Enabling ITS Real World Experimentation in NITOS Future Internet facility

Nikos Makris, Thanasis Korakis, Dimitrios Katsaros and Leandros Tassiulas
Department of Electrical and Computer Engineering
University of Thessaly
Volos, Greece
Email: {nimakris, korakis, dkatsar, leandros}@uth.gr

***Abstract*—Inter-Vehicle Communication is expected to be widely adopted during the next years by the car industry, enabling Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication targeting at safer commuting. Therefore, the research community have been working towards providing a robust communication solution that will interoperate with existing network infrastructure and will provide an effective way of communicating using location information. One of the proposed solutions is the GeoNetworking protocol, standardized by ETSI, using location based addresses and an IPv6 adaptation sublayer for communicating with the Internet. Several implementations of the protocol exist currently, however most of them are integrated in simulation environments, thus ignoring several factors induced when experimenting under real conditions. In this paper we present our own implementation of an ETSI compliant GeoNetworking protocol and its integration in the NITOS Future Internet facility that enables open and remote access to experimenters on a 24/7 basis. This is the first real implementation in an open wireless testbed, which offers to the research community high diversity in the configuration parameters of their ITS experiments. We further explain our extensions that enable real world testing of our protocol implementation and we finally evaluate our solution in a real large scale wireless setup.**

I. Introduction

The fact that the number of vehicles is steadily increasing over the past years has motivated the research community towards designing and developing fertile ground for road safety and Geo-localization applications. Interconnection and communication of the building blocks of a Vehicular network (VANET) have been investigated towards this goal, thus providing several Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication protocols. Although existing network interconnection protocols seem to be sufficient for communication, road safety applications need an approach based on geographical location data. *GeoNetworking* protocol is one of the protocols able to provide location based routing of requests and has been defined and specified by the Car to Car Communication Consortium (C2C-CC) [1].

GeoNetworking has been standardized by ETSI in the standard TS 102 636-4-1 [2]. It is able to provide communication based on geographical location data, as well as transmit information over the Internet by using an IPv6 adaptation sublayer [3]. Although V2V communication is the goal, in cases where the fleet has to be informed in geographical locations that are beyond the coverage area of wireless communication protocols, an Internet backbone connection has to be employed. IPv6 with its extended addressing architecture has been chosen for these cases of communication over the Internet, with the vehicle node usually using a Road Side Unit (RSU) as the communication gateway.

978-1-4799-4937-3/14/$31.00

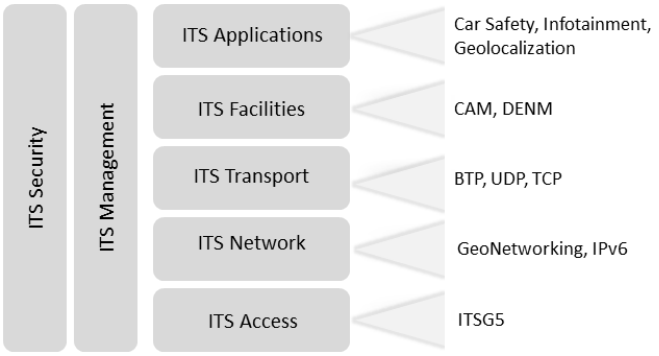

Fig. 1: The basic ITS networking stack

Although GeoNetworking is considered to play a major role in future Inter-Vehicle Communications (IVC), we have noticed a lack of an open experimental environment that a researcher or an SME could use to develop ITS specific applications. Most of the work on this field is limited to simulation environments, where factors existing under real environment settings are ignored or emulated using network models that only apply to a specific subset of use cases [4]. This fact has been identified by the European Commission which has launched several projects that establish network infrastructure widely available and promote experiment-driven research. Experiment-driven research is another emerging trend, where novel protocols and ideas are evaluated and redesigned under real world settings. NITOS Future Internet facility [5], is one major European testbed, initially aiming in wireless networking research. NITOS has been a major component of several projects under the Future Internet Research and Experimentation (FIRE) initiative.

In this work we bundle these two puzzle pieces together; using an implementation of the GeoNetworking protocol and a highly modular architecture, we provide open access to ITS researchers in the NITOS testbed. With the proposed architecture, the experimenter is able to create and evaluate under real world environment applications running on top of the ITS stack (Figure 1). The experiments can utilize heterogeneous network access technologies, such as Ethernet,

WiFi, WiMAX and LTE, thus proving the interoperability of the ITS network stack.

