Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

ETSI compliant GeoNetworking protocol layer implementation for IVC simulations

Ziya Cihan Taysi* and Ali Gokhan Yavuz

*Correspondence:
cihan@ce.yildiz.edu.tr
Department of Computer Engineering, Istanbul, Turkey

Abstract

Inter-Vehicle Communication (IVC) is an important component of the ITS architecture, that enables Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. GeoNet project brought the IVC one step further by enabling transparent IP connectivity between vehicles and the infrastructure. Specifications produced by the GeoNet project have been incorporated into several ETSI standards, which define the geonetworking protocol layer and the IPv6-over-geonetworking adaptation layer. In this paper, we propose an ETSI complaint geonetworking protocol layer on the well known NCTUns simulation framework. We discuss the architecture of our implementation as well as NCTUns specific issues encountered during the implementation. We believe that, with this contribution, the researchers are provided with a scalable and flexible framework with Berkeley socket API access to applications and SQL based performance logging for post simulation evaluations.

Introduction

Inter-Vehicle Communication (IVC) is a promising technology for the next generation of automotive vehicles. Recent advancements in IVC enabled the use of a wide range of traffic safety and efficiency applications. In order to unify and expand services provided by IVC, it is necessary to integrate IVC systems with the Internet. IPv6 with its features, such as extended address space, embedded security, enhanced mobility support and ease of configuration, is the most appropriate candidate for the task.

IPv6 provides an opportunity to use Internet in Intelligent Transportation System (ITS) and expanding it with geonetworking could result in an efficient, scalable and flexible vehicular communication system for safer roads and in a sustainable development of the road network.

The GeoNet European project was the first experimental work investigating the use of IPv6 in vehicular communications. The project ran for two years between 2008 and 2010 and was setup to implement and improve the published specifications of the Car-to-Car Communication Consortium (C2C-CC) [1]. The project proved the concept of combining both IPv6 and geonetworking. It also published experimental test results regarding both indoor and field test performance of IPv6 geonetworking.

Specifications produced by GeoNet project have been incorporated into several standards of the European Telecommunications Standards Institute (ETSI). ETSI TC ITS and

ISO TC204 are developing a set of standards for cooperative ITS, which allow ITS stations including vehicles, road infrastructures and other peers to be reachable through the Internet to cooperate and exchange information with each other for road safety, traffic efficiency and infotainment services. Upon the publication of these standards, ITS technology providers were advised to follow these ETSI standards as baseline for system implementation and testing. As follow-up of the GeoNet project work, GeoNet partners are adapting their system implementations and test environment to be fully aligned with ETSI ITS standards.

There is also ongoing work from academia to widespread the work of GeoNet project. In [2], authors introduce the CarGeo6, an open-source implementation of the GeoNet project. The prototype implementation is a Linux based software programmed in C language and released under a LGPLv2 license. The implementation is currently limited to indoors. However, as suggested by the authors, it is planned to be upgraded for outdoor operations.

Despite of the encouraging work from both industry and academia, there is still an emerging need for researchers to test and validate their contributions in the field of IVC. Especially in real life scenarios, hundreds of vehicles, which are equipped with specific hardware, are required to fully test and validate the requirements of a given application. Thus, a versatile simulation environment with a native support of IPv6 geonetworking would still be an invaluable tool for researchers. In this paper, we present our ETSI compliant IPv6 geonetworking layer implementation in the well known NCTUns simulation environment. The key contributions of our work are as follows:

- An environment for developers and researchers to test and validate their applications on a well defined communication architecture.
- An abstraction layer removing the specific dependencies on the selected NCTUns simulation environment. Thus, minimal modification of the real application, while porting to the simulation environment.
- Unix-like, text-based configuration files to setup characteristics and behavior of the nodes.
- A database oriented logging mechanism to efficiently store and process performance metrics of the simulation.

