Reproducing and Extending Real Testbed Evaluation of GeoNetworking Implementation in Simulated Networks

Ye Tao, Manabu Tsukada, Xin Li, Masatoshi Kakiuchi, Hiroshi Esaki

Reproducing and Extending Real Testbed Evaluation of GeoNetworking Implementation in Simulated Networks

Ye Tao [*]
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-8656, Japan
tydus@hongo.wide.ad.jp

Manabu Tsukada [*]
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-8656, Japan
tsukada@hongo.wide.ad.jp

Xin Li [†]
Beijing University of Posts and Telecommunication
10 Xitucheng Rd., Haidian District, Beijing
100876, China
leexin@bupt.edu.cn

Masatoshi Kakiuchi [‡]
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara
630-0192 Japan
masato@itc.naist.jp

Hiroshi Esaki [*]
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-8656, Japan
hiroshi@wide.ad.jp

ABSTRACT

Vehicular Ad-hoc Network (VANET) is a type of Mobile Ad-hoc Network (MANET) which is specialized for vehicle communication. GeoNetworking is a new standardized network layer protocol for VANET which employs geolocation based routing. However, conducting large scale experiments in GeoNetworking softwares is extremely difficult, since it requires many extra factors such as vehicles, stuff, place, terrain, etc. In this paper, we propose a method to reproduce realistic results in simulation with the same software implementation. The key idea of the method is to calibrate simulator with the results of real world testbed experiments. After the simulator was calibrated, some extended experiments were carried out. Through these experiments, the fundamental functions of the GeoNetworking implementation (BTP, Greedy Forwarding, etc.) are verified, while an issue in algorithm was discovered and analyzed.

1. INTRODUCTION

Intelligent Transportation Systems (ITS) aim at optimization of the road traffic by realizing safe, efficient and comfortable transportation. Within a number of research fields in ITS, Cooperative ITS and vehicular communications became essential for the cooperation of multiple entities in the road traffic (*i.e.* vehicles, roadside infrastructure, traffic control centers) in order to achieve shared objectives (safety, efficiency, and comfort).

In order to connect among vehicles and roadside units, GeoNetworking [1] is employed as one of the network protocols in the ITS Station architecture [2], as shown in Figure 1, because the geolocation based routing features the strength in the network with dynamic topology compared with topology based routing.

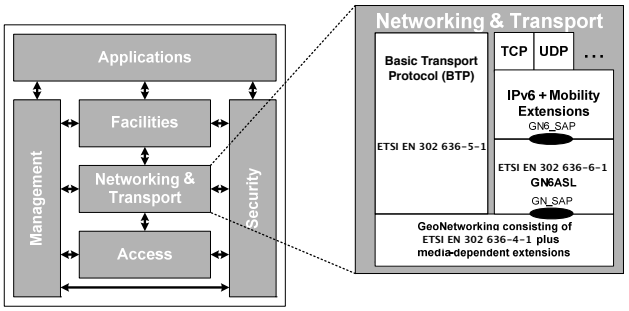


Figure 1: GeoNetworking in ITS Station Architecture

In the literature, the evaluation of GeoNetworking can be performed in flexible and large scale simulated network with low cost. However mere simulations cannot provide realistic evaluation results for a specific implementation of GeoNetworking. In contrast, the experimental evaluation using the implementation in a field operational testbed gives real results in the deployment phase of GeoNetworking. Though in

[*]Graduate School of Information Science and Technology, The University of Tokyo

[†]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications

[‡]Information Initiative Center, Nara Institute of Science and Technology

CFI '15, June 08 - 10, 2015, Seoul, Republic of Korea

ISBN 978-1-4503-3564-5/15/06...$15.00

DOI: http://dx.doi.org/10.1145/2775088.2775092

practice, it requires heavy cost to conduct the experiments in terms of time, manpower, space and expense. In order to take the benefits of real field test and simulation, we reproduce the results of the field experimental evaluation in the simulated network with the same implementation.

The rest of the paper is organized as follows. Section 2 highlights the related works. In Section 3, we describe our objectives in the paper. In Section 4, the experimental evaluation in the real testbed is shown. Section 5 shows the work for the reproduction of the experimentation result in the simulated networks. Section 6 extends the reproduction to the various scenarios in the simulation, and analyses an issue in the algorithm. Finally, Section 7 concludes the paper by summarizing the main results and addressing future works.