The rest of the paper is organized as follows: In section II we provide some background information on the current efforts in ITS experimental research. In section III we describe our implementation's architecture. Details on the mobility emulation environment that we created to test and evaluate our platform are presented in section IV. Finally, in section V we present some experimental results compared to an existing open source solution.

II. Related Work

As IVC has drawn a lot of attention from the research community, several European funded projects have been trying to standardize V2V and V2I communication in collaboration with the car industry. Since the goal has been the creation of a protocol able to offer address and routing of requests based on geographical location data, several algorithms have been proposed on this field, such as GPSR, GeoTORA, GeoGRID, MORA and MOPR [6]. One of these projects, GeoNet project [7], focused on cooperative ITS communications and tried to bundle together useful pieces of the aforementioned algorithms. Its outcomes were further enhanced and standardized as the GeoNetworking protocol by C2C-CC.

Before the standardization of GeoNetworking, evaluation of IVC protocols took place under simulation environments using the aforementioned algorithms. Several additions to widely adopted network simulators have been proposed that enable the operation of the GeoNetworking protocol. For instance, in [8] the authors propose a GeoNetworking protocol layer implementation on the NCTUns simulation framework. Similar to this, in [9] a highway mobility model for VANETS is proposed, integrated in the ns-3 simulator. Using these proposed models, the researchers are able to further enhance their ITS specific simulations using parts of the ITS stack.

However, although the accuracy of the simulation solutions has significantly improved throughout the last few years, real world effects are usually left out of the simulation parameters or are modeled using mathematical models. As an outcome, simulation results are proven to be incomparable with results received under real world settings, and especially when these are conducted using wireless networks. In [4] the authors compare results from NS2 and GloMoSim simulators with that of those received under an emulation testbed. As they prove, the simulation results were even not comparable with each other in some specific scenarios. Similar results to these are presented in [10] where a comparison among experiments conducted in the most widely adopted simulators are described. The incompatibility of the solutions presented by simulation results is ought to the assumptions that they make. This applies best in experiments conducted using wireless networks where the external factors can be non-deterministic and can have a large impact on the conducted experiment[11]. Therefore, the ITS research on VANETs and their intercommunication conducted under a simulation environmet may be highly questionable on their results and their application under real environment conditions.

As a result of these, the need of the experimental evaluation of any new protocol and idea under real world settings is obvious. However, for the ITS specific research there is no similar environment for benchmarking applications and services. This happens due to the lack of publicly given platforms able to be setup easily, even from the inexperienced users. To the best of our knowledge, the only open source and publicly available implementation compatible with the GeoNetworking protocol is the CarGeo6 [12]. Its development sticks to the delivered architecture of the GeoNet project, while the source files of it are given publicly. CarGeo6 is also using the IPv6 adaptation sublayer, rendering several instances of it able to intercommunicate through the Internet, as it is standardized in [3].

However, the ITS stack is using different protocols at each layer, each one with its own restrictions and limitations towards meeting the requirements of vehicular applications. Each layer (Figure 1) is using different mechanisms crucial for the successful communication among the VANET components. For instance, apart from the GeoNetworking core, a separate transport protocol has to be employed to ensure data transferring among applications running on top of the stack. Separate position update modules have to be used in order to ensure that position data received from the GPS module attached on the ITS node keep the GeoNetworking core constantly updated. Different protocols running on top of the transport layer, by means of a facilities layer, should be able to trigger the appropriate messages by sending predefined data indications to the GeoNetworking core. However, CarGeo6 is only provided without the support of any higher layers as it focuses in providing network interoperability as the authors state in [12].

In this work, we move a step beyond this work in ITS research; we establish an application space ITS stack on a widely used EU wireless testbed. We use our own implementation in C++ language of the different ITS components and finally evaluate it against the CarGeo6 open source implementation. Since our goal is not to provide a better implementation than the existing ones but an interoperable one with a full ITS stack support, we conduct our experiments and indicatively measure delays of multihop and singlehop delay in the NITOS testbed.

III. Implementation Details

In this section we present some more details on our implementation and its building blocks. The developed GeoNetworking functionality is running as a user space application daemon. In order to provide functional experimentation with the GeoNetworking core we had to develop minimal versions of other key component layers of the ITS protocol stack, such as an appropriate Transport layer, a Facility layer and a Management layer. The details for each layer and the architecture of communication among our developed applications are described in the next subsections.