This paper is organized as follows: Section II briefly describes the ETSI standards, which are used in the implementation of our simulation framework. Section III provides a comparison of the existing methods for IVC simulations. In Section IV, we provide detailed information about our implementation such as the implementation layer, application interface and so on. Section V describes the steps for the configuration of the NCTUns simulation environment and Section VI concludes the paper.

ETSI TC ITS standards

ETSI produces standards for information and communications technologies, including fixed and/or mobile radio, and Internet technologies. ETSI is recognized by the European Union as a European Standards Organization. Several specifications were published by ETSI to define different aspects of the ITS systems such as *Road Transport and Traffic Telematics, Geonetworking and ITS Applications*.

The geonetworking protocol layer is a network layer protocol that provides packet routing in an ad hoc network. It makes use of geographical position information for packet transport. Geonetworking supports the communication among individual ITS stations as well as the distribution of packets in geographical areas. Geonetworking can be executed over different ITS access technologies for short-range wireless technologies, such as ITS-G5 and infrared. The ITS access technologies have many technical commonalities, but also differences among themselves. In order to reuse the geonetworking protocol specification for multiple ITS access technologies, the specification is separated into *media-independent* and *media-dependent* functionalities. Media-independent functionalities are those, which are common to all ITS access technologies for short-range wireless communication to be used for geonetworking. The media-dependent functionalities extend the media-independent functionality for a specific ITS access technology. Therefore, the geonetworking protocol specification consists of the standard for media-independent functionality and at least one standard for each media-dependent functionality.

ETSI defines the geonetworking in ETSI TS 102 636 family of documents. The core of the geonetworking protocol layer is defined by the ETSI TS 102 636-4-1 standard, in the document titled "Geographical Addressing and Forwarding for Point-to-Point and Point-to-Multipoint Communications; Sub-part 1: Media-Independent Functionality". Our focus in this work is to implement a geonetworking protocol layer, which is compatible with the ETSI media-independent standard. This will enable researchers to test their applications in an ETSI complaint simulation environment.

Simulation environment selection

Realistic simulation of IVC protocols is one of the main challenges in Vehicular Ad Hoc Network (VANET) research domain. Performance evaluation of the developed IVC protocols and applications is typically accomplished by means of simulation techniques, since realistic field tests are still infeasible due to high cost of labor and equipment. Therefore, field tests are generally bound only to a few cars. Modern ITS applications employ both driver behavior information and communication between vehicles and the infrastructure or other vehicles. In order to perform valid testing of these applications, based on simulation, the ability to accurately simulate each component of the system is required [3].

Several different approaches have been employed to satisfy this requirement. One of the most common approaches is to use integrated simulation environments such as ns-2 [4], ns-3 [5], OMNeT, NCTUns, GTNets [6]. The main advantage is the availability of precise and well tested models of communication protocols. However most of these simulation environments provide limited support for the node mobility.

Another widely used approach is the decoupling of road traffic simulators and network simulators. In this approach a network simulator is fed with the traces from a road traffic simulator. Traces of road traffic can be produced by several well known micro simulators such as *CityMob, MOVE [7],* and *SUMO [8]*. The most common of these traffic simulators is SUMO. Its main features include support for different vehicle types, collision free vehicle movement, multi-lane streets with lane changing, dynamic routing, and a graphical user interface. SUMO is also capable of generating traffic traces for different simulation environments.

A newer approach is the use of bi-directionally coupled road traffic and network simulators. This approach provides two main benefits; deeper insights into the impact of the network protocols on the road traffic and the ability to obtain more realistic simulation results. TraNS [9], iTETRIS [10], and VEINS [11] are just a few examples of such simulation environments. TraNS was developed to couple SUMO and ns-2. TraNS has the problem of being no longer actively supported since 2008, which restricts its use with the current version of the coupled simulator SUMO. iTETRIS adopts a 3 block architecture. SUMO and ns-3 are two open source platforms, respectively for traffic and wireless simulation, which are coupled by a third block called iTETRIS Control System (iCS).

