VSSA: A Service-oriented Vehicular Social-Networking Platform for Transportation Efficiency

Xiping Hu
Dept. Electrical and Computer Engineering, The University of British Columbia
Vancouver, Canada V6T 1Z4
xipingh@ece.ubc.ca

Weihong Wang
Institute of Information and Technology, Hebei University of Economics and Business
Heibei, China 050061
wangwhs@gmail.com

Victor C.M. Leung
Dept. Electrical and Computer Engineering, The University of British Columbia
Vancouver, Canada V6T 1Z4
vleung@ece.ubc.ca

ABSTRACT
This paper proposes a novel service-oriented vehicular social networking platform called VSSA for transportation efficiency. VSSA consists of two layers: service layer and application layer. The service layer is implemented based on the service-oriented architecture (SOA) to ease application developments. By extending and composing the web services in this layer, application developers can efficiently and flexibly develop different new functions or applications for transportation scenarios. The application layer is oriented to vehicular users and it integrates with functions of mobile social networks. With dynamic and automatic service collaboration support, it enables people to easily collaborate and help each other through their mobile devices in transportation situations. Moreover, VSSA provides a context-awareness mechanism to support automatically and intelligently predicting the potential incoming traffic congestions.

Categories and Subject Descriptors
C.2.1 [**Computer Systems Organization**]: Network Architecture and Design- Network communications.

General Terms
Algorithms, Design, Economics, Human Factors, Languages, Theory.

Keywords
Vehicular social networks, SOA, service, application development.

1. INTRODUCTION
Every day, a large number of people in urban areas would spend hours traveling along the same routes at about the same time on their commute to and from work. Their travel patterns are highly predicable and regular. Consequently, there is an opportunity to form recurring virtual mobile communication networks and communities between these travelers or their vehicles [1]. Vehicular social networking (VSN) is social net- working that occurs in a virtual community of vehicles and their drivers and passengers, where one or more individuals in such an environment have similar interests, objectives or commonalities, and have the capability to interact with each other. Different from conventional mobile social networks, which participants are humans who converse with one another using mobile phones, the participants of VSNs are heterogeneous, which include vehicles, devices onboard vehicles, as well as drivers and passengers. Thus, three types of relationships are found in VSNs: (i) between human and human; (ii) between human and machine; and (iii) between machine and machine. Meanwhile, research has shown that knowledge of nodes' social interactions can help improve the performance of mobile systems [2, 3], and the cooperative intelligent transportation system can potentially improve traffic efficiency [4-7]. For example, the applications over VSN can support the users' collaboration and provide them with up-to-date traffic information enabling them to improve the traffic efficiency and driving behavior;

In order to develop a VSN system, we should first establish a vehicular ad-hoc networking (VANET) connecting the hardware devices onboard vehicles and users, and determine its specifications. The second step is to develop a software platform on which to develop and install different applications that function in the VSNs. As the advanced mobile devices such as smart phones now have the capability of forming opportunistic networks in an ad-hoc manner through WiFi Direct or Bluetooth connectivity [8], and people normally carry with them in their vehicles every day. Thus, the users can use their mobile devices on board vehicles to establish a VANET independently of a fixed infrastructure in an inexpensively and conveniently way [9].

Regarding the software platform for VSNs, as the dynamism and opportunism features of VANETs, thus the situations and service requirement of VSN applications for transportation efficiency in vehicular environments are complicated. For instance, the situations of different street/road could be complicated with different geography and traffic density, and the traffic density is also dynamic; different people may concern about different things simultaneously, some may more concern about the route for shortest time, some more concern about the route for fewest fuel consumption, while some is looking for the nearest parking place (but normally cannot make sure currently there are available positions in such places or not) and so on. However, most of the current software applications/platforms based on VANET for transportation efficiency are normally for specific applications and could hardly meet the diverse service requirements as mentioned above [1, 10, 11]. Consequently, it is crucial to provide a software platform with well extensibility support for developing diverse VSN applications for transportation efficiency, and enabling the developed applications could support collaboration among the vehicular users intelligently.

DIVANet'12, October 21–22, 2012, Paphos, Cyprus.

Service-Oriented Architecture (SOA) [12] is a set of specifications and methodologies for designing and developing software applications/services in the form of interoperable service components. Services in SOA are loosely coupled, have unified specifications for service interactions but do not depend on system-specific languages. This flexible architecture can effectively support dynamic service composition and customization for individual application/user. Thus, application developers can flexibly and efficiently combine different services in SOA to generate many kinds of new applications according to their diverse service requirements for transportation efficiency. Even though the developed applications are different, they still have the capabilities to collaborate (i.e., sharing data) with each other across different operating systems in their run-time.

Much research work based on SOA for transportation efficiency has been done. A SOA based communication architecture for transportation cooperative systems in Europe [13] was designed. Using the SOA concept, an integrated system for delivering real-time travel and traffic information on mobile navigation devices was presented [14]. Based on SOA and multi-agent system, a distributed traffic Simulation system was proposed [15]. However, all of the current SOA based solutions for transportations employ SOA clients in mobile devices [16-18] that access via the Internet or roadside infrastructures web services deployed on specific servers; thus SOA based mobile applications are not capable of running in the mobile devices independently and directly collaborating with each other over ad-hoc manner. As the coverage regions of vehicular applications are usually very wide, and due to the high speed of vehicles, the requirement for their communication delay is strict, thus the cost would be pretty high and hardly affordable to deploy enough roadside infrastructures and centre SOA servers in practical situations. With the improvements of computing and communication abilities of mobile devices, the conditions are becoming mature to support the deployment of light weigh SOA servers and web services on the mobile platforms.