A. Communication Between Layers

Towards realizing our GeoNetworking implementation, we had to come up with a communication solution among the different sub-applications that comprise our solution. Towards this goal, we employed the standard UNIX socket API. The different sub-applications needed to run on our platform for real world experiments are

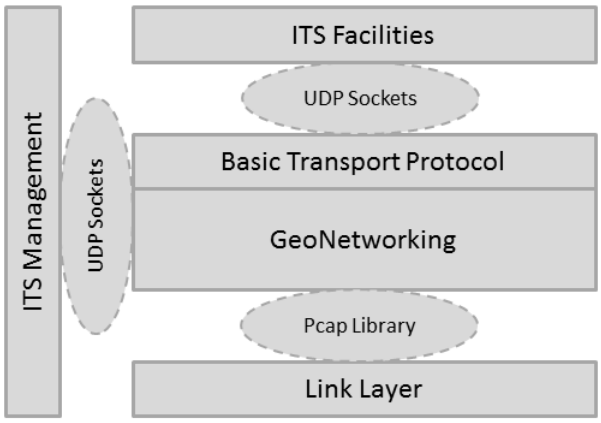

Fig. 2: The architecture of our GeoNetworking protocol implementation

- the basic GeoNetworking core,
- the Basic Transport Protocol (BTP) as a means of a Transport layer protocol,
- a minimal facility layer used to generate traffic,
- a minimal implementation of the ITS management layer.

The communication among these sublayers is done through network sockets, communicating over the ITS station's localhost interface. The packets are exchanged through the UDP sockets and involve the information described in the ETSI Annex of GeoNetworking.

Apart from these implementations, the selection of a MAC protocol standardized by ETSI for ITS communication is of paramount importance. ITSG5 and its variations of ITSG5A, ITSG5B and ITSG5C rely on a distributed wireless protocol, supporting adhoc communication in the 5.9GHz band, exploiting the IEEE 802.11p protocol. For these reasons, we use a commercial IEEE 802.11 protocol compliant card, compatible with the *ath5k* Open Source driver that supports packet injection of already formed packets through the *pcap* library.

To sum up, the communication of the different subparts of the ITS stack is described in Figure 2. Basic Transport protocol is implemented inside the GeoNetworking core, and communicates via UDP sockets with the upper facility layer. GeoNetworking core communicates likewise with the management layer and using inject functions with the IEEE 802.11 and 802.3 link layer. In the next subsections we discuss about our implementations on each layer, the currently supported operations and the issues we had to resolve in order to come up with a functional solution.

B. The GeoNetworking Layer

Our implementation realizes the ETSI TS 102 636-4-1 for Geographical addressing and forwarding for point-to-point and point-to-multipoint communications, providing media independent functionality. The address format specified in the standard is using a 64-bit representation, comprised of a bit for distinguishing whether the address is set automatically or manualy, a 4-bit field representing the ITS station type (roadside unit or a vehicle), a 1-bit indicator of whether the station is a private or public transport vehicle, a 10-bit field indicating the country code and finally a 48-bit field with the MAC address of the station. Since our implementation targets at giving open access to the experimenters to a working GeoNetworking protocol, we use a configuration file as an input to the GeoNetworking daemon. As it is specified in the standard, a GeoNetworking implementation should be configured through a Management Information Base (MIB). We consider that provisioning a MIB is beyond our initial goals, so we expose some of the MIB parameters through an external configuration file. All these aforementioned parameters can be setup through the configuration file provided to the experimenter.

Moreover, through the configuration file the experimenters can setup the communication parameters with the facility and management layer, in case they want to use their own implementation, and the primary interface that will be used for communication. It is worth to mention that we currently support only Ethernet and WiFi interfaces, but we intend to extend it to using WiMAX and LTE interfaces.