Another approach to perform simulation of IVC protocols is to use mathematical models. The main advantage of this approach is the ability to perform simulations in a short time. In [12], authors integrate a macroscopic traffic model, an information propagation VANET model and a discrete event-driven protocol model into a framework by using Matlab/Simulink. The proposed framework is able to simulate 2,000 seconds of simulation time in a few seconds. Unfortunately, the accuracy of the simulation results are not so realistic.

Besides the approaches discussed above we also had to take account the fact that the requirements for a given simulation scenario could differ according to the type of the application and the environmental conditions. For example, in safety application scenarios, like collision avoidance, more emphasis would be on the vehicle control based on incoming communication. On the other hand, applications like file sharing require more emphasis on the communication requirements such as bandwidth. Therefore, we eliminated the coupling and decoupling based approaches, since our primary focus is on the communication aspects of the IVC applications.

In order to achieve more realistic results, we decided to employ the integrated environment approach. It also provided us with an easier installation and rapid development. After careful consideration, we decided to use NCTUns as our simulation environment. Besides support for *node mobility, native network socket programming interface along with TCP/IP, and position information*, NCTUns also allowed us to use the real IEEE 802.11p physical layer in our simulations.

System overview

During the implementation, we had to make several architectural decisions such as where within the NCTUns framework our implementation would be fitted in or how user and/or application interaction would be provided. Also we had to choose the most appropriate programming model for a seamless and problem-free integrating with the NCTUns framework.

Implementation layer

The existing protocol stack of NCTUns allows us to integrate geonetworking protocol layer in parallel to the existing WAVE Short Messages Protocol (WSMP) and on top of the IEEE 802.11p. However, in order to avoid many changes or adaptations, researchers in GeoNet project implemented their geonetworing layer in the application layer [13]. Following their footsteps, we also implemented the ETSI compliant geonetworking protocol layer as an user-space application (see Figure 1).

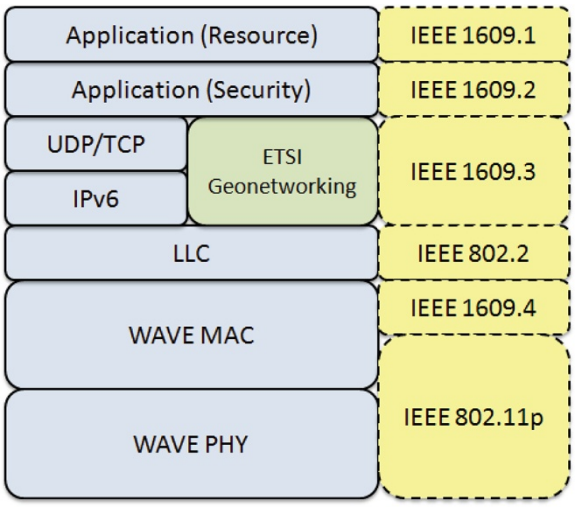

Figure 1 Protocol stack of the NCTUns simulation environment. A block diagram of protocols within the NCTUns simulation environment.

Application interface

We based our implementation of the application interface on the well know Berkeley socket API in order to make its use as seamless as possible for the applications (see Figure 2). Applications, using our geonetworking layer, would connect to a predefined port in order to send and receive packets. Packets exchanged between the application and the geonetworking layer consists of two parts, namely, *the control block* and *the payload*. The control block holds the packet type and addressing information, whereas the payload block holds the actual data. Algorithm 1 shows the flow of a simple traffic generator application for the proposed framework.

