vehicle communication. Our mechanisms decouple routing from publication by first routing the notification into the area and by starting the spreading later. If fixed infostations are available in the area and the notification comes from a private backbone network, the information is routed to the nearest infostation through standard backbone routing.

We have already designed the geographic routing algorithm [5] that is using the navigation system's suggested routes in order to perform opportunistic geographical routing. Packet carriers are selected using a utility function that considers the destination and the suggested route of the neighbours and the destination of the packet

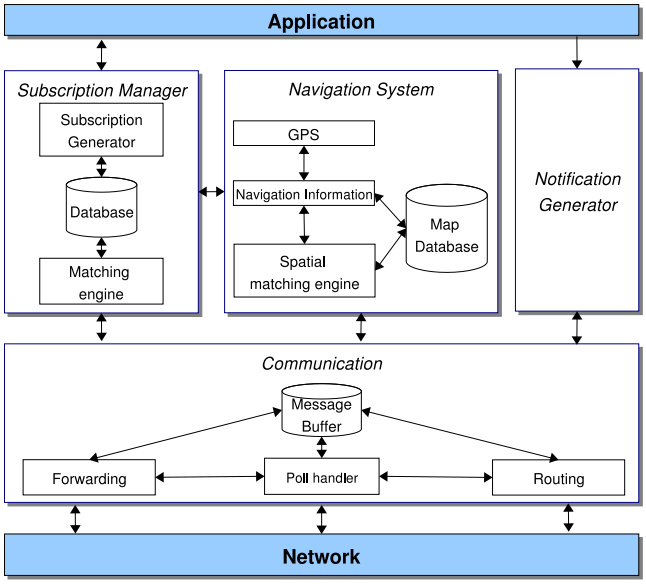


Figure 4: Middleware architecture.

(i.e. how close will the neighbours drive compared to the destination of the packet).

Notification Storing: Once the notification has been routed to the area, it is stored there (in an infostation, if available, or in a vehicle inside the publication area) until the start of the notification time. Further replication could be straightforwardly introduced for fault tolerance. The middleware is responsible for keeping the notification inside the notification area by routing it back inside in case the vehicle moves outside (with the aid of the navigation system).

Notification Spreading: Since the notification may be stored in the area before T_{start}, the publisher might allow the dissemination to start earlier in order to spread the notification and to adapt the notification algorithm to the network conditions. Notice that before T_{start} the notification is not delivered to the application layer.

Notification Publishing: Due to mobility, new subscribers in the area need to be informed. During the publication time, a mechanism of message broadcasting is employed. This step involves the dissemination of the notification in the publication area. To perform the dissemination, our middleware creates *a number of replicas* that become the notification broadcast points. Vehicles that carry these replicas are considered as *mobile infostations*. Our aim is to use the subscriptions in order to efficiently disseminate the notifications in areas where there is high interest.

We have also implemented the disseminate algorithm presented in [6]. This algorithm takes advantage of two key components: 1) the subscriptions (to disseminate only in areas where there are subscribers) and 2) the navigation system to extract information about possible locations of subscribers. The notification is then continuously propagated into the *affected areas* using some of the vehicles as mobile infostations.

4.5 STEPS Middleware Architecture

In this section we illustrate our framework. As you can see in Figure 4, the architecture is composed by a number of components that interact. The key components are 1) the subscription and notification components, that are used to set up filters of incoming notifications (and to filter incoming notifications that are only relevant), to aid the publishers to describe the affected areas, and to aid the communication component to perform the dissemination, 2) the navigation system that is needed to perform the dissemination (in areas with subscribers), routing (using the suggested routes) and to use location in the topics, and 3) the communication protocols that are responsible to route and deliver the notification to subscribers as instructed by the middleware primitives. We now examine these components in detail.

4.5.1 Application:

The application invokes the `notify()` primitive of the *Notification* component in order to create a publication. It invokes the `subscribe()`, `unsubscribe()` primitives from the *Subscription Handling* component to add or remove a subscription. The application is informed by the same component about a received notification.

4.5.2 Notification Generator:

This component provides the `notify()` primitive to the application. When `notify()` is invoked, this component calculates all the required parameters (e.g. metadata for subscription). Then, it forwards the notification over to the routing component.

4.5.3 Communication :

When a message from a network layer arrives to the communication component it is stored to its message buffer (so that forwarding/polling/routing subcomponents can access it). Furthermore, if it is a notification it is forwarded to the subscription component so that the application can be notified. This component is also responsible for routing and disseminating the notification inside the affected areas. It is composed by the following subcomponents:

- *Routing:* The geographic routing component is invoked every time a message in the *message buffer* is not inside the dissemination area (need to be routed back), or every time a route packet has not yet reached it's destination. It interacts with the polling component to find the best neighbour that is likely to deliver the packets to its destination. The routing component either keeps the packet or sends the message to the network layer to be transmitted to the appropriate neighbour. The algorithm can be found in [5].