In this paper, we propose and design VSSA, a new service-oriented VSN platform for applications supporting transportation. Compared to existing SOA-based systems for transportation efficiency, VSSA fully supports SOA based mobile applications working independently without roadside infrastructures in popular mobile platforms (e.g., Android) in an ad-hoc manner with low resource overhead in mobile devices. VSSA aims to support application developers to efficiently develop diverse and useful mobile applications that can used for supporting transportation, and to enable people in transportation environment to effectively collaborate with others in their vicinity through their mobile devices, so as to improve transportation efficiency.

The organization of the rest of the paper is as follows. Section 2 introduces the overall design of the VSSA system, presents its unique services, functions and mechanism. Section 3 analyzes the development challenges and implementation strategies of the VSSA. Section 4 shows experiment results to demonstrate and evaluate the proposed VSSA system. Section 5 concludes the paper with remarks on the prospects of the developed VSSA for future applications.

2. SYSTEM DESIGN

The overall architecture of VSN used in transportation is shown in Figure 1. VSSA as the software platform of VSN, its system architecture is shown in Figure 2. As mentioned in Section 1, VSSA aims to provide a universal platform to support the diverse service requirements of users in vehicular environments, so as to

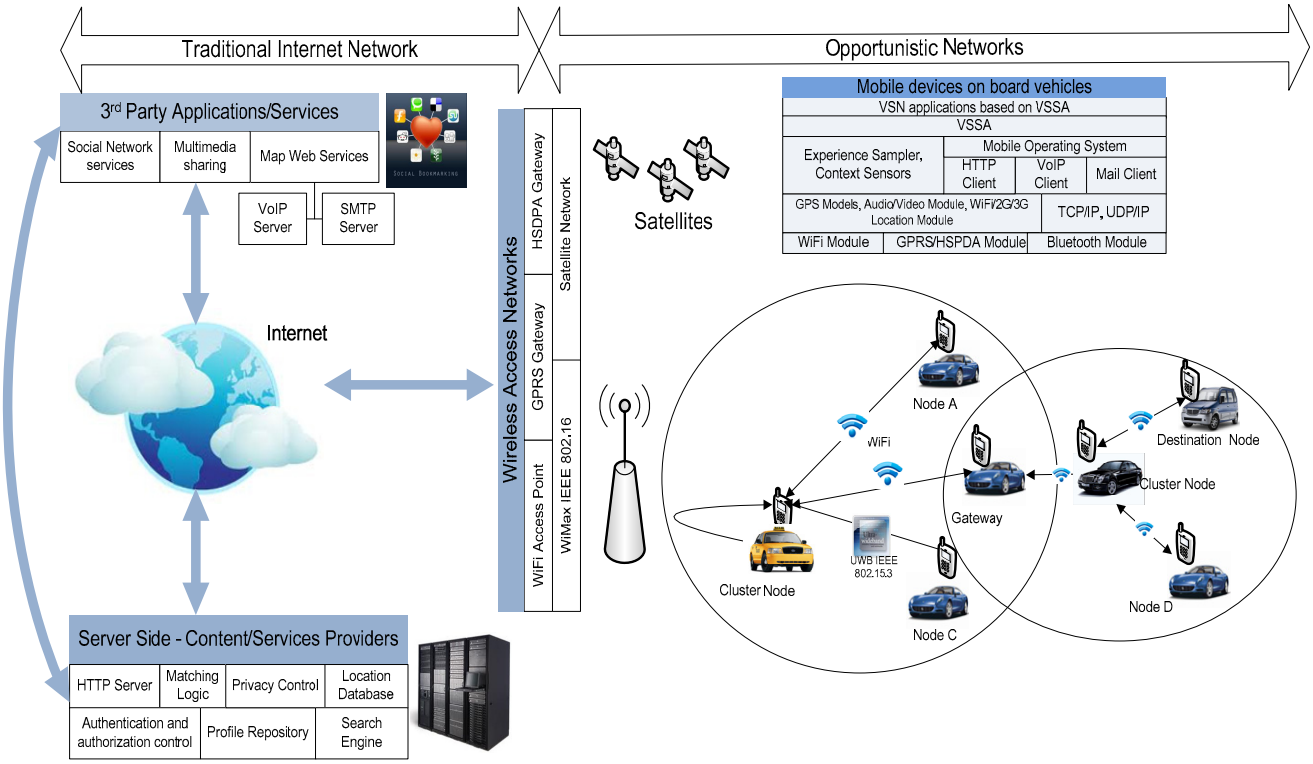


Figure 1. The overall architecture of VSN in transportation.

improve the transportation efficiency. It consists of two components: the application layer and the service layer. The application layer is oriented to end users who can directly use the VSSA system in their mobile devices through VSSA's user interface; while the service layer is transparent to end users and oriented towards supporting efficient development of applications for the improvement of transportation efficiency. Using the service-oriented programming model, software developers can easily and efficiently implement diverse mobile applications for transportation scenarios.