2. RELATED WORKS

2.1 GeoNetworking and the Implementation

Vehicular ad-hoc network (VANET) is a particular case of Mobile Ad-hoc Network (MANET), which is not restricted by the battery consumption of the communication nodes and are also characterized by the high speed movement of nodes, the availability of GPS information, and a regular distribution and predictable movements. *Greedy Perimeter Stateless Routing (GPSR)* [3] employs GPS information to forward the packets in order to adapt VANET characteristic where routes become quickly unavailable in high mobility scenarios. In GPSR, the nodes do not need to maintain part of the network structure in order to forward packets towards the destination node, but make a forwarding decision based on the destination position and neighbor positions. Within the ITS standardization domain, GeoNetworking is being completed by ETSI at the moment, integrating several geo-aware strategies to route packets better in vehicular networks.

GeoNetworking [1] is standardized by ETSI as a network layer protocol as in Figure 1, integrating several geo-aware strategies including Greedy Forwarding (GF) [3] (Also known as *GPSR*), which chooses an directly reachable node which is closest to the destination based on GPS location obtained by *Location Service(LS)* request action, to route packets better in vehicular networks. Above the GeoNetworking, there are two different layers. One is *Basic Transport Protocol (BTP)* [4] which provides basic functions of the transport layer to GeoNetworking, the other is *GeoNetworking to IPv6 Adaptation Sub-Layer (GN6ASL)* [5] in order to enable standard IPv6 over GeoNetworking.

All the GeoNetworking nodes send beacons in a specific interval, and the neighbor nodes maintain its latest geographical location in the *location table (LocT)* from the received beacons. Other GeoNetworking packets delivered in the network contain the location of *source (SO)*, *sender (SE)* and *destination (DE)*; in the case that the location information in the packet is newer than the one in the location table, the location table is updated. Each *location table entry (LocTE)* has a lifetime counter, and the entry is removed when it is reduced to 0. When the source node does not have location of the designation in its location table, the node triggers the *Location Service (LS)* request message in order to obtain the location of the destination. ETSI defines the flooding based request-reply location service to get the destination location.

More than fifteen software implementations of GeoNetworking join ETSI plugtest that provides interoperability test opportunity every year. The CarGeo6 project[1] provides GeoNetworking implementation in open source [6]. The GeoNetworking function and the BTP function are implemented as daemons called `itsnet` and `btpecho`, respectively in the CarGeo6 implementation as in the Figure 2. In source node, `btpecho` (client mode) sends a BTP packet via inter-process communication to `itsnet`. If the destination location is in `LocT`, `itsnet` forwards the packet to next hop selected by GF, otherwise it triggers an LS request. Finally, when the BTP echo request is forwarded to the destination, `itsnet` send the packet to `btpecho`. On the other hand, `btpecho` (reflector mode) in the destination node sends a BTP echo reply back to the source once it received a request. The echo reply is forwarded by GF too, thus the reply packet may be delivered via a different route from the request packet.

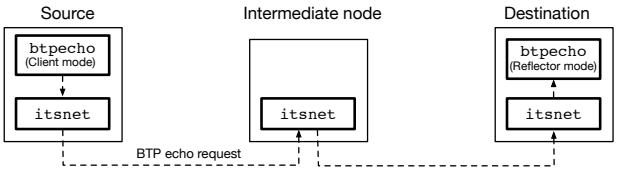


Figure 2: Overview of CarGeo6 programs

2.2 Experimentation and Simulation

The evaluation of GeoNetworking is performed a number of times in the simulations [7, 8], because it is costly to make the experimentation in a field testbed with real vehicle integrations. There are a few experimental evaluations with real vehicles, however the number of vehicles is limited. For example, [9, 10, 11] described a field experimental evaluation performed with up to four vehicles.

Network Simulator 3 (ns-3) is an open source programmable network simulator with many capabilities. Direct Code Execution (NS3-DCE, or DCE) is a module for ns-3 to provide the ability to run Linux programs directly in its simulated environment. It enables users to do experiments with their programs in the simulated network environment without doing major source code modifications. DCE supports several types of network software infrastructures, specifically, network protocol stacks Linux. One of them is the DCE-linux protocol stack, which adapts the protocol stack of real-world Linux kernel into DCE. It has much more Layer 3 and 4 facilities than the default ns-3 protocol stack, and allows program running on it to have direct access to the MAC Layer, which is required by CarGeo6. To explain in a technical way, DCE runs several Library Linux Operating Systems [12, 13] in a single process, and connect its networking and timing backend to ns-3 facilities. The user programs can be executed in the simulated library OS efficiently.

3. OBJECTIVE AND APPROACH

Experimentation in real testbed using GeoNetworking implementation can provide very precise evaluation result in

[1]http://www.cargeo6.org/

the deployment phase of the implementation, however it requires heavy cost (*i.e.* vehicles, equipment, drivers, time, etc.). On the other hand, the simulation can provide the evaluation result in flexible networks with various scenarios in a low cost, however the result is based on the preconfigured model in the simulation and often diverse from the experimental evaluation in the real testbed.