The standard specifies six different types of packets;

1) Beacons
2) Single Hop Broadcast (SHB)
3) Topologically Scoped Broadcast (TSB)
4) GeoBroadcast/GeoAnycast
5) GeoUnicast
6) Location Service packets

Our implementation is able to support all GeoNetworking type of packets, except for the creation of the Location Service packets. These type of packets are triggered when no entry of the location of a specified host is found in the Location Table (LocT). Beacon packets are exchanged when a specific timer expires, which is reset each time that the GeoNetworking enabled station sends a packet of a different type. This feature explicitly defines that Beacon packets are sent only when no other communication involving the specific station takes place, so as for it to advertise its location. The rest type of packets are triggered when receiving an appropriate indication by the facility layer. TSB messages are used for broadcasting messages to nodes located more than one hop away. GeoBroadcast messages are used in the case of broadcasting/anycasting messages inside a geographical region, and are used mainly for road safety and event information applications implemented in the application layer. Finally, GeoUnicast messages are used for transmitting messages to a specific host whose location information is already known.

The implementation of the LocT is done using a hash table with a simple hash function, relying on the MAC address of the received packet. Since the hash function is a rather simple one, we have predicted and developed a hash collision mechanism using separate chaining of the collided entries using a linked list. The hash function is using as input the MAC address field of each received packet (as it is extracted from the GeoNetworking address) to retrieve each LocT entry. This is a notable difference compared to the CarGeo6 implementation,

which is using linked lists that can have a large scanning overhead in dense networks.

Our daemon is written in a way that can scale in terms of heavy node network congestion. It is multi-threaded, using different threads for the process of beaconing, receiving packets from the network, receiving packet requests from the facility layer, receiving management messages from the management layer, holding the location table and finally for invalidating expired entries in the LocT. The threads responsible for receiving messages are subsequently forked in the case that a transmission of a packet is required, either when the station is a transmitter and the network packet has been received from a higher layer or when the station is a forwarder of a packet already received from the Link Layer.

Furthermore, since the GeoNetworking protocol is built so as to maximize the spreading of information in a geographical area, each packet might be retransmitted several times within it. Typical examples are the TSB and GeoBroadcast packets, where each receiving host will retransmit the packet if either the hop count is greater than zero or the host is inside a defined geographical area. For this reason, a duplicate packet detection mechanism has to be used. The packets which are meant to be retransmitted bear a sequence number (SN) incremented by one each time that such a packet is triggered. By retaining an entry in the LocT with the last SN received from each neighbour, each host manages to keep track of the packets received per host. Using this kind of mechanism, GeoNetworking core is able to detect duplicate packets and discard them without any further processing.

In order to accomplish true communication using location information, we used the definitions of a geographical area as defined in the ETSI EN 302 931 standard [13]. The standard defines three different types of geographical areas; circular, rectangular and elliptical. Each one of the areas is defined using two points and the azimuth angle of the longest axis. The GeoNetworking protocol is using this information in the cases of GeoBroadcast and GeoAnycast messages only and should totally be agnostic to these parameters; the protocol receives these parameters by means of a data request received from the facility layer.

C. Basic Transport Layer

The proper selection of a Transport layer protocol is crucial for the successful communication of multiple running instances of the GeoNetworking daemon. In ETSI TS 102 636-5-1 standard [14], the Basic Transport Protocol (BTP) is described for connectionless transport service among ITS stations. BTP is responsible for communicating the requests to/from the facilities and the GeoNetworking layers.

In the standard two variations of the protocol are described; BTP-A and BTP-B. Depending on the facilities layer protocol used, one of the two variations of the protocol is chosen. Its advantage is that it is a lightweight transport service protocol using very simple headers. In fact, it only uses a 4-byte long header for destination port and source port or destination port information indications. Due to its simplicity we decided in our implementation to integrate it with our GeoNetworking daemon, thus reducing the overhead of communication over a single node's localhost interface. When the daemon receives a request from the upper layers, BTP is used to encapsulate the packet based on the information indicated in the request and subsequently delivers the packet to the GeoNetworking layer for further processing.

Fig. 3: The Mikrotik R52 Card for 5.9 GHz band operation

D. Facility Layer

In order to create some traffic over our test environment, we had to create a traffic generation scheme. As in our implementation we have been sticking to the GeoNetworking standard, which describes the overall architecture as a modular system, we had to implement a minimal upper facility layer that would generate the requests. It is worth to mention that in our setup the facility layer is given as an Open Source solution that enables the experimenter to configure it appropriately in order to suit his/her experiments. The two different protocols operating in the facility layer, which are standardized by ETSI, are the Cooperative Awareness Messages (CAM) [15] and the Decentralized Environmental Notification Messages (DENM) [16].

