Rapid Prototyping of Cooperative Caching in a VANET: A Case Study

Steve Glass, Imad Mahgoub, and Monika Rathod
Department of Computer and Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, USA
Email: sglass2@fau.edu, mahgoubi@fau.edu, mrathod@fau.edu

Abstract—**Due to the complex nature of mobile ad hoc networks, simulation is an important tool used to explore approaches and validate research. De facto simulation tools for networking research such as ns-3 are extremely powerful. However, the time to develop, execute, and analyze a meaningful simulation in these tools can be significant. In addition to the unavoidable learning curve associated with the tool itself, the developer must have significant coding skills and domain knowledge. Once development of the simulation is complete, the time required to execute the simulation can be significant. In this paper, we present a case study that focuses on the use of the Netlogo tool to explore the impact of certain parameters on a cooperative cache deployed within a vehicular network. The results of the Netlogo simulation are then compared to the results of a similar simulation developed in ns-3. Our work shows that meaningful results can be obtained using Netlogo with less investment of time at the expense of flexibility and realism when compared to ns-3. With this approach, if the results of the Netlogo based simulation look promising, a greater investment in a more powerful and accurate simulation using ns-3 can be made. Utilizing this process saves the developer time by focusing on those approaches that look promising.**

***Keywords*—cooperative caching, privacy, vehicular ad hoc network (VANET), Netlogo, ns-3**

I. Introduction

In our work, we desired a process and a set of tools which would allow us to rapidly prototype ideas and gain meaningful insight into the impact of those ideas before investing significant time in a more powerful, realistic simulation. Specifically, we were exploring the use of a cooperative cache deployed within a vehicular network with the goal of improving privacy.

In this paper, we capture the process that we used which utilizes the Netlogo simulation tool for rapid prototyping of our ideas. We found that the Netlogo tool requires much less investment of development time than ns-3. However, within the networking domain, Netlogo is much less flexible and realistic. Therefore the penalty is that the results of a Netlogo simulation may not be as accurate as the ns-3 simulation. Our thought is to initially trade investment for accuracy as long as the results demonstrate similar patterns. If the results of the Netlogo simulation show promise, additional time can be invested in a more accurate simulation. The results of our work show that meaningful information can be gathered from the Netlogo based simulation with less effort.

The remainder of the paper is organized as follows. Section II provides a short introduction to cooperative caching and the ideas that we explore. Section III captures our basic approach to rapid simulation using Netlogo and ns-3. Section IV captures and compares the results of our simulation. Section V concludes the document.

II. Exploring VANET Based Cooperative Caching

A. Properties Of VANETs

Vehicular networks allow for direct data communication between vehicles and communications between vehicles and fixed Road Side Units (RSUs). In addition to the various challenges found in the more general mobile ad hoc networks, vehicular networks present additional challenges such as the potential for the network to cover a large geographic region, many more nodes within the network, and a very dynamic network topology.

Due to these challenges, it is very difficult to validate research on a real network. In many cases, researchers turn to simulation to test proposed approaches.

B. Cooperative Caching

In our case, we were interested in exploring techniques to improve privacy in vehicular networks. One way to improve privacy is to avoid sending requests for data items to a central server whenever possible. A central server may represent a single point where an attacker can focus their efforts.

One published approach titled ”Hiding in the Mobile Crowd” leverages a cooperative cache to improve privacy [1]. A cooperative cache is a cache that is distributed throughout the vehicles that exist in the mobile network. When a vehicle needs a particular data item, it will first ask the neighboring vehicles if they have the data. If so, the neighboring vehicle will reply with the data. If not, the request will be sent to a data server on the fixed network. When a vehicle receives a data item from the server, it will store it in it's local cache in case another vehicle needs the same data item in the near future.

By distributing the data items throughout the network, in many cases vehicles can obtain data items from other vehicles without ever exposing the need for the data to a server on the fixed network.

C. Caching In VANETs

Our desire was to have the capability to rapidly evaluate a hypothesis with respect to improving privacy in a vehicular network via cooperative caching. For example, does the number of vehicles in the network have a significant impact on the Hiding Probability? How about the communication range of the vehicles themselves? Which has a greater impact?