Our objective is to investigate realistic behavior of GeoNetworking by simulation in various scenarios. To realize the objective, we take the benefits of both experimentation in real testbed and simulation in the following approach as in Figure 3. In this paper, we combine experimentation in real testbed and simulation. Firstly, we conduct experimentations in real testbed using an open source GeoNetworking implementation (Section 4). Secondly, we reproduce the experimental evaluation result of GeoNetworking in the simulated network using the same implementation (Section 5). Finally, we extend the simulation to a large scale network with various scenarios (Section 6).

The method developed in the paper has three aims. First, the developers of the GeoNetworking implementation can understand the realistic behavior of the software in large scale networks under various scenarios. Second, by understanding the behavior of the implementation, it eases the debugging and the performance improvement of the implementation. Last, it facilitates the development of ITS applications working on the GeoNetworking implementation.

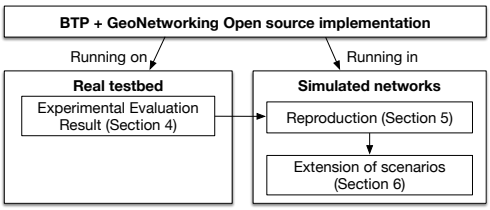


Figure 3: Our approach

4. EXPERIMENTAL EVALUATION IN REAL TESTBED

We performed an experimental evaluation using CarGeo6 version 0.9.9, on 4 ARM boxes with real hardware specifications as listed in Table 1. A mac filter was applied on each node, to limit the neighbour that it could communicate with adjacent nodes only, achieving an up to 3 hop topology as depicted in Figure 4(a).

Table 1: System configuration

Item	Specification
CPU	Dual Core ARM11 600MHz SoC
Memory	128 MB RAM
Storage	16 MB Flash
Kernel	Linux kernel 2.6.35.13
MAC protocol	IEEE 802.11p (ETSI G5)
Wireless Interface	Unex DCMA-86P2

The round-trip times (RTTs) are measured between the source and the destination in the case from single hop to 3 hops with various packet sizes (varying from 20 bytes to 1500 bytes by increasing the size by 20 bytes). The `btpecho` (client mode) sent the BTP echo request 100 times to the `btpecho` (reflector mode) in each test. There is no traffic besides the echo request, echo reply and the beacons during the tests.

Figure 6(a) shows the result of average RTT in the experimental evaluation (In order to save the space of the paper, the figure also shows the uncalibrated reproduction results which is explained in Section 5).

The RTT increases along with the packet size in all tests (from single hop to 3 hops). When the packet size is 20 bytes, RTT on 2 hop and RTT on 3 hop have 1.5 ms and 3.4 ms greater than the one in single hop, respectively; when packet size is 1380 bytes, they are 5.3 ms and 10.5 ms greater. BTP GeoNetworking does not process the packet bigger than the MTU because the fragmentation is not defined in the specification. Therefore all the packets bigger than 1380 bytes were lost in the experiments.

5. REPRODUCTION OF REAL TESTBED RESULT IN SIMULATED NETWORKS

In the last section, we described how the experiments are done in the real testbed. Nevertheless, the real testbed has limitations: high cost, limited scale, inflexible in configuration, etc. In order to overcome these limitations, a realistic simulated environment called Direct Code Execution was employed. With minor and trivial modifications to the CarGeo6 source code as well as some parameter calibrations, we successfully reproduced the real testbed results in the simulated environment. In this section, we describe the successful reproduction of the real testbed results in the simulated environment.

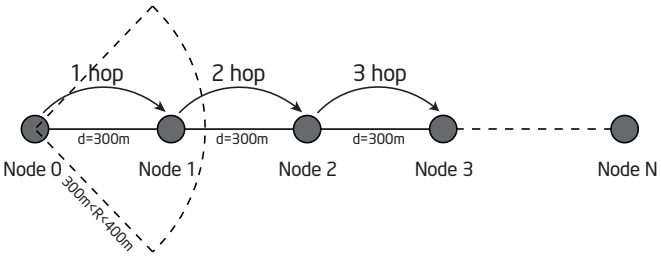
5.1 Simulation configuration

Ns-3 and DCE has many parameters which can be tuned to reproduce the real testbed environment. In order to tune and calibrate the simulator, we use a simple linear topology which is shown in the Figure 4(a): all nodes are configured with the same Wi-Fi parameters, and kept in a same Ad-Hoc cell; each node are in a line with 300m distance to adjacent node. With a negative receiver antenna gain, the wireless radio range is adjusted to 300-400 meters. That means nearly all packets in 300m range can be delivered, yet nearly all packets from 400m away were lost. The configuration ensured each node can and can only reach the adjacent nodes. The detailed configuration in DCE is shown in Table 2.

