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 shows the strength in the network with dynamic topology compared with topology based routing.

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 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.

Figure 1: GeoNetworking in ITS Station Architecture
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. 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.

GeoNetworking [1] is standardized by ETSI as a network protocol [15]. The present document enables the usage of NEMO BS over the ETSI GeoNetworking protocol [i.25] and an IPv6-compliant protocol layer and extended with mobility extensions.
layer protocol as in Figure 1, integrating several geo-aware strategies including Greedy Forwarding (GF) [3] 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 IP6 Adaptation Sub-Layer (CN6 ASL) [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.

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.

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 simulation 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. 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

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 experiments 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).

Figure 3: Our approach

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.

4. EXPERIMENTAL EVALUATION IN REAL TESTBED

We performed an experimental evaluation using CarGeo6 version 0.9.9 on the real hardware where the specifications are listed in Table 1. The configuration of the network is similar to the topology as shown in Figure 2, with up to four nodes in order to obtain maximum 3 hop topology.

### Table 1: System configuration

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

\(^1\)http://www.cargeo6.org/
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 5(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 simulation 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 simulation 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 simulation environment. In this section, we describe the successful reproduction of the real testbed results in the simulation 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: 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.

![Figure 4: Topology of 300m distance](image)

Table 2: DCE network configuration

<table>
<thead>
<tr>
<th>Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi-Fi Standard</td>
<td>IEEE 802.11g</td>
</tr>
<tr>
<td>Wi-Fi Phy</td>
<td>ERP-OFDM, 6Mbps</td>
</tr>
<tr>
<td>Wi-Fi Mac</td>
<td>Ad-Hoc</td>
</tr>
<tr>
<td>Receiver Gain</td>
<td>-10dBi</td>
</tr>
<tr>
<td>Propagation Delay</td>
<td>Constant</td>
</tr>
<tr>
<td>Propagation Loss</td>
<td>Friis</td>
</tr>
<tr>
<td>Node Mobility Model</td>
<td>Static</td>
</tr>
</tbody>
</table>

5.2 Processing Delay in DCE

In networking, processing delay means the time for a device to process a packet, which can affect the result of experiments. 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 simulation 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.

5.2.1 Detection and Analysis

The processing delay issue was first detected in preliminary experiments, when we were trying to reproduce the real testbed results in simulation environment. With the same hardware parameters, simulation produced fairly realistic results, shown in Figure 5(a). However, a constant difference was observed between real and simulation 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 hop. 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.

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), \(D_r, D_t\) as the proportion of observed value from terminal node and intermediate node respectively, while \(P_r, P_t\) for the parameters of internal task scheduling delay in DCE.

In the configuration of in Figure 4, the relationship among \(D_n, D_r\) and \(D_t\) should follow equation 1:

\[
D_n = 2 \times D_r + (n - 1) \times D_t
\]  

5.2.2 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 5(a). The result was slightly different from proportional expectation: $D_1 = 1600$ ms, $D_2 = 2350$ ms, $D_3 = 3100$ ms. Thus $D_T = 800$ ms, $D_I = 750$ ms. 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 **190 microseconds**: in Figure 5(b), the simulation line overlaps with the testbed one.

**Step II: Calibration of intermediate node** After Step II, the shift distance of the simulation result of two-hop configuration is the same as one hop, as shown in Figure 5(c). Calibrate the $P_I$ on two-hop configuration, with $P_T = 190$ us. Another binary search was carried out to find the optimal value: **480 microseconds**.

**Post-Calibration Verification** Verify the three-hop result with $P_T = 190$ us and $P_I = 480$ us, as shown in Figure 5(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 simulation with calibration is credible, thus it can be extended to large scale network simulations.

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, experimentations 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 experimentation in NS3-DCE, with any desired number of nodes, and varied $ND$s which is difficult under real experimentation with limited manpower and resources. Thus we can examine how Packet size and Number of hops affect the results in Simulation, and whether the implementation can be properly functional with varied $ND$s and $TD$s. Therefore, the prediction of realistic behavior of the implementation is viable.

We first measured the network delay perceived by the bt-
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 experimentation. Finally, compare and analyse the data obtained in simulation 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 simulation, as depicted in Figure 6. The RTT increases as the packet size increase as we have in the previous sections. However the constant increase ends when packet size is 1080 bytes in 2 hops scenario, and it is observed from smaller packet size in more hops. With the packet size of 20 bytes, the RTT increases by 1.87ms each hop increase. When the packet size comes to 1380 bytes, the increment of RTT is 5.54ms 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 7. The impact of ND and TD on realistic behavior of the implementation can be predicted through the data.

We use the same linear topology but the ND of 10m; observed that RTTs are constant regardless of TD. It indicates the GF algorithm selected the terminal node directly.

With the 100m ND, when TD rises, notable steps can be observed, which indicates that the GF algorithm worked in the simulation to forward the packets via a multi-hop route. 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. It indicates that, the Nodal delay (the sum of all latency delays of a node) of an intermediate node is constant in a fixed packet size.

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.

6.2.1 Low PDR in long distance

As described in Section 5, Friis Propagation Loss Model is employed in the simulation, making the communication channel unstable; this is the scenario we’ve never realized in the real tested. In this simulation, 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 8.

6.2.2 Analysis

As depicted in Figure 8. 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 deliver ratio of $P_{DR}$. 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 hit Bob, with a possibility of $P_{DR}$. 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 the particular simulation, 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, $P_{DR}$ is the deliver ratio of BTP Echo packet, which is supposed to be smaller than $P_{DR}$ due to different packet length. To simplify, assume that $P = 0$, 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 encounter. Thus, there is supposed to be $4 	imes (20 / 0.5) = 160$ packet losses in the experimentation, and it is in accordance with the result. With $N$ as the total number of beacons from a sender (SE), $Nd$ as the number of delivered beacons from sender (SE).
Then, do an preliminary calculation on the beacon packet lost ratio in equation 2:

$$P_{DRB} = N_d / N$$

With $N_d = 41$, $N = 3254$, we get a deliver ratio of 1.3%.

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:

$$total\ loss = (1 - P_{DRP}) \times (1 - (1 - P_{DRB}))^{(T \times f_B)}$$

To simplify again, assume $P_{DRP} = 0$ when $P_{DRB}$ is close to 0. Hence the loss ratio is only connected with $f_B$, $P_{DRB}$ 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, $P_{DRB} = 0.013$, $T = 20$ s, then we have the loss ratio $P_l$ of 8.35%. Recall the simplification we made, the observed value may be slightly different. The observed value of 7.3% in the simulation proved this point. According to the Equation 3, the packet loss could be extremely high under certain condition as depicted in Figure 8. Another simulation proved it, and it will be discussed in further research.

8. REFERENCES