Each facility layer protocol triggers a different type of messages in the Transport and GeoNetworking layer; CAMs encapsulate into BTP-A and SHB or TSB packets while the DENMs into BTP-B and GeoBroadcast or GeoAnycast messages. This choice has been made by ETSI in order to focus on road hazard safety messages that get transmitted in a single geographical region usually in the case of a traffic accident.

E. Link Layer

In order for our implementations to stick to the ETSI standards for PHY communication, we had to adopt a solution that is close to the ITSG5 description. For this reason, we employed the Microtic R52 miniPCI card on the testbed nodes (Figure 3), able to transmit in the 5.9GHz frequency and configurable in adhoc mode. To the best of our knowledge, this setup is the closest one to the IEEE 802.11p implementation. These mini-pci cards use an Atheros based chipset and therefore are compatible with the existing Open Source driver *ath5k*. A feature of this driver is that it supports packet injection, which can override the standard driver process and inject an already formed packet directly onto the card. In order to be able to inject packets directly to the driver, the use of the *pcap* library is required.

The exact reverse process is taking place in the case that a packet is received from the network. The *pcap* API gives

access to the packets received from the network by duplicating them from the network buffer into our application, before being processed by the MAC driver. So our GeoNetworking application has a thin layer responsible for stripping off the MAC headers and delivering the packet to the GeoNetworking core. Apparently this approach is not the most effective one, since any packets which are not destined to the specified host are processed twice, initially by the default MAC layer driver and one more time in our daemon before getting discarded. However this is a trade-off that we have in order to facilitate "direct" access over the network. It should also be noted that the GeoNetworking type of packets not processed by our daemon get discarded by the MAC layer of the host before delivering them to the normal IP layer.

F. Management Layer

A final application that we had to develop in order to have full functionality of our GeoNetworking daemon is a minimal implementation of an ITS Management layer. The management layer is of paramount importance in the GeoNetworking protocol specification, as it is responsible for informing any position updates to the network layer, updating the GeoNetworking address in case that a duplicate is found and informing of any time changes. Our management layer is using the communication prototypes specified by ETSI for communicating with the core daemon and receiving updates from it. In the following section we give some more information on its necessity and the role it plays in the overall mobility emulation framework.

G. Logging Support

In order to provide the experimenter with the appropriate feedback on the ITS experiment, separate logging mechanisms have been developed running at each ITS stack layer. The GeoNetworking core is monitoring and logging every incoming packet, and provides feedback on the timestamp with microsecond accuracy on the received packet and the position information of the host it has sent it. In some cases that the packet is a forwarded one, and thus the source of the packet is different than the sender, this information is logged as well. Logging this kind of information has a low cost overhead, since the GeoNetworking core inspects these packet fields for keeping the location information on the receiving node updated. Moreover, the facility layer logs in separate log files the packets sent to and received from the GeoNetworking core. Finally, the management layer is keeping information on the position update messages sent to the core system.

IV. Mobility Emulation Environment

As we have already described, the evaluation of our implementations took place in the NITOS Future Internet facility. NITOS is an outdoor deployed, large-scale wireless testbed, currently consisting of 50 operational WiFi nodes in the premises of a University of Thessaly campus building (Figure 4). The testbed is using the state-of-the-art control and management framework for testbeds, namely OMF [17], for conducting distributed large scale experiments easily. OML, OMF's accompanying library is used for collecting experiment specific measurements in a database.

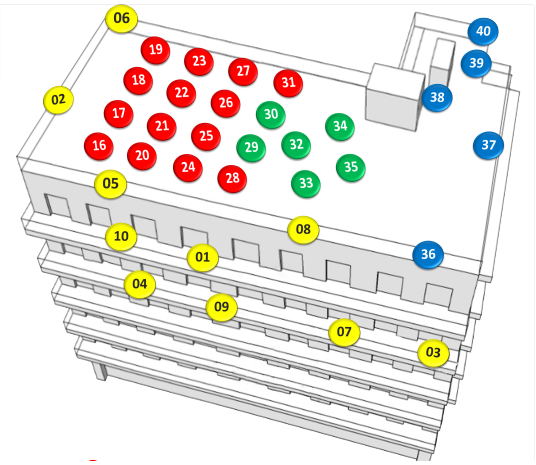

Fig. 4: The NITOS wireless testbed

Since the nodes comprising the testbed are static, we had to create an environment that would emulate the mobility at least at the GeoNetworking layer.