- *Forwarding:* The forwarding component is responsible to create and maintain the appropriate number of mobile infostations. During the notification time, it is invoked periodically to broadcast the notifications to nearby subscribers. Before the actual broadcasts it consults the polling component to get a list of subscribed neighbours in order to control the spreading in the affected areas (the details of this algorithm are outside the scope of this paper and can be found in [6]).

- *Polling:* The polling component is responsible to handle poll requests from neighbours. It contacts the subscription and the navigation system in order to reply to the enquiring vehicles. Furthermore, it is responsible to initiate polls and to get the results to the forwarding and routing component.

- *Message Buffer:* The message buffer contains the replicas and the packets to be routed.

4.5.4 Network:

The network can be any type of wireless network that supports broadcasts to the neighbours (e.g., 802.11b, 802.11p). Messages can be delivered to/from the communication layer.

4.5.5 Navigation System:

The navigation system hold the navigation information of the vehicle (suggested route, estimated time, map database etc). Furthermore, we need to add a *spatial matching engine*. This spatial matching engine will be used to match the locations of the subscriptions and the incoming notification. The suggested route may be used for this calculation (match if the notification POI is on the suggested route). Furthermore, the spatial matching engine is able to calculate the required parameters by the routing algorithm. Finally the navigation system can provide information like speed, position, direction required by the algorithms of the forwarding and polling component.

4.5.6 Subscription Manager:

This component supplies the application with the calls to handle the node's subscriptions: `subscribe(topic)`, `unsubscribe (topic)`, `addAttribute (attribute, values, parent)`, `removeAttribute (id)`, etc. The *subscription generator* component manages these subscriptions in the subscription *database* with the aid of the navigation system.

The *matching engine* examines if there is a matching subscription when a message arrives from the *communication* component. If the topic contains geographical information (i.e. POI) it handles this information to the *spatial matching engine* of the navigation system. If there is a matching subscription it handles the message to the application component. Furthermore, it replies to the communication component as *interested, not-interested, received before*.

5. FUTURE WORK AND CONCLUSIONS

During the first one and a half year of our research we have designed the primitives and the routing and dissemination algorithms required to create our notification middleware. In this paper, we present an overview of the initial design of our Publish/Subscribe middleware.

Although we spent a lot of research time to design the communication algorithms, our middleware design is still in its early design stages. There are still interesting questions and challenges on how to improve our architecture and on how to design the interaction of all these components. The most important challenge is to provide the mechanisms (e.g. an API) to interact with existing satellite navigation systems in order to perfrom the spatial matching of the subscriptions and this is where we will focus our research now. Furthermore, we need to accurately define the XML structures that will be used to describe and match the topics.

Finally, we plan to evaluate our approach using simulation with realistic mobility traces (from ETH Zurich). We have already implemented parts of our protocols in OmNet++[11] and some of the initial results can be found in [5, 6]. Furthermore, we will implement and test a simple notification middleware using a small number of vehicles equipped with the appropriate hardware.

Acknowledgements: I would like to thank my supervisor, Cecilia Mascolo for guidance, helpfull suggestions and support. Additionally, I would like to thank Genaina Nunes Rodrigues for commends on an earlier draft. Furthermore, we would like to to acknowledge the support of the EPSRC through Project CREAM.

6. REFERENCES

[1] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM*, April 2006.

[2] V. Bychkovsky, K. Chen, M. Goraczko, H. Hu, B. Hull, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, and S. Madden. The cartel mobile sensor computing system. In *SenSys '06*, pages 383–384, New York, USA, 2006. ACM Press.

[3] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, 2001.

[4] Ludger Fiege, Felix C. Gärtner, Oliver Kasten, and Andreas Zeidler. Supporting Mobility in Content-Based Publish/Subscribe Middleware. In *Middleware*, pages 103–122, 2003.

[5] Ilias Leontiadis and Cecilia Mascolo. GeOpps: Opportunistic Geographical Routing for Vehicular Networks. In *Proceedings of the IEEE Workshop on Autonomic and Opportunistic Communications. (Colocated with WOWMOM07)*, Helsinki, Finland, June 2007. IEEE Press.

[6] Ilias Leontiadis and Cecilia Mascolo. Opportunistic Spatio-Temporal Dissemination System for Vehicular Networks. In *In Proceedings of the First International Workshop on Mobile Opportunistic Networking (ACM/SIGMOBILE MobiOpp 2007). Colocated with Mobisys07*, Puerto Rico, USA, June 2007.

[7] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad-Hoc Networks. In *1st International Workshop on Distributed Event-Based Systems (DEBS '02), Vienna, Austria*, 2002.

[8] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru internet environments. *INFOCOM 2005*, 3:1849–1862 vol. 3, 2005.

[9] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[10] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling a three-tier architecture for sparse sensor networks. In *Sensor Network Protocols and Applications*, 2003.

[11] András Varga. The OMNET++ Discrete Event Simulation System. *Proceedings of the European Simulation Multiconference (ESM'2001)*, 2001.

[12] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. pages 187–198, 2004.

[13] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04*, pages 187–198, New York, NY, USA, 2004. ACM Press.