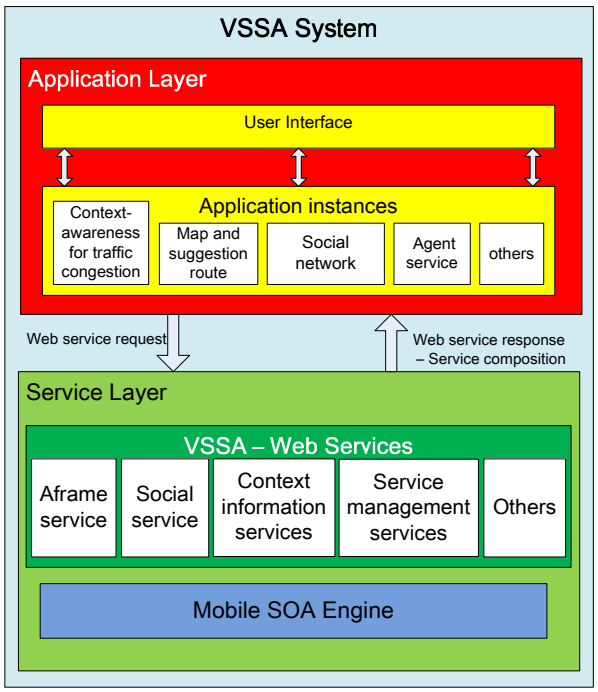


Figure 2. The VSSA platform.

2.1 Service layer

As discussed in Section 1, due to the different geography of roads/streets, diverse service requirements of drivers and passenger, and quickly changing environments of vehicular networks, it is crucial to provide an application development environment that enables mobile applications to operate over networks formed in an ad-hoc manner, and supports collaborations between these applications in their run-time. The web service layer is designed to provide basic service components and mechanism that develophers can incorporate into applications for improving transportation efficiency. All the services in this layer are SOA based web services. Thus application developers can take full advantage of SOA to efficiently and flexibly extend the web services, or integrate the web services in this layer to generate different application services. Moreover, developers can also design mechanisms or algorithms for service compositions in this layer, so that multiple applications developed on VSSA can automatically and intelligently access, use or collaborate with the running web services shared among the mobile devices at run-time. In the current version of VSSA, there are mainly three types of web services in this layer: Aframe [19, 20] services, social service, context information services, and service management services.

2.1.1 Aframe service

Aframe is an agent-based application programming framework for mobile wireless ad-hoc network, which we have designed and implemented previously. By integrating AmbientTalk (a new programming language for mobile ad-hoc networks) [21] and software agent technique in a multi-layer framework, Aframe can simplify and hide the complexity of handling different connectivity status of mobile wireless ad-hoc networks, various services requirements and diverse mobile devices. It supports collaborations among multiple agents working in multiple devices simultaneously, such as supporting multiple mobile agents to dynamically and self-adaptively execute different applications around wireless ad-hoc networks with the help from resident agents. For the VSSA system, we implement the whole Aframe framework as a web service in this layer. Further, as Aframe also consists of several services of its own, these Aframe services can be seen as sub-services in this web service layer of VSSA. There are mainly two kinds of services inside Aframe: application specific services and framework services, where application specific services can be developed and extended by application developers according to their specific service requirements, while framework services are generic services as follow:

ID name service: It generates a universally unique identifier (UUID) as ID name for each mobile device.

Data fetching and storage service: It provides shared and non-shared data services to mobile agents through resident service agents.

Network status service: It provides real-time information of currently connected mobile devices in the local mobile wireless ad-hoc networks.

Migration service: Using this service, mobile agents can migrate around the mobile devices in the underlying mobile wireless networks.

Current time service: It provides local timing information in a mobile device.

Deployment service: Using this service, resources and specific services can be deployed and installed around the local mobile devices dynamically by mobile agents.

Moreover, in order to support the functionality of VSSA, we also implement two application specific services inside Aframe: Location service, which is used to get the location information of mobile devices; and Map service that is used to invoke the Google Map APIs in mobile device and sign the location information in the digital map. In addition, since the current mobile version of Google Map supports pre-store the digital map and offline map wraps which enable it to work independently from Internet, thus based on these, the Map service of VSSA could work independently from Internet. Further, taking advantages of SOA, the Map service can collaborate with each other among the mobile devices in their run-time, such as it supports sharing and compositing the offline map wraps of Google Map when the information of digital map is not enough in individual's mobile device.

2.1.2 Social services

This type of service makes use of the open APIs of popular MSN applications like Facebook and Google+, and encapsulates them as web services in VSSA, so as to leverage their functions to VSSA and enable them to work in an ad-hoc manner but not

limited by Internet. Application developers can extend, invoke or composite this type of service with other web services in this layer. For example, developers can composite these social services with network services of Aframe, then it can support interactions between people through conventional MSN application over VANETs. In other words, it leverages the advantages of the MSN applications in VSSA, since most people are familiar with their operations, and they have already provided mature and sufficient ways of social interaction. The current version of VSSA mainly consists of three services as follows:

Social interaction service: It is used to support the information interaction, such as sending messages between the users.

Information publication service: It is used to support publication of information to the nearby mobile devices connected to the local wireless networks.