Like "Hiding in the Mobile Crowd", we use Hiding Probability as our metric. However, in our case the Hiding Probability is simply the percentage of vehicles that were not exposed to the server. For example, if we have 100 vehicles and during the course of distributing data, the server sees 10 of the vehicles, the Hiding Probability would be 90%. In other words, 90% of the vehicles remained hidden from the server.

III. Rapid Prototyping

A. Development Cycle

The first step in our process is to hypothesize that a particular approach or modification may impact privacy in a positive way. In our case, the thought was that more vehicles and a reduced transmission range may be beneficial. In order to determine the potential impact, the next step is to implement a simulation of the cooperative cache in the vehicular network. Finally, the simulation is executed and the results analyzed.

We wanted the process of implementing and executing models to be efficient in order to cater for the rapid, initial validation of various hypotheses. If a particular approach looked promising, we wanted to be able to then implement that approach in a more detailed simulation. This meant having a simulation tool that was easy to use, did not require significant code development, and executed quickly. Ease of use and reduced code development speeds up the implementation step of the process. Executing quickly speeds up the execution step of the process.

References [2] and [3] propose Agent-Based Simulation and, specifically, Netlogo as an alternative to other network focused simulations. Netlogo is a generic, Multi-Agent simulation tool that looked promising for our needs. Our goal was to determine if Netlogo could be used to quickly implement and test the impact of the number of vehicles and the communication range of the vehicle on the Hiding Probability while still demonstrating patterns similar to a more accurate and costly ns-3 simulation. If this proved to be the case, it would validate the potential for utilizing Netlogo in early stages of research and simulation.

Rather than simply utilizing Netlogo as another simulation tool, our desire was to treat Netlogo as a complementary tool that could be used to rapidly simulate high level concepts before implementing in more complex simulation tools.

B. Netlogo

Netlogo is a tool that has been used to develop simulations in fields that include: Art, Biology, Chemistry & Physics, Computer Science, Earth Science, Mathematics, Networks, Social Science, and System Dynamics (refer to the Netlogo Models Library for more details) [4]. It is especially useful to demonstrate emergent behavior in groups of agents that act independently with some level of uncertainty built in.

We found the Netlogo language to have a fairly shallow learning curve. The Netlogo tool can be thought of as an overall Integrated Development Environment (IDE). A screenshot of the tool is captured in Fig. 1.

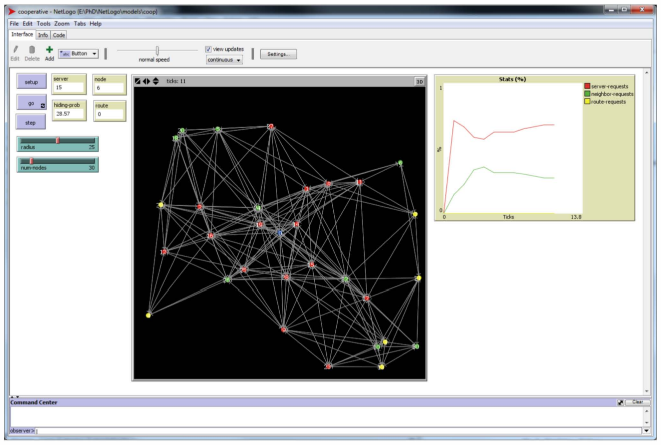

Fig. 1. Netlogo IDE

The IDE focuses on a graphical representation of the simulation. The IDE also enables the deployment of various controls to be used as inputs and mechanisms to display output (such as monitors and plots). The graphical representation makes it very easy to observe and analyze the simulation as it executes.

Our experience was that the development effort was relatively low however the flexibility of the language was less than that of a more complex networking focused simulator that allows for the development of lower level constructs (such as IP packet construction, etc.). With respect to execution time, we found that even simulations with a large number of permutations ran in a reasonable amount of time.