Architecture

The ETSI complaint Geonetworking protocol layer can be implemented using either a multi-threaded or monolithic single process programming model. Both programming models have their advantages and disadvantages, such as a multi-threaded approach

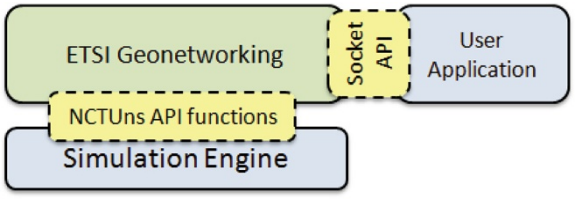

Figure 2 Geonetworking layer application interface; packets are exchanged between the application and the geonetworking layer over the socket API. A block diagram showing the connection between geonetworking layer and user application.

can handle concurrent packets, while a single process approach is easier to implement. We based our implementation on the monolithic single process programming model, as initial tests showed that, multi-threaded extensions to the NCTUns framework were causing instability issues. We employed a modular approach and realized each module as a function.

Algorithm 1 Flow of a simple traffic generator program, which is using our framework for routing packets

```
//
// The control block holds the packet type and addressing
   information.
//
connect to the geonetworking layer through Berkeley socket API;
while TRUE do
    select a random destination for the traffic;
    prepare a dummy payload;
    prepare a control block;
    send control block and payload to geonetworking layer;
    check for incoming packages from geonetworking layer (select I/O);
    if there is any incoming packet then
        process received packet;
        if reply-requested then
            generate a reply package;
        end
    end
end
```

A block diagram of our implementation is given in Figure 3, where each module is represented with a block. Our implementation consists of 10 different modules, which perform specific tasks. The *packet receiver* module listens for incoming packets, and performs type checking using the *HT* field in the common header of the packets, and then puts them into the respective queue. Each packet type is handled by a packet handler module specific to the that type. Thus, we implemented 5 different packet handler modules, namely; *beacon handler, geounicast handler, geoanycast handler, geobroadcast handler and topo-broadcast handler*. Each packet handler module has a circular buffer for incoming packets. On the other hand, they use a shared circular buffer for outgoing packets.

Beacon handler first checks the validity of a received beacon, and then searches for a Location table entry (LocTE) related to the sender of that beacon. If no match is found, a new LocTE is added to the location table (LocT). If a match is found, a timestamp checking is performed to validate the freshness of the received information. If the received packet is newer, then the LocTE is updated with the fresh information. Otherwise, the beacon packet is discarded. The beacon handler is also responsible of generating beacon packets and handing them over to the packet sender for broadcasting. Flow of the beacon handler module is given in Algorithm 2.

Algorithm 2 Flow of beacon handler module

```
// Modules are implemented with cooperative multi-tasking
   approach.
// Thus, there is no blocking operation within these
   modules.
//
check for a packet from the Packet Receiver Module;
if received any beacon packet then
    check the validity of the received beacon packet;
    search for a LocTE belonging to the sender of received beacon;
    if found then
        perform timestamp checking;
        if beacon is fresh then
            update the LocTE entry;
        else
            discard the beacon packet;
        end
    else
        create a new LocTE for the sender;
    end
end
```

Other packet handler modules work in a similar fashion to each other. Each module first performs a duplicate packet checking, then updates LocT by using the position information of the source, the destination, and the previous node. If the packet's final destination is the node itself, then the packet is passed to the upper layer for further processing. If the packet is destined to an other node or a geographical location, it is enqueued to be forwarded by the packet sender module with a new next hop. A generic flow for tha packet handler modules is given in Algorithm 3.

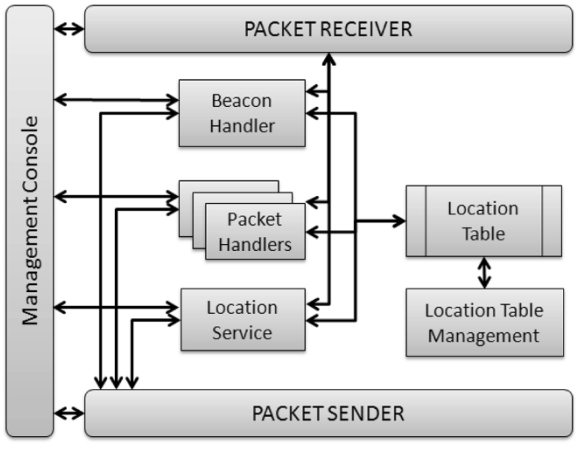

Figure 3 Block diagram of our ETSI complaint Geonetworking protocol layer implementation. A block diagram showing the modules of our geonetworking layer implementation.