2.1.3 Context information services

This type of service consists of two parts: Part A is about the source of context information, which can be used to provide the source of context information; while Part B is a mechanism, it is used to process the context information provided by the former part and other services in the service layer, and predict the abnormal transportation situations, such as the potential traffic congestions, so as to support the improvement of transportation efficiency.

2.1.3.1 Part A

Social contacts service: Normally, the popular MSN applications can synchronize their users' contact information to the address books of the smart phones. This service makes use of the related open APIs of the MSN applications and smart phones, and integrates all the contact information as a service that can provide the contact information of MSN users.

User history service: This service also makes used the APIs provide by the MSN applications and smart phones, than can provide the operation history of users when they are traveling, such as the history of traffic routes where the users usually go through.

OBD service: It is used to support the connection with the OBD devices on board vehicles, and fetch the related data from OBD devices, such as the current speed and fuel consumption of vehicles.

Weather service: It makes used the APIs provide by third party applications for weather broadcast, so as to provide the update to date weather information. For instance, it could be used to alert the users that their destinations are in horrible weather condition when they are travelling.

2.1.3.2 Part B

Context collection service: This service collects all kinds of context information coming from multiple information sources, such as social contacts service, user history service, OBD service and so on. Meanwhile, it can collect some context information coming from other services in the service layer and application layer as well. Thus, the core contexts impacting traffic congestion were extracted based on the collected context information, and storage in this service. The core contexts mainly include the road state, vehicular velocity, weather and the model of vehicles etc.

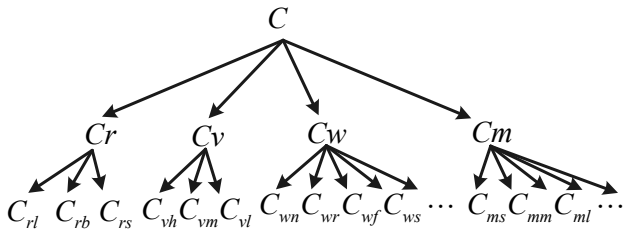
Cr denotes the road state context, then Cr can be defined as a set as follow: $Cr = \{C_{rl}, C_{rb}, C_{rs}\}$, where C_{rl} denotes the count of lanes, C_{rb} denotes the roadblock information in the road, C_{rs} denotes the road smooth degree information.

Cv denotes the vehicular velocity context, then Cv can be defined as a set as follow: $Cv = \{C_{vh}, C_{vm}, C_{vl}\}$, where C_{vh} denotes the high velocity, C_{vm} denotes the middle velocity, C_{vl} denotes low velocity.

Cw denotes the weather context, then Cw can be defined as a set as follow: $Cw = \{C_{wn}, C_{wr}, C_{wf}, C_{ws}, \cdots\}$, where C_{wn} denotes the normal sunny day, C_{wr} denotes the rainny day, C_{wf} denotes the fog day, C_{ws} denotes the snowy day, and so on. In another way, if Cw denotes the weather which doesn't have negative influence on the traffic, then $\neg Cw$ denotes the weather which has the negative influence on the traffic. Thus, $\neg Cw$ is a nonvoid subset including C_{wr}, C_{wf}, C_{ws} etc.

Cm denotes the vehicular velocity context, then Cm can be defined as a set as follow: $Cm = \{C_{ms}, C_{mm}, C_{ml}, \cdots\}$, where C_{ms} denotes the small vehicle, C_{mm} denotes the middle vehicle, C_{ml} denotes large vehicle.

Thus, the core contexts C in context collection service can be defined as a set $C = \{Cr, Cv, Cw, Cm\}$, and the contexts information tree can be constructed as follows.



In the traffic situation, the contexts may be not discrete event data, such as vehicle velocity. In addition, some contexts information may be only qualitative description available. Therefore, the quantization and semantic abstraction of context information should be done before inference, and each core context and element of set can be mapped a value or semantic class.

Context rules service: This service provides the rules of context awareness and reason for traffic congestion, such as the decision condition if the traffic congestion state happened under some specific Cr, Cv, Cw and Cm.

Context inference service: This service provides the context inference based on contexts and context rules. For example, some threshold values can be gotten by history context statistics and application cases. And based on it, some conclusions can be done

that the traffic congestion don't arise if the values of Cr, Cv, Cw and Cm are less than the relevant threshold values. These can be described as follow.

Assuming that δr, δv, δw, and δm are the threshold values of Cr, Cv, Cw, and Cm respectively, then some inference as follow can be expressed :

TC == False if $f(Cr) \le \delta r \wedge f(Cv) \le \delta v \wedge f(Cw) \le \delta w \wedge f(Cm) \le \delta m$

TC==True if $f(Cr) > \delta r \wedge f(Cv) > \delta v \wedge f(Cw) > \delta w \wedge f(Cm) > \delta m$

TC = False if $f(Cr) \le \delta r \vee f(Cv) \le \delta v \vee f(Cw) \le \delta w \vee f(Cm) \le \delta m$

TC=True if $f(Cr) > \delta r \vee f(Cv > \delta v) \vee Cw > \delta w \vee Cm > \delta m$

In the formula, *TC ==False* means the traffic congestion did not arise under the states of $f(Cr) \le \delta r \wedge f(Cv) \le \delta v \wedge f(Cw) \le \delta w \wedge f(Cm) \le \delta m$. And *TC==True* means the traffic congestion arose under $f(Cr) > \delta r \wedge f(Cv) > \delta v \wedge f(Cw) > \delta w \wedge f(Cm) > \delta m$. Also, the double equal sign in the formula expresses the traffic congestion happened certainly. Instead, the single equal sign expresses that the traffic congestion happen probable. In the latter, more context conditions will be added to the inference process for getting some specific decision.