In addition, Netlogo has a users group and a models library included that proved very useful in understanding how to approach particular problems.

C. ns-3

The ns-3 tool is a full featured network simulator that is heavily utilized by networking researchers worldwide [5]. The simulator is written in C++ and simulations can be written using C++ and/or Python. In general, it seems that ns-3 does have a fairly steep learning curve. It requires developers to have a working knowledge of C++, Object Oriented Programming techniques, and detailed networking constructs.

Unlike Netlogo, ns-3 does not include a comprehensive IDE. Most simulations are coded using a simple text editor and built using a simple terminal. For our development, we did use PyVis which allowed us to visualize the basic simulation equivalent to the geographic region in Netlogo. It should be noted that PyViz seems to still be maturing [6].

The overall development effort to create a basic simulation focused on the higher level aspects of interest was significant. However, the tool provided much more flexibility and realism. The execution time was significant when running a simulation with a large number of vehicles and a number of permutations. A very active users group supports ns-3.

D. Comparison

Table I summarizes the initial comparison between Netlogo and ns-3.

TABLE I. SIMULATOR COMPARISON

Criteria	Netlogo	ns-3
Networking Focused		✓
Learning Curve	Shallow	Steep
IDE Support	✓	
Development Effort	Low	High
Networking Flexibility	Low	High
Execution Time	Faster	Slower
Support (Examples, User Group, etc.)	✓	✓

IV. SIMULATION

A. Settings

Table II captures the various parameters used in our Netlogo and ns-3 simulations.

In the next section, we capture the results of our cooperative cache simulations using Netlogo and ns-3. The goal of the simulations is to demonstrate the development effort and execution times while comparing the results of the simulations.

TABLE II. SIMULATION PARAMETERS

Parameter	Setting
Simulation Area	1000 x 1000
Number Of Nodes	20 to 200 (Step 20)
Transmission Range	50 to 500 (Step 50)
Request Rate	10%
Mobility Model	Random Walk
Vehicle Speed	13.4112 m/s
Number of Repetitions	10

A few of the parameters may require additional clarification. The area of the simulation is considered 1000m x 1000m. The number of nodes and the transmission range (in meters) are varied through the simulations. For example, the number of vehicles used in the simulations is in increments of 20 (20, 40, 60, and so on). The request rate is the rate at which vehicles request the data. The number of repetitions is the number of times one particular scenario is run.

Since our goal was to compare the simulation tools with respect to investment and the relative accuracy of the results, we opted to utilize a simple random walk mobility model. Although this does not reflect the mobility patterns that may be found in a typical vehicular networking scenario, it is sufficient for our evaluation of the tools and our process.

B. Results

First the simulation was run with all combinations of number of vehicles and the transmission range of the vehicles. The results for the Netlogo simulation are captured in a contour plot (Fig. 2). The results for the ns-3 simulation are also captured in a contour plot (Fig. 3).

As can be seen, in general the basic patterns seen on the Netlogo contour plot are similar to those seen on the ns-3 plot. There are various bands of hiding probabilities as the transmission range increases. Also, as the number of vehicles