Algorithm 3 General flow of the packet handler modules

```
// Modules are implemented with cooperative multi-tasking
   approach.
// Thus, there is no blocking operation within these
   modules.
//
check for a packet from the Packet Receiver Module;
if received any packet then
    perform duplicate packet checking;
    if NOT duplicate then
        update all LocTEs matching the source, destination, and previous node;
        check the destination of the packet;
        if destination address == my address then
            deliver packet to the user application;
        else
            find a next hop for the packet;
            if NOT found then
                put the packet to output queue;
                sender module will try to find a next hop;
            else
                put the packet to output queue with next hop info;
            end
        end
    else
        drop the packet;
    end
end
```

The packet sender module scans the shared circular output buffer of the packet handling modules and sends packets, which are labeled as ready, to their next hop. It also takes care of pending packets, which are missing their next hops. It scans the LocT to find an appropriate next hop.

LocTEs are created by packet handling modules and maintained by the LocT management service. The location table management module checks the validity of LocTEs by comparing the timestamp field of the LocTEs with the current timestamp. The location table management module is also responsible of updating the self position information. Basically it gets the position information from the simulation API and applies the cartesian to geodesic conversion, which is discussed in detail, in Section 1.

The management console shown in the left hand side of the Figure 3 is not part of the ETSI specification, but we added it to our implementation for management and testing purposes. It provides a user interface to fully manage, configure and track the execution of each module. The management console makes it possible to enable or disable individual modules or change their operating parameters, such as *beacon interval, location table entry expiration time*.

ETSI geonetworking specification includes a location service to obtain the position information of other nodes. Whenever a node requires the position information

of a node other than its neighbors, it starts a location service query by broadcasting an *LS Request* packet. Neighbors, receiving this request, reply with the position information of the requested node, or forward this query to their neighbors by broadcast. As it was not our primary concern, we implemented the location query service in a more simplified way. Each node updates its current position information in a shared buffer. This buffer is only used to obtain the position information of a destination, before starting a communication. While forwarding packets, nodes make use of the LocT.

Logging mechanisms

Along a realistic and accurate simulation of any given IVC application, it is also important to gather crucial simulation data for offline processing and evaluation. Therefore, we paid special attention to the logging mechanism. Our primary logging implementation was a simple file system based mechanism. However we observed that this method decreased the performance of the simulation drastically, especially as the number of the vehicles in the simulation increased. Thus, we switched to a database oriented logging system.

Instead of embedding blocking SQL statements into the application, we provided the applications with a UDP based access to the SQL logging. The SQL based logging server runs outside of the simulation environment as a separate process. It receives performance data regarding to simulation from the application in the form of UDP messages and commits them into the database. The benefit of this approach is two fold. First, SQL coding is decoupled from the geonetworking layer, second geonetworking layer does not block on time consuming SQL transactions.

Node configuration

In our framework, we employed Unix-like text based configuration files to configure characteristics and behavior of each node in the simulation. In the configuration files, mechanisms are provided to set the role of the node as either OBU or RSU, or to define the number of RSU, and their unique IDs. A script is also provided for generating configuration files automatically for large deployments of VANETs.

Time measurements

There are two mechanisms to get accurate time measurements. The first mechanism is to use the *time()* function, which can be used by the inclusion of the <*time.h*> library. However, this function only supplies the current timestamp with a second-level precision. Thus, another mechanism is required to make more precise delay calculations.