Context inference service is the foundation of context-awareness for traffic congestion in application layer. The functions of prediction and warning can be provided based on the results of context inference.

2.1.4 Service management services

This type of service is used to manage all the available web services of VSSA. Application developers can also design new mechanisms or extensions based on these web services before they develop concrete applications, and it contains the following services:

Service list service: It is used to fetch the current available web services in the local mobile device and the underlying VANETs.

Registration service: It is used to register new web services that are extended to the VSSA system during run-time.

Monitor and control service: It is used to monitor the status of all the web services of VSSA

Service transfer and adaption service: It is used to transfer and deploy the web services of VSSA among the mobile devices, and support the replacement of the web services in their run-time.

Schedule and coordinate service: It is used to configure the current web services and support the collaboration among the web services of VSSA.

2.2 Application layer

Every application instance in this layer is implemented through invoking or compositing the web services in the lower layer as mentioned above. The application instance can work independently, and it is oriented towards the realization of concrete applications. Application developers only need to develop the user interface with the related application instance in this layer, through which the end users of VSSA can directly use the corresponding function. Moreover, since each function in this layer is implemented through the compositing of the underlying web services, thus users could also collaborate with each other through service collaboration. Application developers could also implement additional functions here according to their practical requirements. Currently, there are mainly four application instances implemented in this layer to support the functionality of VSSA: Context-awareness for traffic congestion, Map and suggestion route, social network, and agent service.

Context-awareness for traffic congestion: This function is based on the context information service. Based on the context information provided by services (e.g., map, location etc.) in VSSA and devices aboard vehicles (e.g., OBD), it can predict the potential incoming traffic congestion situation to users. Meanwhile, users can also share the information about this to the vicinal vehicles automatically through the software agent, so as to help each other in vehicular environments.

Map and suggestion route: This function based on the network service and map service mentioned in service layer, it can help the users find out the real-time traffic information nearby through the Google Map. Meanwhile, it can support the users figure out the ideal route, such as which route is the shortest to their destinations, and which route could be the fastest.

Social network: This function is based on the social network service mentioned above. Similar to the use of conventional social networking applications, users can interact with the nearby people when they are found, sending text messages with photos, publishing information among people whose mobile devices are connected to the local VANETs or Internet, so as to collaborate with each other in transportation situations. For instance, users can find out the nearest available parking place with the help from other users through publishing the related questions to the local VSN. In addition, users of VSSA can also predefine a message that would be sent out automatically through the software agent when the user is busy with driving while somebody calling him.

Agent service: This function is used to support automatically mode as people may usually busy with driving and can hardly do the related operation in their mobile devices, such as the example mentioned above.

3. IMPLEMENTATION STRATEGIES

As indicated in Section 2, in order to implement the VSSA system, the major task is to implement its web service layer. Currently, the web services enabling most SOA based mobile applications are deployed on centralized servers while the mobile devices only run the SOA clients, which normally interact with the web services using Simple Object Access Protocol (SOAP). However, the web service interactions among the mobile devices need to rely on the central servers, which may need to be deployed as roadside infrastructures that need relative high cost. Therefore, we adopt a new SOA design style - Representational State Transfer (REST) [22] to implement the web service layer of VSSA. Comparing to SOAP, the REST based SOA has several advantages in mobile platforms, including being: (i) more light weight with lower hardware requirements than SOAP based SOA servers that could hardly run on the mobile devices; and (ii) less power consuming than SOAP based SOA. In the rest of this section, we introduce

the implementation architecture and process flow of the mobile SOA and related web services in the VSSA system.

3.1 The overall implementation of mobile SOA in VSSA

The overall implementation architecture of mobile SOA in VSSA is shown in Figure 3. At the bottom of the architecture is the mobile SOA engine, which is the SOA server that runs on the mobile device's operating system and provides the basic execution environment. This SOA engine is implemented using I-Jetty [23], an open source web server based on Android. On top of the mobile SOA engine are the Java class libraries, which mainly consist of the encapsulated Aframe components and other Java class libraries, such as Aframe's communication module. This module is implemented in AmbientTalk to support VANET communication, and the resident agent module to support the service invocation with mobile agents. In, addition, VSSA also supports extend new VANET routing protocols according to practical requirements, but not only the default routing protocol provided by AmbientTalk. All of the web services based on the REST style are deployed in the mobile SOA framework and supported by the underlying Java class libraries. Finally, external to the mobile SOA are the mobile clients implemented to enable specific applications. VSSA uses the mobile agent implemented in Aframe as the default mobile client. However, application developers can also develop other types of mobile clients that are not necessarily agent based. One requirement here is that the implementation of a mobile client needs to correspond to the resource operation of the underlying REST style based web services, and their mapping relation is implemented through the configuration files in the mobile SOA framework.