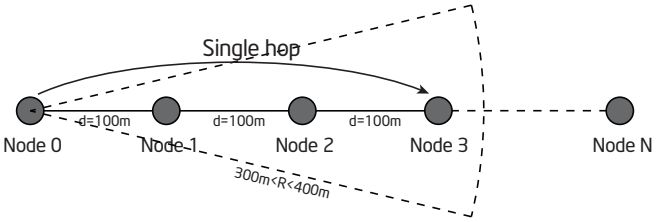
Table 2: DCE network configuration

Item	Specification
Radio Frequency	5.9GHz
Wi-Fi Phy	ERP-OFDM, 6Mbps
Wi-Fi Mac	Ad-Hoc
Receiver Gain	-10dBi
Propagation Delay	Constant
Propagation Loss	Friis
Node Mobility Model	Static

5.2 Modification of GeoNetworking implementation



(a) Topology (ND = 300m)



(b) Topology (ND = 100m)

Figure 4: Topologies

Since the CarGeo6 is not fully compatible with libc and kernel used in DCE, some modifications in the code must be done. To Avoid messing up the original code and keeping backward compatibility, a separate module called *dce_compat* is employed. Most of the compatibility code goes to the compatibility module, and modifications in other modules are minimized and the backward compatibility are maintained. The modifications can be divided into 4 parts:

5.2.1 Socket Incompatibilities

DCE provides a limited subset of socket features. However, several ones which used in CarGeo6 are not supported. Sequenced packet socket, which is one of it, must be modified to Streaming socket, with manually framing. Another one is peeking read in socket, which should be cared manually.

5.2.2 Timing Function Incompatibilities

Another big issue is about timing. DCE does not support monotonic clock, which is considered rigorous in timing. Moreover, minimum resolution of is microsecond instead of nanosecond in modern operating systems. Some workarounds are employed in *dce_compat*. E.g., using real time clock instead of monotonic ones. Consequential bugs in CarGeo6 are fixed.

5.2.3 Lack of Math Functions

Several math functions lacked in DCE are mandatory for geolocation calculation, which includes sine, cosine, arctangent, square root, absolute value, etc. These functions are implemented by hand in *dce_compat* module, and will be contributed to upstream of DCE.

5.2.4 Lack of Other Functions

Besides math functions, DCE lacks some other functions such as BSD compatible functions, some of pthread features, etc. Some of them can be skipped safely, while the others are implemented manually in *dce_compat*.

5.3 Packet Size issue in DCE

During preliminary experiments, we noticed an issue regarding the RTT, which leads to the steps as depicted in Figure 5. We inspected this issue and found it is caused by a hack of link layer packet size, which is located in the Linux kernel of library OS used by DCE[2]. The hack is an attempt to solve an issue in TCP, therefore it is not needed for GeoNetworking reproduction. We had modified the code to bypass the hack, and got perfect results as shown in Figure 6.

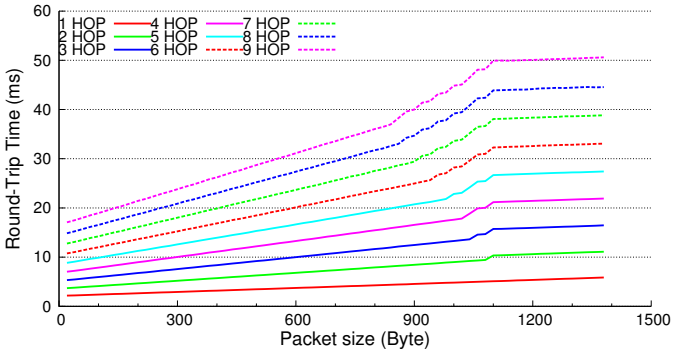


Figure 5: Wrong RTTs cause by Packet Size issue

5.4 Processing Delay in DCE

In networking, processing delay means the time for a device to process a packet, which can affect the result of experiment. In Linux kernel, the cause of processing delay is rather complicated and can be affected by many factors, including task scheduling, interrupt handling, Wi-Fi antenna delay, etc. That means, fully modeling processing delay in DCE is impossible.

The result of CarGeo6 reproduction was greatly impacted by it. An approach must be carried out to calibrate it. DCE has some facilities to model the processing delay of each simulated operating system through the task scheduling. We simplified the model by aggregating other factors into task schedule delay.