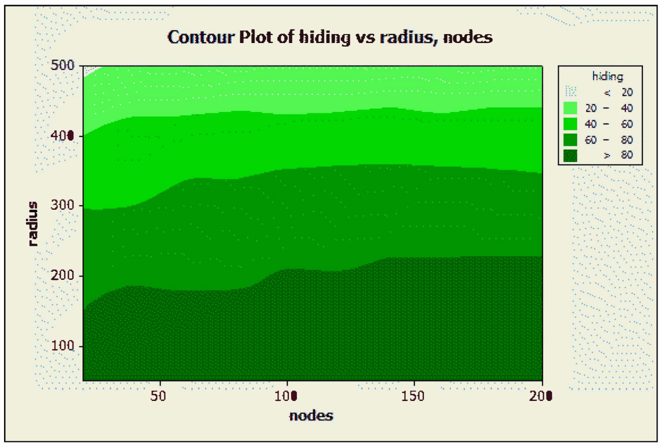

Fig. 2. Hiding Probability - Netlogo

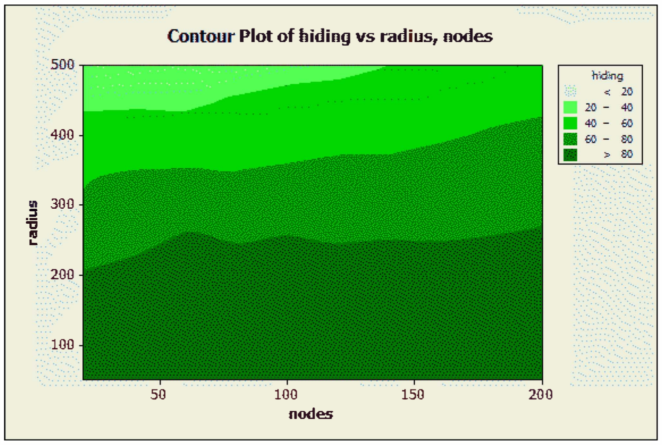

Fig. 3. Hiding Probability - ns-3

increases, there seems to be a slight trend upward in hiding probability.

Next we ran experiments varying the number of vehicles with a fixed range (250 meters) and varying the range with a fixed number of vehicles (100). The results of those experiments are captured in scatterplots (Fig. 4, Fig. 5, Fig. 6, and Fig. 7). Note that Hiding Probability Node (HPN) in the title of the scatterplots indicates the Hiding Probability with a fixed number of nodes. Hiding Probability Range (HPR) indicates Hiding Probability with a fixed transmission range.

When comparing the results of the Netlogo and ns-3 simulations when the number of vehicles are varied, we see that they result in a very similar pattern. In general, as the number of vehicles increase, the hiding probability increases slightly.

When comparing the results of the Netlogo and ns-3 simulations when the range is varied, we again see that they result in a very similar pattern. However, in this case, the communication range has a much more significant impact on the hiding probability.

Table III captures an indication of the development effort reflected by the number of lines of code it took to develop the simulation and the approximate amount of time the simulation