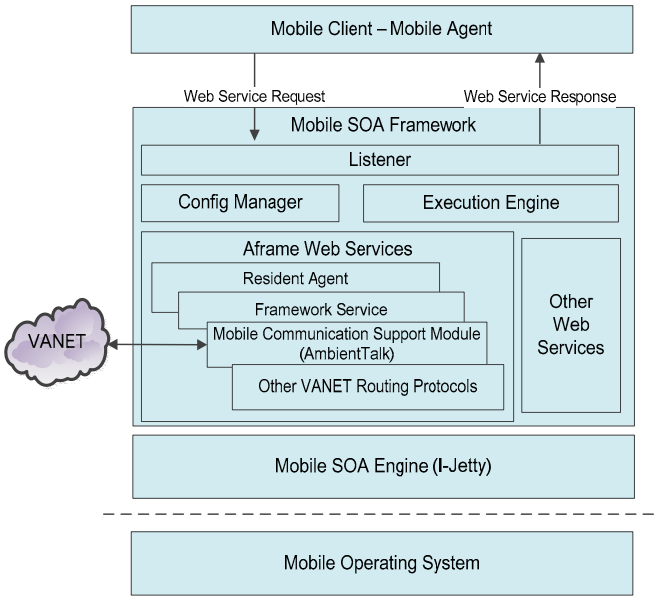


Figure 3. The overall implementation architecture of mobile SOA in VSSA.

The process flow of mobile SOA in VSSA is as shown in Figure 4. During run-time, the mobile SOA framework will firstly receive the web service request from the mobile client, and then analyze the request Uniform Resource Locator (URL) and the related parameters encapsulated by HTTP, so as to determine the specific Java class to invoke the corresponding web services based on the configuration files. Finally, after the operation of the related web services, the mobile SOA framework returns the results to client in the form of REST style data through HTTP. One advantage of this architecture design is that the application developers do not need to be concerned with issues of mapping relation about specific URL to corresponding service resources, but can focus on the development of the application itself.

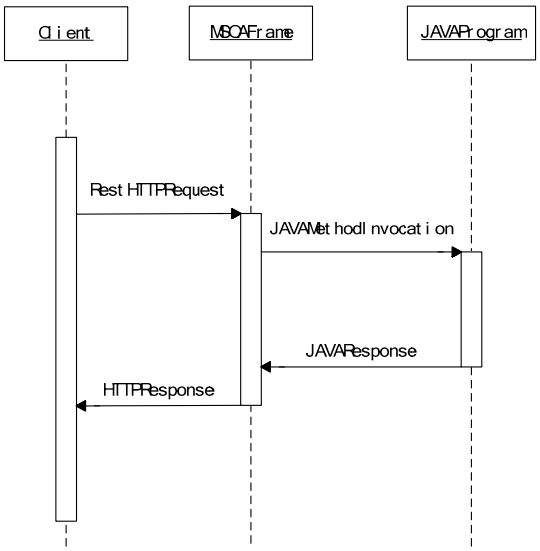


Figure 4. The process flow of mobile SOA in VSSA.

3.2 Implementation of the Aframe web service

As the Aframe service plays an important role in the web service of VSSA, and the implementation methodologies of other web services are similar to the Aframe service, this section focuses on the introduction of the Aframe service implementation in VSSA.

Different from the traditional SOA design concept like SOAP, which is process-oriented, the REST based SOA is resource-oriented, such that every URL in HTTP corresponds to a specific resource type, and the operations to the resource like get, create, update and delete correspond to the method GET, POST, PUT and DELETE of HTTP in REST. Therefore, as a first step the related URL of every key resource of Aframe in VSSA needs to be specified. For instance, if we consider the mobile VANET communication support module - AmbientTalk of Aframe to be a resource, then we can assign an URL to this resource like /rest/ambienttalk, and it will create a new ad-hoc network based communication resource to the mobile client when the client sent a POST request to this URL. In the second stage, the components of Aframe need to be encapsulated in terms of the Java operations, and finally a configuration file needs to be designed to map the implementation method from URL to Java operations. An example of a configuration file for the AmbientTalk module is shown in Table 1.

Here we use an example to illustrate how the service of Aframe works in the mobile SOA framework. As mentioned in Section 2.1, a location service exists as an application specific service in Aframe. In order to implement this service as a web service in VSSA, firstly we can configure the resource type of this service as

Global Positioning System (GPS) information with corresponding URL: /rest/GPS, and then encapsulate the get method of the GPS information in the executeService() method of GPSService class, and register this method as the type of application specific service in resident agent.

Table 1. Configuration file for AmbientTalk of Aframe in VSSA

URL	HTTP method	Java class	Operation
/rest/ambientalk	GET	org.ubc.aframe.AmbientTalk	get AmbientTalk ()
/rest/ambientalk	POST	org.ubc.aframe.AmbientTalk	create AmbientTalk ()
/rest/ambientalk	PUT	org.ubc.aframe.AmbientTalk	Update AmbientTalk ()
/rest/ambientalk	DELETE	org.ubc.aframe.AmbientTalk	Remove AmbientTalk ()

The process flow of this service is shown in Figure 5. Firstly, the mobile agent as a client will send the Get request to URL: /rest/GPS, and then the mobile SOA framework will invoke the executeService() method in resident agent based on the configuration file when it receives the request. Secondly, the resident agent will find out the corresponding service - GPSService based on the parameters of the request, and then invoke this service and return the result to mobile SOA framework. Finally, the framework will encapsulate the result as HTTP response and return it to the mobile agent.