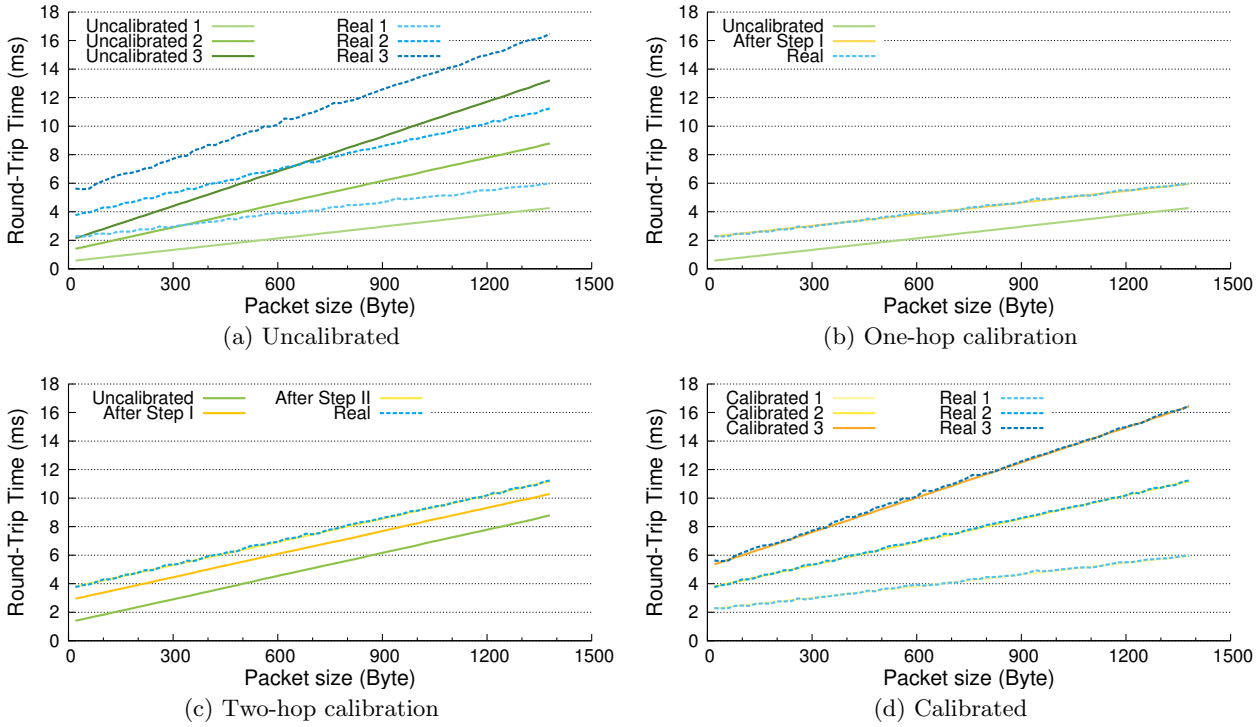
5.4.1 Detection and Analysis

The processing delay issue was first detected in preliminary experiments, when we were trying to reproduce the real testbed results in simulated environment. With the same hardware parameters, our experiments reproduced fairly realistic results, shown in Figure 6(a). However, a constant difference was observed between real and reproduction results.

In the figure, we noted that the delay is approximately in proportion to the number of nodes invoked. By this evidence, the possibility of propagation delay can be ruled out since it is related to number of hops. To classify, there are two types of nodes in these experiments which should be considered separately, as in Figure 2:

Intermediate node only executes `itsnet` program and in charge of packet routing and forwarding.

[2]https://github.com/direct-code-execution/net-next-sim/blob/sim-ns3-3.14.0-branch/arch/sim/sim-device.c#L130



(a) Uncalibrated

(b) One-hop calibration

(c) Two-hop calibration

(d) Calibrated

Figure 6: Processing Delay Calibration Steps

Terminal node invokes `itsnet` and `btpecho`, has evidently more work than **intermediate node**.

Define D_n as the total observed delay in n-hop case ($n+1$ nodes in total), D_T, D_I as the proportion of observed value from **terminal node** and **intermediate node** respectively, while P_T, P_I for the parameters of internel task scheduling delay in DCE.

In the configuration of in Figure 4(a), the relationship among D_n, D_T and D_I should follow equation 1:

$$D_n = 2 \times D_T + (n-1) \times D_I \quad (1)$$

5.4.2 Calibration

Several steps are taken carefully to calibrate the processing delay in the nodes:

Pre-calibration Calculation Before calibration, we did a refined measurement and calculation on the observed difference in each experiment, which is shown in Figure 6(a). The result was slightly different from proportional expectation: $D_1 = 1700$ us, $D_2 = 2450$ us, $D_3 = 3200$ us. Thus $D_T = 850$ us, $D_I = 750$ us. With a preliminary test, we found there were no clear relationship between P_T and D_T, nor P_I and D_I.

Step I: Calibration of terminal node Calibrate the P_T on single-hop configuration. We did a binary search for the P_T, and evaluate the difference between simulation and testbed results. Optimal value were found at **200 microseconds**: in Figure 6(b), the simulation line overlaps with the testbed one.