Therefore, we employed a tick mechanism to make these precise measurements. Whenever the application goes into the run mode from the sleep mode, it increases a variable, which is named as *tick*. This tick variable, together with the *time()* function, enables us to perform high precision timing measurements. These mechanisms are also used in the calculation of end-to-end delays and to achieve a global synchronize among the processes representing the vehicles.

NCTUns specific issues

This section explains NCTUns specific issues that we had to deal with during the implementation of our ETSI complaint geonetworking layer.

Position information

ETSI specification endorses geodesic position information, whereas NCTUns provides only cartesian positions. Thus, a conversion between these units is required. The conversion is performed using the equations given below. In the following equations, *x, y,* and *z* represent cartesian coordinates returned by the simulation API. Longitude (*long*), latitude (*lat*) and altitude (*h*) of the node are calculated by the equations given in 1, 2 and 3 respectively. In the these equations, *a* represents the equatorial radius of the world, which is 6,378,137m and *b* represents the polar radius of the world, which is 6,356,752 m. The flattening, *f*, is defined as in equation 5. The eccentricity of the ellipsoid, *e*, is related to the flattening, and calculated by using equation 4.

$$long = \arctan(\frac{y}{x}) \times 180^o/\pi \tag{1}$$

$$lat = \arctan(\frac{(z + E \times b \times \sin(u)^3)}{(p - e^2 \times a \times \cos(u)^3)}) \times 180^o/\pi \tag{2}$$

$$h = p \times \frac{1}{\cos(f) - \nu} \tag{3}$$

$$e^2 = 2 \times f - f^2 \tag{4}$$

$$f = 1 - \frac{b}{a} \tag{5}$$

$$p = \sqrt[2]{(x^2 + y^2)}, \quad u = \arctan(\frac{z \times a}{p \times b})$$

$$E = \frac{e^2}{1 - e^2}, \quad \nu = \sqrt[2]{\frac{a}{1 - e^2 \times \sin(f)^2}}$$

Process synchronization

We implemented each module as a function, and each function has a specific soft deadline to be executed depending on its task. As a result of our monolithic programming model, all the functions are contained within a single main process. This process uses a sleep-and-run approach. At the beginning of each cycle, the main process checks for pending incoming packets, and puts them into the appropriate packet handler queues. Then, the main process passes the control to modules with due deadlines. As we cooperate with the NCTUns simulation engine, we need to relinquish the control to it periodically. This is accomplished by sleeping for a pre-defined amount of time after the execution of any pending module is completed. Initially, we used the *sleep()* function as it was encouraged in the documentation of the NCTUns simulation tool. Unfortunately, we experienced several problems such as *segmentation faults, unexpected hang of application and/or simulation*. Consequently to solve these problems, we needed to replace the *sleep()* function with the *usleepAndReleaseCPU()* function. Algorithm 4 show pseudo code the main process.

Algorithm 4 Flow of the main process

```
// each module is implemented as a function, and depending
   on its task
// each function has a specific run interval.
//
Initialize GeoNetworking layer; ID, address, timers, etc.;
Initialize GeoNetworking port;
Initialize user application port;
while TRUE do
    check for a packet from geonetworking;
    if received any packet then
        put the received packet to the incoming queue;
        check packet type and invoke appropriate module;
    end
    check for a packet from user application;
    if received any packet then
        put the received packet to the output queue;
    end
    if beacon interval then
        invoke beacon sender;
    end
    if sender interval then
        invoke packet sender;
    end
    if LocTE time-out then
        invoke position management module;
    end
    usleepAndReleaseCPU() ;
end
```

Conclusion and future work

There are several projects and researches based on the integration of IVC systems and Internet. This integration is only possible with the use of IPv6 with its features, such as extended address space, embedded security, enhanced mobility support and ease of configuration. ETSI TC ITS and ISO TC204 developed a set of standards for cooperative ITS, which allow ITS stations including vehicles, road infrastructures and other peers to be reachable through the Internet to cooperate and exchange information with each other for road safety, traffic efficiency and infotainment services.