To this aim, we employed the *gpsfake* and the *gpsd* applications. The former is used for parsing a file with NMEA sentences and feeding this data directly to the latter. *Gpsd* is the state-of-the-art application in UNIX systems for parsing data from a GPS device. Any application can query a running *gpsd* instance for position information which is returned to it in a JSON format.

Our Management layer is using this exact procedure; queries the *gpsd* daemon, parses the JSON file and creates the appropriate position update message for the GeoNetworking daemon. By using this approach, we have managed to emulate mobility in the Network layer of the NITOS nodes. The validity of our approach has been verified through our experimental framework where the nodes inside a geographical area only retransmit the GeoBroadcast packets in the case that they are all located inside it. However, action has to be taken in order to have a fully working mobility setup in the Link layer as well, so as the nodes defined under different geographical areas are outside each other's coverage area. We are able to currently support this action manually by carefully selecting the nodes that comprise our experiment and change their transmission power. Using the *ath5k* driver we are able to configure the transmission power of the WiFi card with a value between 0-27dBm. Although this process takes place by issuing the appropriate commands, we are currently in the process of extending this kind of support by using UNIX *debugfs* filesystem. Directly from the management layer we will inform the wireless driver about the node's position to appropriately adjust its transmission power, thus enhancing our framework with a cross-layer approach.

V. Evaluation Results

In this section we describe our experimental results from bundling together our applications. Our evaluation takes place in two parts. Initially we setup the nodes in a wireless adhoc network and set their wireless parameters in a way that we create a three hop wireless network and measure the end-to-end

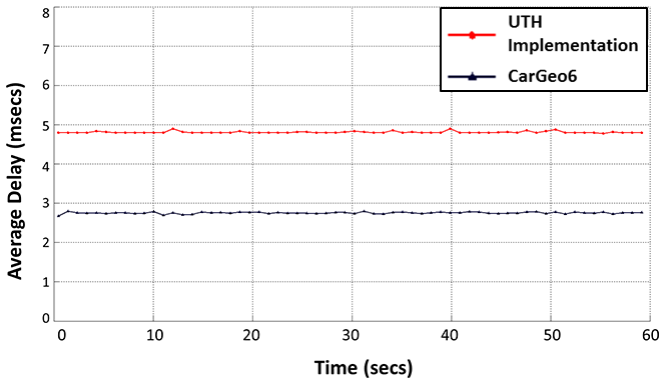

Fig. 5: Delay measurement for intensive traffic

delay. As a second phase of our evaluation, we perform some stress tests on our application in order to check its performance under heavy traffic, compared to the CarGeo6 application.

A. Multiple Hop end-to-end delay

This performance evaluation is provided as a proof of concept that our implementation of the protocol is fully functional. For this reason we employ four NITOS nodes. However, even if we setup the transmission power to 0dBm, some of the nodes are still able to communicate with each other. Therefore, we had to use the *dummynet* application, able to perform layer 2 filtering on the nodes based on a list of given MAC addresses.

By applying the aforementioned rules, we conduct our experiments using four nodes. Since no traffic generator is currently compatible with the GeoNetworking protocol, we had to develop our own. By using the facilities layer we trigger CAM packet requests every 2 seconds, while the beacons are configured to be transmitted every 3 seconds. The facility layer requests prompt GeoNetworking to use a 3-hop TSB message. The facility application is started after all nodes have exchanged their beacon packets and have each other's neighbour in their LocT. The experiment's duration is 60 seconds and takes place over IEEE 802.11a using channel 40, which is RF isolated.

We measure the timestamps of consecutive CAM messages on the receiving node. As we expected to see, the interarrival times of the consecutive CAM messages is steady and almost equal to the 2 seconds that each CAM request is generated. However in order to measure the Round Trip Time (RTT) achieved using our implementation we had to further tailor the facility layer of the receiving node to trigger back a reply. Comparing to the IP layer, when using the exact same setup, we measured average RTT values of 3.1 msecs with our application layer measurement tool, utilizing UDP sockets. Our implementation achieves a little bit higher RTT values of around 5.2 msecs for the exact same setup.