Step II: Calibration of intermediate node After Step I, the shift distance of the simulation result of two-hop configuration is the same as one hop, as shown in Figure 6(c). Calibrate the P_I on two-hop configuration, with $P_T = 200$ us. Another binary search was carried out to find the optimal value: **540 microseconds**.

Post-Calibration Verification Verify the three-hop result with $P_T = 200$ us and $P_I = 540$ us, as shown in Figure 6(d). The simulation line is close to the testbed one, which verifies our conjecture. The fact that $P_T << P_I$ while $D_T > D_I$ is reasonable: the **terminal nodes** have heavier load than **intermediate nodes**, the Kernel scheduling was done more time, thus the delay should be lower.

After proper calibration, the results practically overlap the real ones, with maximum absolute error of 60 microseconds. The results suggest that our reproduction with calibration is credible, thus it can be extended to large scale network scenarios.

6. EVALUATION IN LARGE SCALE AND FLEXIBLE SIMULATED NETWORKS

In this Section, we provide preliminary evaluation results regarding the performance of the implementation from the point of RTTs, with varied node distances,packet sizes and number of hops. We define Node distance (ND) as the distance between adjacent nodes, Terminal Distance (TD) as distance between destination and source nodes.

With the calibration made in Section 5, we conducted several experiments. First, experiments with customized number of hops were conducted to evaluate the Greedy Forwarding algorithm, which shows that the algorithm successfully to choose a multi-hop route to forward packets up to 9 hops.

Then, *ND* and *TD* are modified to examine their impact on the network, and we found there could be a extremely high packet loss under certain conditions.

6.1 Extended scenarios

We successfully extended the experiments in NS3-DCE with the topology as depicted in Figure 4(a), with any desired number of nodes (N), and varied *NDs* which is difficult under real testbed with limited manpower and resources. The topology with *ND* of 100m is shown in Figure 4(b), when it comes to 10m or 50m, the topologies will be slightly different. Thus we can examine how packet size and number of hops affect the results in reproductions, and whether the implementation can be properly functional with varied *NDs* and *TDs*. Therefore, the prediction of realistic behavior of the implementation is viable.

We first measured the network delay perceived by the `bt-pecho` initiator with packet size range from 20 bytes to 1380 bytes in a single hop. Then, extend to 2, 3 and finally 9 hops, and repeat the first experiment. Finally, compare and analyse the data obtained in reproduction to find out how packet size and number of hops impact the network delay. For all delay measurements we measured 1000 BTP echo Request RTTs between the two terminal nodes with interval time of 0.5 seconds.

6.1.1 More hops

RTTs of different hops and packet sizes were obtained from reproduction, as depicted in Figure 7. The RTT increases as the packet size increase as we have in the previous sections. With the packet size of 20 bytes, the RTT increases by 1.55ms each hop increase. When the packet size comes to 1380 bytes, the increment of RTT is 5.23ms each hop.

6.1.2 Different node distances

We select some data with some specific *NDs* and fixed packet size of 80 bytes, but with varied *TDs*, as depicted in Figure 8. Figure 4(b) depicts a specific scenario with $ND =$ 100m. The impact of *ND* and *TD* on realistic behavior of the implementation can be predicted through the data.

With the 10m *ND*, it can be observed that RTTs are constant regardless of *TD*. It indicates the *GF* algorithm selected the terminal node directly, thus the number of hops is 1. With the 100m *ND*, when *TD* rises, a notable rise of RTT can be observed when TD rises from 300m to 400m, which indicates that the *GF* algorithm worked in the reproduction to forward the packets via a multi-hop route when the destination is out of its radio range. With the 300m *ND*, a tendency can be observed that, the network delay presents a perfect linear rise, with the growth of *TD*, which can be considered as the number of hops with such large enough *ND*. The result shows the tendency that how hop of the route changes with different *NDs* and *TDs*.

$$MinimalRequiredHops = \lceil TD/RadioRange \rceil \quad (2)$$

6.2 Packet loss issue in Greedy Forwarding algorithm

During experiments, a packet loss was detected when *Node 0* try to send BTP echo request to *Node 4*, same is the *Node 0* try to response, with a particular scenario that 4 nodes in a line, *ND* is 100m and *TD* is 400m. In this Section, we discuss the cause of the packet loss, and how to quantify it.