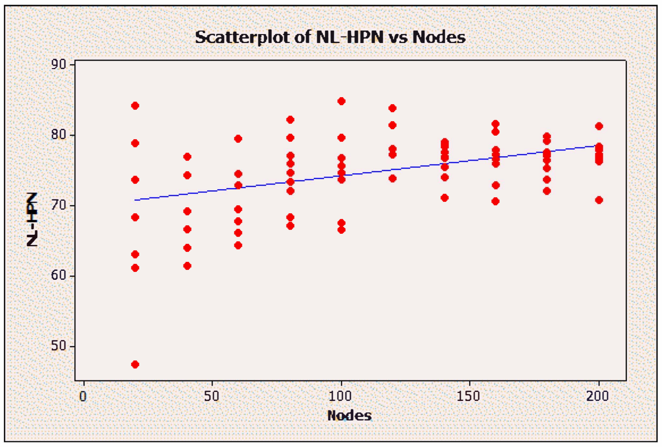

Fig. 4. Fixed Range - Netlogo

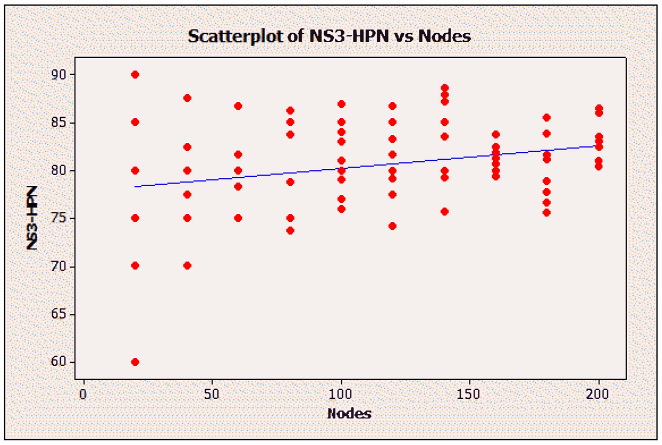

Fig. 5. Fixed Range - ns-3

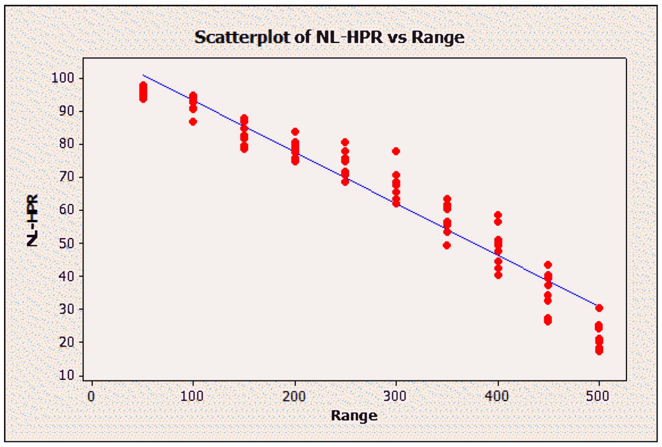

Fig. 6. Fixed Nodes - Netlogo

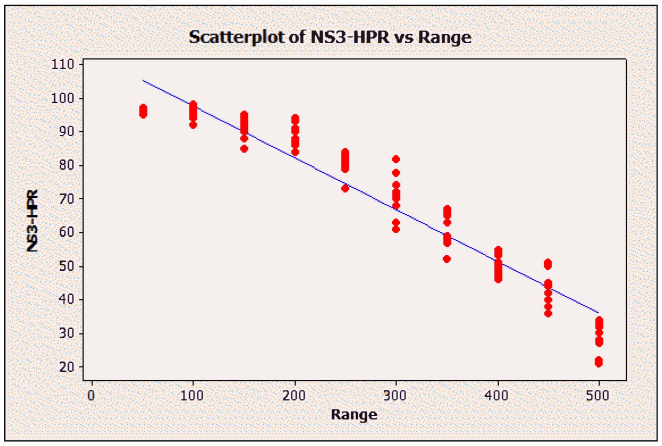

Fig. 7. Fixed Nodes - ns-3

took to execute on an i7 based laptop running Ubuntu 14.04 LTS with a 500GB SSD and 12GB of RAM.

TABLE III. SIMULATION METRICS

Tool	Lines Of Code	Execution Time
Netlogo	109	0 hours 12 minutes
ns-3	1077	5 hours 7 minutes

As can be seen, there is a significant difference in the number of lines of code required to develop the basic simulation and, more importantly, the time to execute the simulation.

V. CONCLUSION

In this paper, we have presented the results of our effort to find a process that allows us to rapidly prototype enhancements to a basic vehicular network based cooperative cache with little investment and meaningful results. Our results show that with the appropriate level of abstraction, Netlogo can be successfully utilized in the early stages of simulation to identify promising areas to explore with less effort and time when compared to a more powerful ns-3 simulation. These tools can be used to complement one another in certain scenarios.

ACKNOWLEDGMENT

This work is part of the Smart Drive initiative at Tecore Networks lab at Florida Atlantic University.

REFERENCES

[1] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J.-P. Hubaux, "Hiding in the mobile crowd: Location privacy through collaboration," *Dependable and Secure Computing, IEEE Transactions on*, vol. 11, no. 3, pp. 266–279, 2014.

[2] M. Niazi and A. Hussain, "Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks," *IEEE Communications Magazine*, vol. 47, no. 3, pp. 166–173, 2009.

[3] M. Babiš and P. Magula, "Netlogoan alternative way of simulating mobile ad hoc networks," in *Wireless and Mobile Networking Conference (WMNC), 2012 5th Joint IFIP*. IEEE, 2012, pp. 122–125.

[4] Netlogo homepage. [Online]. Available: https://ccl.northwestern.edu/netlogo/

[5] Ns3 homepage. [Online]. Available: https://www.nsnam.org/

[6] Pyviz homepage. [Online]. Available: https://www.nsnam.org/wiki/PyViz