This delay is considered to be fair and tolerable when comparing the two protocols; Our implementation runs as an application space daemon and has to go through the LocT entries for each received packet, form a new packet and deliver it to the facility layer, update the headers and then inject it again on the network. These operations clearly introduce more delay. However, as we have already mentioned, this is our first step as a proof of concept that our application is functional.

B. One-Hop Delay Measurement

In this subsection we present our experimental results with regard to an existing open source solution of the GeoNetworking standard, namely CarGeo6. CarGeo6 is an implementation of not only the GeoNetworking protocol, but bundles together other ITS specific ETSI standards, such as the adaptation sublayer for IPv6. Nonetheless, it currently cannot provide a full ITS stack support as it does not have any higher layer functionality. The source of this implementation is given publicly available. Therefore, in order to directly compare our efforts with it, we decided to make CarGeo6 able to cooperate with our higher layer applications.

TABLE I: Basic Setup of NITLAB nodes

CPU / Memory	2.26 GHz Intel Core 2 Duo P8400 / 2048 MB RAM
Operating system	Ubuntu 11.04 / kernel ver 2.6.38-16
Wireless cards	Atheros 802.11a/b/g & Atheros 802.11a/b/g/n (MIMO)
Wireless driver	ath5k
Ethernet interfaces	2 NICs with 1Gbps connection

For the evaluation of our implementation we set up the CarGeo6 and our applications between two nodes of the NITOS wireless testbed. Although the nodes feature multiple wireless interfaces, we choose to use the node's experimental Ethernet interfaces as both the ingress and outgress interfaces of the nodes. We choose to use this kind of connectivity since NITOS is an outdoor testbed with high external interference that could prove to be detrimental for conducting this kind of benchmarking measurement experiments. We decide to conduct delay measurement experiments between the two protocols in order to prove that the two implementations are close to each other. Delay measurements from a higher layer and using CSMA/CD as a medium access protocol will provide us insights on how the applications are built and how they respond to heavy network traffic.

For this purpose, we use our implementation of the facility layer and configure it to generate CAM messages with high intensity. This is done by sending consecutive requests without any sleep interval. In order for it to communicate with the CarGeo6 implementation, we had to further add the same hooks in the code that make it communicate with our facility layer; this is basically a UDP server listening for higher layer requests and a UDP client for delivering received CAM messages to the facility layer.

We finally setup the facility layer to generate massively CAM requests delivered to the GeoNetworking core. By using the OML library (OMF Measurement Library) we collected timestamps of the reception time of consecutive data indications in the facility layer. Using two NITOS nodes, we measured average delays of approximately 4.5-4.8 msecs for our implementation and 2.6-2.8 msecs for the CarGeo6 application (Figure 5). These delays are measured per packet for an experiment lasting for 60 seconds. Measurements gathered within a second are aggregated and averaged. Since this kind of experiment can be dependent on the hardware specifications of the node, we provide this information in Table I.

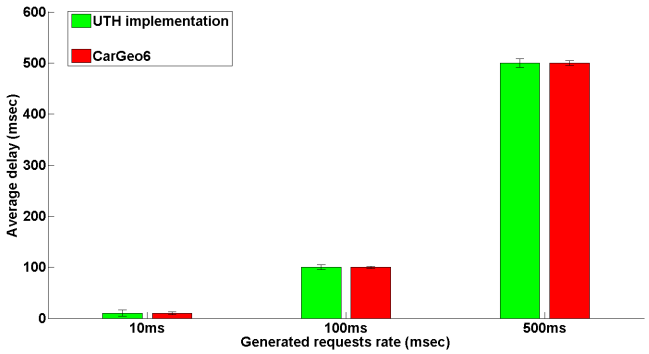

Fig. 6: Delay measurement for traffic generated per 10, 100 and 500 msecs

Although the differences in time are observable, the generation of measurements is relying on the way the two applications are developed; Our implementation uses the *pcap* library for injecting the traffic on the network and registers a callback function triggered by every incoming packet, while the CarGeo6 opens raw sockets over the network interface. Our approach is dependent on the implementation of the library and how it gets access over the network which can significantly decrease performance. On the other hand, raw sockets give the flexibility to handle each incoming packet directly from the network by assigning a thread to continuously listen to it.

All in all, although the first impression is that our application is outrun by CarGeo6, this performance overhead is created by the access mechanism that we use over the network and in the cases of heavy traffic. We repeat the same experiment for different time intervals of traffic generation and measure our results with a microsecond accuracy.