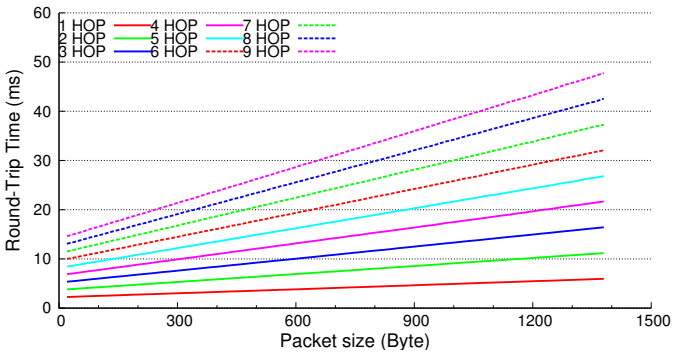


Figure 7: Average RTT in Simulation

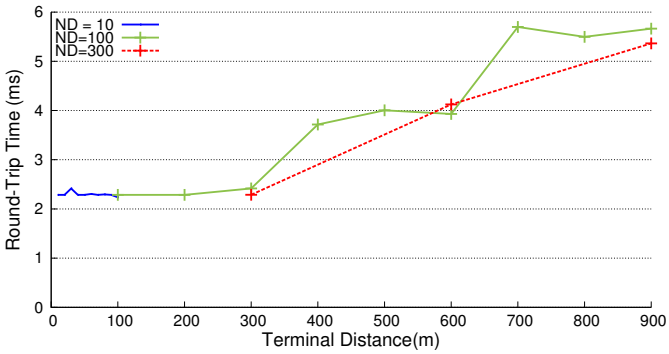


Figure 8: RTT-Distance

6.2.1 High PLR(Packet Loss Ratio) in long distance

As described in Section 5, *Friis Propagation Loss Model* is employed in the reproduction, making the communication channel unstable; this is the scenario we've not realized in the real testbed yet. In this reproduction, the maximum stable transmitting radius of a node is in the range from 300m to 400m. Thus *Node 4* is at the edge of the transmitting radius of *Node 0*, which means the communication channel between them is unstable. With an extremely unstable channel, beacon messages from *Node 4* to *Node 0* are unexpected. An unexpected beacon makes a location table entry that can hardly be reached as in Figure 9.

6.2.2 Analysis

As depicted in Figure 9. Consider the following scenario: *Alice* (Node 0) wants to communicate with *Bob* (Node 4), the channel between them is unstable due to the distance or obstacle. But *Carol* (an intermediate node) has good communication channels with *Alice* and *Bob* without message loss. The three repeatedly broadcast beacon message to inform each other their existence. However *Alice* still can receive some beacon messages from *Bob* randomly with a loss ratio of PLR_B. When *Alice* gets any message (usually a beacon message) directly sent from *Bob*, she will instantly label *Bob* as her *Neighbour* in her *LocT* as an entry with a default lifetime (*T(LocTE)*) of 20s. And then, *Alice* will directly send messages to *Bob*; the packet could be lost, with a possibility of PLR_P. Otherwise, in order to reach *Bob*, *Alice* will deliver her message to *Carol*, letting her to forward the

message to *Bob*, with no loss.

In a particular reproduction, *Alice* received 41 out of 3254 unexpected beacon messages from *Bob*, which is founded in the log file. 4 among the 41 total beacon messages from *Bob* is encountered during the BTP echo operation. Remember, PLR_P is the loss ratio of BTP echo packet, which is supposed to be higher than PLR_B due to different packet length. To simplify, assume that $PLR_P = 0$, and the unexpected beacon message effective periods(20s) on both sides do not collide or overlap with each other. With the BTP echo interval of 0.5s, there is supposed to be 40 packet losses every unexpected beacon message encountered. Thus, there is supposed to be $4 \times (20/0.5) = 160$ packet losses in the experiment, and it is in accordance with the result. With N as the total number of beacons from a sender (SE), N_d as the number of delivered beacons from a sender (SE). Then, do an preliminary calculation on the beacon packet lost ratio in equation 3:

$$PLR_B = 1 - (N_d/N) \tag{3}$$

With $N_d = 41$, $N = 3254$, we get a loss ratio of 98.7%.

In a statistics point of view, assume f_B as the frequency of beacon message, T as the lifetime of LocTE, then we have the expectation of loss ratio in total:

$$total\ loss = PLR_P \times (1 - PLR_B^{(T \times f_B)}) \tag{4}$$

To simplify again, assume $PLR_P = 100\%$ when PLR_B is close to 100%. Hence the loss ratio is only connected with f_B, PLR_B and T. According to the GeoNetworking implementation, each node broadcasts a beacon message every a little bit longer than 3 seconds. With $f_B = 1/3$ Hz, $PLR_B = 0.987$, $T = 20$ s, then we have the loss ratio of 8.35%. Recall the simplification we made, the observed value may be slightly different. The observed value of 7.3% in the reproduction proved this point. According to the Equation 4, the packet loss could be extremely high under certain condition as depicted in Figure 9. Another reproduction proved it, and it will be discussed in further research.