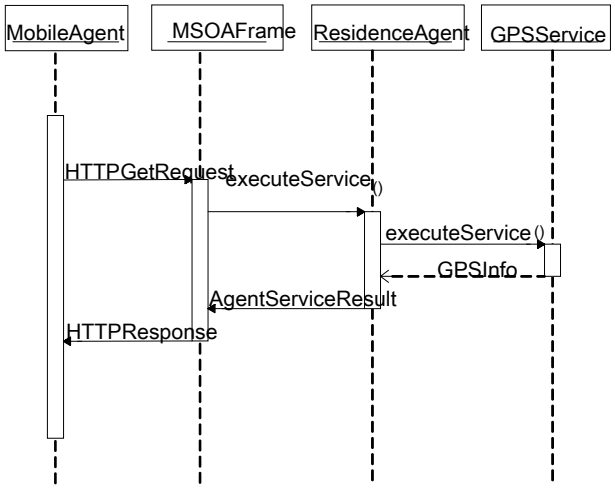


Figure 5. The process flow of location service

The process flow of mobile SOA in VSSA is as shown in Figure 4. During run-time, the mobile SOA framework will firstly receive the web service request from the mobile client, and then analyze the request Uniform Resource Locator (URL) and the related parameters encapsulated by HTTP, so as to determine the specific Java class to invoke the corresponding web services based on the configuration files. Finally, after the operation of the related web services, the mobile SOA framework returns the results to client in the form of REST style data through HTTP. One advantage of this architecture design is that the application developers do not need to be concerned with issues of mapping relation about specific URL to corresponding service resources, but can focus on the development of the application itself.

4. EXPERIMENT

In this section, we conducted an experiment to verify the reliability of the VSSA system and evaluate its ability by sending MSN messages over VANETs.

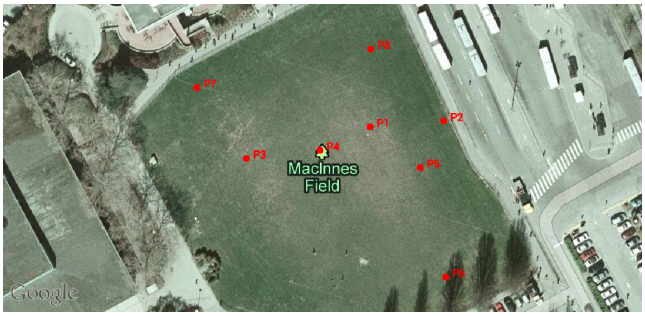


Figure 5. The map area of the experiment

The experiment consists of eight persons each carrying an Android device (six Android phones and two Android tablets) equipped with 802.11n wireless network module and running VSSA to set up a local VANET. The mobile devices normally have an ideal communication range of 500 meters when there is no obstacle. Each person carries a mobile device denoted as node Mx and moves in an area of about 400*400 m2 as shown in the map in Figure 5. Consequently, their message interactions are in the communication coverage factor of $8*5002*\pi /4002 \approx 39.26$. As shown in Figure 5 and Table 2, there are eight predefined positions, and each person sends MSN messages every 50s through VSSA to 2− 3 destination nodes when he/she moves from one position to another at a normal speed of 20km/h. The duration of this experiment last for 15 minutes, and it was found that all sent messages were received successfully.

Table 2. Experimental results

Node	Position	Message destination	Success
M_1	P_1, P_3	M_2, M_3, M_5	Yes
M_2	P_2, P_4	M_1, M_3	Yes
M_3	P_7, P_8	M_4, M_7	Yes
M_4	P_5, P_6	M_1, M_2, M_8	Yes
M_5	P_2, P_3	M_6, M_7, M_8	Yes
M_6	P_4, P_5	M_4, M_6	Yes
M_7	P_6, P_7	M_1, M_3, M_5	Yes
M_8	P_5, P_8	M_2, M_4, M_6	Yes

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed VSSA, a service-oriented VSN platform for transportation efficiency, and presented its implementation. By integrating SOA and MSN techniques, as well as multi-agent framework Aframe, VSSA can simplify and hide the complexity of handing different communication status, varying service requirements of mobile applications for transportation. To application developers, VSSA provides a high level software platform to support the efficient and flexible development of diverse and intelligent mobile applications with extensibility support. VSSA provides an easy and convenient mobile application with multiple functions to end users, and supports dynamic web service collaboration at run-time, enabling

the people in vehicular environment to easily collaborate with each through conventional ways of social networking which they are already familiar with. To the best of our knowledge, VSSA is the first solution that fully supports SOA based mobile applications working independently in mobile platforms. Moreover, a context-awareness mechanism has been designed to support automatically and intelligently predicting the potential incoming traffic congestions.

There are a number of challenges that need to be addressed in further development of VSSA systems. Although VSSA has integrated the routing protocols (provided by AmbientTalk) for VANETs, but the situations are usually complicated in vehicular environments and the requirement of time delay is usually very high, thus more efficient routing protocols that can be used to support VSSA working upon VANETs needs be investigated. As the SOA based architecture designs for vehicular applications are normally based on desktop platforms with central servers that do not work independently as VSSA on mobile platforms, the conventional evaluation platforms, methodologies for evaluating SOA based systems may not be suitable for the evaluation of VSSA. Therefore, a new methodology for the evaluation of mobile platform based SOA systems like VSSA is needed.