As we observe in Figure 6, for traffic generated per 10, 100 and 500 msecs, the delays for each implementation converge together. When conducting experiments with generating traffic at a shorter interval than 10 msecs, we observed that for this setup of nodes, our implementation was able to efficiently deliver packets with the same rate they were generated for an application rate of up to 2 msecs. For higher rates, our solution was unstable, started dropping packets and was inconsistent in the delivery rate.

VI. Conclusion

In this paper we presented our ITS stack solution for experimenting under real world settings using the NITOS wireless testbed. We described the communication architecture of the different ITS stack components that we implemented and how they intercommunicate in order to provide GeoNetworking support to the testbed nodes. We finally evaluated our implementation in a real testbed setup and benchmarked our implementation with an open source solution. The results we obtained are very encouraging and we believe that we will be able to further enhance our solution by altering the library we are using for medium access.

Our future work will be on enhancing our system and performing field tests by mounting the NITOS nodes in cars. Another enhancement that we will work towards to will be the implementation of the IPv6 adaptation sublayer, that will enable the protocol's interoperability with existing network tools. This enhancement will give us the opportunity to further benchmark our solution by using widely adopted traffic generators, such as Iperf.

References

[1] Car to Car Consortium, http://www.car-to-car.org/.

[2] "ETSI TS 102 636-4-1: "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality " V1.1.1 (2011-06)", http://www.etsi.org/deliver/etsi_ts/102600_102699/1026360401/01.01.01_60/ts_1026360401v010101p.pdf.

[3] "ETSI TS 102 636-6-1: "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 6: Internet Integration; Sub-part 1: Transmission of IPv6 Packets over GeoNetworking Protocols " V1.1.1 (2011-03)", http://www.etsi.org/deliver/etsi_ts/102600_102699/1026360601/01.01.01_60/ts_1026360601v010101p.pdf.

[4] Furqan Haq and Thomas Kunz. Simulation vs. emulation: Evaluating mobile ad hoc network routing protocols.

[5] NITOS: Network Implementation Testbed Laboratory using Open Source platforms, http://nitlab.inf.uth.gr/NITlab.

[6] Z. Cihan Taysi and A. Gkhan Yavuz. Routing protocols for geonet: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 13, 2012.

[7] GeoNet project, http://www.geonet-project.eu.

[8] Ziya Taysi and Ali Yavuz. Etsi compliant geonetworking protocol layer implementation for ivc simulations. *Human-centric Computing and Information Sciences*, 3, 2013.

[9] Hadi Arbabi and Michele C. Weigle. Highway mobility and vehicular ad-hoc networks in ns-3. In *Proceedings of the Winter Simulation Conference*, WSC '10, 2010.

[10] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: The incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4), oct 2005.

[11] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '04, 2004.

[12] T. Toukabri, M. Tsukada, T. Ernst, and L. Bettaieb. Experimental evaluation of an open source implementation of IPv6 GeoNetworking in VANETs. 2011.

[13] "ETSI EN 302 931: "Intelligent Transport Systems (ITS); Vehicular communications; Geographical Area Definition " V1.0.0 (2010-12) " http://www.etsi.org/deliver/etsi_en/302900_302999/302931/01.00.00_20/en_302931v010000c.pdf.

[14] "ETSI TS 102 636-5-1: "Intelligent Transport Systems (ITS); Vehicular communications; Basic Set of Applications; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol " V1.1.1 (2011-02)" http://www.etsi.org/deliver/etsi_ts/102600_102699/1026360501/01.01.01_60/ts_1026360501v010101p.pdf.

[15] "ETSI TS 102 637-3: "Intelligent Transport Systems (ITS); Vehicular communications; Basic Set of Applications; Part 2: Specifications of Cooperative Awareness Basic Service " V1.2.1 (2011-03)" http://www.etsi.org/deliver/etsi_ts/102600_102699/10263702/01.02.01_60/ts_10263702v010201p.pdf.

[16] "ETSI TS 102 637-3: "Intelligent Transport Systems (ITS); Vehicular communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service " V1.1.1 (2010-09)", http://www.etsi.org/deliver/etsi_ts/102600_102699/10263703/01.01.01_60/ts_10263703v010101p.pdf.

[17] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. OMF: a control and management framework for networking testbeds. *ACM SIGOPS*, 2010.