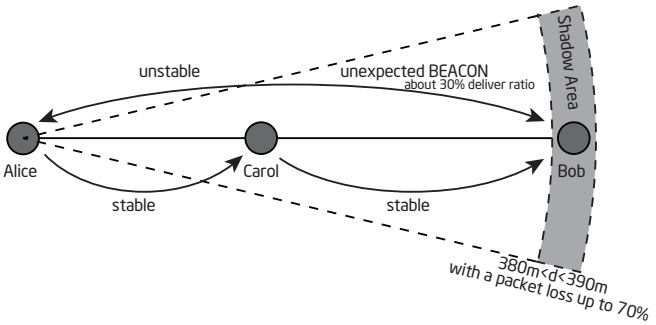


Figure 9: Unexpected Beacon loss

7. CONCLUSIONS AND FUTURE WORKS

In this paper, we have successfully reproduced real testbed experiments in simulated network environment using NS3-DCE. With the proper wireless configuration and the calibrations, the reproduction produced realistic results, which can be used to predict the behavior of GeoNetworking implementation in real world. Based on the fact, several extended experiments of the GeoNetworking was conducted with NS3-DCE. The results indicate that, the implementation succeeded in delivering packets up to 9 hops with any desired number of nodes; meanwhile the *GF* algorithm functions properly; yet a packet loss is observed in configurations of a critical communication distance. Finally we quantified the packet loss, and found it is only related to 4 factors: beacon frequency, lifetime of *LocTE* and PLR of beacon and BTP; moreover, it could be extremely high under some certain conditions.

As a future work, we consider the followings: First, more factors should be introduced to the reproduction *E.g.*, a moving mobility model, a complicated 2D distribution of nodes, other communications modes. Second, IPv6 over GeoNetwork should be ported to NS3-DCE for more extended experiments. Last, more efficient routing strategy can be evaluated in the simulated networks, to solve the packet loss issue discovered in the extended scenarios. it is worth discussing whether the current routing strategy is the best and most efficient.

8. REFERENCES

[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, July 2014.

[2] ISO 21217:2010 Intelligent transport systems – Communications access for land mobiles (CALM) – Architecture, April 2010.

[3] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *6th Annual International Conference on Mobile Computing and Networking, MobiCom 2000, August 6.-11., 2000, Boston, Massachusetts, USA*, pages 243–254. ACM / IEEE, August 2000.

[4] Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol, August 2014.

[5] Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 6: Internet Integration; Sub-part 1: Transmission of IPv6 Packets over GeoNetworking Protocols, May 2014.

[6] Thouraya Toukabri, Manabu Tsukada, Thierry Ernst, and Lamjed Bettaieb. Experimental evaluation of an open source implementation of IPv6 GeoNetworking in VANETs. In *ITST 2011 : 11th International Conference on Intelligent Transport System Telecommunications*, Saint-Petersburg, Russia, August 2011. Conference is technically co-sponsored by IEEE Communications Society and co-organized by the Technical Sub-Committee on Vehicular Networks and Telematics (VNAT).

[7] ZiyaCihan Taysi and AliGokhan Yavuz. Etsi compliant geonetworking protocol layer implementation for ivc simulations. *Human-centric Computing and Information Sciences*, 3(1), 2013.

[8] Victor Sandonis, Ignacio Soto, Maria Calderon, and

Manuel UrueÃśa. Vehicle to internet communications using the etsi its geonetworking protocol. *Transactions on Emerging Telecommunications Technologies*, pages n/a–n/a, 2014.

[9] J.J. Anaya, E. Talavera, F. Jimenez, J.G. Zato, N. Gomez, and J.E. Naranjo. Geonetworking based v2v mesh communications over wsn. In *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pages 2421–2426, Oct 2013.

[10] Manabu Tsukada, José Santa, Satoshi Matsuura, Thierry Ernst, and Kazutoshi Fujikawa. AnaVANET: an experiment and visualization tool for vehicular networks. In *9th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2014)*, Guangzhou, China, May 2014.

[11] Manabu Tsukada, José Santa, Satoshi Matsuura, Thierry Ernst, and Kazutoshi Fujikawa. On the experimental evaluation of vehicular networks: Issues, requirements and methodology applied to a real use case. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 14(1), 12 2014.

[12] D. Camara, H. Tazaki, E. Mancini, T. Turletti, W. Dabbous, and M. Lacage. DCe: Test the real code of your protocols and applications over simulated networks. 52(3):104–110, 2014.

[13] Hajime Tazaki, Frédéric Uarbani, Emilio Mancini, Mathieu Lacage, Daniel Camara, Thierry Turletti, and Walid Dabbous. Direct code execution: Revisiting library os architecture for reproducible network experiments. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 217–228, New York, NY, USA, 2013. ACM.