6. ACKNOWLEDGMENTS

This work is support in part by the NSERC DIVA Strategic Research Network.

7. REFERENCES

[1] S. Smaldone, L. Han, P. Shankar, and L. Iftode. 2008. RoadSpeak: Enabling Voice Chat on Roadways using Vehicular Social Networks. In Proceedings of the First International Workshop on Social Network Systems

[2] C. Boldrini, M. Conti, and A. Passarella. 2007. Impact of Social Mobility on Routing Protocols for Opportunistic Networks. In Proc. of AOC Workshop

[3] A. Miklas, K. Gollu, K. Chan, S. Saroiu, K. Gummadi, and E. Lara. 2007. Exploiting Social Interactions in Mobile Systems. In Proc. of Ubicomp

[4] L. Figueiredo, I. Jesus, J. Machado, J. Ferreira, J. Carvalho. 2001. Towards the development of intelligent transportation systems. IEEE Intelligent Transportation Systems. Page: 1206–1211. DOI:10.1109/ITSC.2001.948835.

[5] H. Hartenstein, KP. Laberteaux. 2008. A tutorial survey on vehicular ad hoc networks. IEEE Communications Magazine 2008; 46(6): 164–171. DOI: 10.1109/ MCOM.2008.4539481.

[6] Institute of Electrical and Electronics Engineers. IEEE Standard for Information Technology – Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments 2010. DOI: 10.1109/IEEESTD.2010.5514475.

[7] K. Katsaros, R. Kernchen, M. Dianati, D. Rieck, and C. Zinoviou. 2011. Application of vehicular communications for improving the efficiency of traffic in urban areas. Wireless Communications and Mobile Computing, Special Issue: Special issue on the selected papers of IWCMC

[8] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao. 2009. Understanding and capturing people's privacy policies in a mobile social networking application. Personal Ubiquitous Computing, Vol. 13, No. 6. , pp. 401-412, doi:10.1007/s00779-008-0214-3

[9] H. Schumacher, C. Priemer, and E.N. Slottke. 2009. A simulation study of traffic efficiency improvement based on Car-to-X communication. VANET '09 Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking. Doi: 10.1145/1614269.1614274

[10] T. Kitani, T. Shinkawa, N. Shibata, K. Yasumoto, M. Ito, and T. Higashino. 2008. Efficient VANET-Based Traffic Information Sharing using Buses on Regular Routes. IEEE Vehicular Technology Conference, VTC Spring 2008. Page(s): 3031 – 3036.

[11] E. Schoch, F. Kargl, and M. Weber. 2008. Communication patterns in VANETs. IEEE Communications Magazine. Volume: 46 , Issue: 11. Page(s): 119 – 125.

[12] T. Er. 2005. Service-Oriented Architecture: Concepts, Technology, and Design. Book Service-Oriented Architecture: Concepts, Technology, and Design Prentice Hall PTR Upper Saddle River, NJ, USA ISBN: 0131858580.

[13] T. Kosch, I. Kulp, M. Bechler, M. Strassberger, and Benjamin Weyl. 2009. Communication architecture for cooperative systems in Europe. IEEE Communications Magazine. Volume: 47, Issue: 5. Page(s): 116 – 125.

[14] Marius G. Minea, Iulian G. Bădescu, and S. Dumitrescu. 2011. Efficiency of multimodal real-time travel and traffic information services employing mobile communications. 10th International Conference on Telecommunication in Modern Satellite Cable and Broadcasting Services (TELSIKS). Volume: 2. Page(s): 765 – 768.

[15] B. Jiang, and H. Zhang. 2009. Architecture of Distributed Traffic Simulation System Based on SOA and Multi-Agents. . International Conference on Computer Technology and Development, ICCTD '09. Volume: 1. Page(s): 216 – 220.

[16] A. Gupta, A. Kalra, D. Boston, and C. Borcea. 2008. MobiSoC: a middleware for mobile social computing applications. Mobile Networks and Applications, Volume 14, Number 1, 35-52, DOI: 10.1007/s11036-008-0114-9.

[17] R. Tergujeff, J. Haajanen, J. Leppänen, and S. Toivonen. 2007. Mobile SOA: Service Orientation on Lightweight Mobile Devices. IEEE International Conference on Web Services (ICWS), Page(s): 1224 – 1225.

[18] Y. Zhang, L. Zhou, and E. Mao. 2011. A Survey of Mobile Internet Data Management: Models and Searching Methods.

[19] X. Hu, W. Du, and B. Spencer.2011. A Multi-Agent Framework for Ambient Systems Development. Procedia CS 5, pp. 82-89

[20] X. Hu, J. Zhao, D. Zhou, and V. C.M. Leung. 2011. A semantics-based multi-agent framework for vehicular social network development. In Proceedings of the first ACM international symposium on Design and analysis of intelligent vehicular networks and applications, ACM

[21] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt and W. De Meuter.2006. Ambient-oriented Programming in AmbientTalk", In Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag, Vol. 4067, pp. 230-254

[22] L. Richardson, S. Ruby, "Preface". RESTful web service. O'Reilly Media. ISBN: 978-0-596-52926-0. Retrieved 18 January 2011

[23] http://code.google.com/p/i-jetty/