The primary difficulty of testing IVC applications lies in the scalability of the approach and methodology of the testing itself. As IVC applications mean very large self organizing ad hoc networks, even a simple application requires hundreds of vehicles equipped with OBUs along with wireless infrastructure. Thus, a versatile and accurate simulation environment with flexible configuration and IPv6 support is still the most promising solution to the evaluation and performance testing of IVC applications.

Therefore, we implemented an ETSI compliant IPv6 geonetworking layer in the well known NCTUns simulation environment. In this paper, we discuss the architecture of our implementation as well as NCTUns specific issues encountered during the implementation. We believe that with this contribution the researchers are provided with a scalable and flexible framework with Berkeley socket API access to applications and SQL based performance logging for post simulation evaluations.

Competing interests
Both authors declare that they have no competing interests.

Authors' contributions
ZCT was responsible for the implementation of the ETSI complaint geonetworking protocol layer on the NCTUns simulation framework . ZCT and AGY drafted the manuscript. Both authors read and approved the final manuscript.

Received: 30 September 2012 Accepted: 4 March 2013
Published: 25 March 2013

References

1. Mariyasagayam M, Menouar H, Lenardi M (2008) GeoNet: A project enabling active safety and IPv6 vehicular applications. In: Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE international conference on, 312—316
2. Toukabri T, Tsukada M, Ernst T, Bettaieb L (2011) Experimental evaluation of an open source implementation of IPv6 GeoNetworking in VANETs. In: ITS Telecommunications (ITST), 2011 11th international conference on IEEE, 237–245
3. Baur M, Fullerton M, Busch F (2010) Realizing an effective and flexible ITS evaluation strategy through modular and multi-scaled traffic simulation. Int Trans Syst Mag IEEE 2(3): 34–42
4. The Network Simulator - ns-2. [http://isi.edu/nsnam/ns/]
5. Arbabi H, Weigle M (2010) Highway mobility and vehicular ad-hoc networks in ns-3. In: Winter Simulation Conference (WSC), Proceedings of the 2010. IEEE, pp 2991–3003
6. Riley G (2003) Large-scale network simulations with GTNetS. In: Simulation Conference, 2003. Proceedings of the 2003 Winter, Volume 1, vol 1, Washington DC, USA, pp 676–684
7. MObility model generator for VEhicular networks (2007) [http://www.lens1.csie.ncku.edu.tw/MOVE/]
8. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) SUMO - Simulation of Urban MObility: An overview. In: SIMUL 2011, The third international conference on advances in system simulation, XPS (Xpert Publishing Services), Barcelona, Spain, pp 55–60
9. Piorkowski M, Raya M, Lugo A, Papadimitratos P, Grossglauser M, Hubaux JP (2008) TraNS: Realistic joint traffic and network simulator for VANETs. ACM SIGMOBILE Mobile Comput Commun Rev 12: 31–33
10. Kumar V, Lin L, Krajzewicz D, Hrizi F, Martinez O, Gozalvez J, Bauza R (2010) iTETRIS: Adaptation of ITS technologies for large scale integrated simulation. In: Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st. Taipei, Taiwan, pp 1–5
11. Sommer C, German R, Dressler F (2011) Bidirectionally coupled network and road traffic simulation for improved IVC Analysis. Mobile Comput. IEEE Trans 10(1): 3–15
12. Torok A, Jozsef D, Sonkoly B (2011) A hybrid simulation framework for modeling and analysis of vehicular ad hoc networks. In: Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. pp 1–5. doi:10.1109/VETECS.2011.5956711
13. GeoNet-D2.2 Final GeoNet Specification. [http://www.geonet-project.eu.]

doi:10.1186/2192-1962-3-4
Cite this article as: Taysi and Yavuz: **ETSI compliant GeoNetworking protocol layer implementation for IVC simulations.** *Human-centric Computing and Information Sciences* 2013 **3**:4.