MASTER

OMNeT++ based model for real-time MANET simulations

Narayan, G.

Award date:
2014

Link to publication
OMNeT++ BASED MODEL FOR REAL-TIME MANET SIMULATIONS

Master Thesis

GOVIND NARAYAN

Supervisors:
Dr. Sander Stuijk
Dr. Julio de Oliveira Filho

Version 1.0

Eindhoven, August 29, 2014
Abstract

Application developers and architects of large wireless networks often need to test and validate their solutions to meet the performance requirements such as throughput, reliability and latency. Deployment of such large wireless networks for testing is often not feasible at the initial stages of development cycle due to large investment of time and money. As an alternative, designers use the hardware in loop (HIL) simulation approach to understand the real effect of the network behavior on the application under development. HIL simulators tend to become slow for large scale network simulation and thus do not operate in real time. In this work we provide solution to this problem by developing real time hardware in loop model for the OMNeT++ simulator capable of simulating large scale mobile ad hoc networks. The developed model uses an abstract implementation of the physical layer and the wireless channel to reduce the computational load of the simulator. The experiments conducted on our model show that it is capable of operating in real time for a data rates up to 16 kbps with 100 nodes. For the investigated scenarios and boundary conditions the relative error falls within 2 to 8%.
Acknowledgement

I would like to thank TNO and its employees for providing me friendly and motivating environment for my research. Special thanks to my supervisors Dr. Sander Stuijk and Dr. Julio de Oliveira Filho for their valuable guidance, critical insight and encouragement throughout my project. Furthermore, I would like to thank Dr. Twan Basten for introducing me to this project. Thanks to Rob Boekema for the knowledge transfer during the initial phase of the project. I am forever grateful to my parents for their endless love and support. Finally, I would like to thank one and all who helped me directly or indirectly in completing the project. This work was carried out as part of COMMIT project: Sensor Networks for Public Safety – SenSafety (P08).
Contents

Abstract ...................................................................................................................................... ii
Acknowledgement ................................................................................................................... iii
I. Introduction ............................................................................................................................ 1
II. Related Work......................................................................................................................... 5
III. Background .......................................................................................................................... 7
    A. Network Simulator ............................................................................................................ 7
    B. Network Model .................................................................................................................. 7
IV. Analysis of the existing OMNeT++ simulator .................................................................... 9
    A. Profiling the OMNeT++ code ........................................................................................... 9
    B. Real-time simulator ........................................................................................................... 9
    C. INET Detailed & INET Abstract model .......................................................................... 11
    D. Accuracy of the INET Abstract Model ........................................................................... 16
V. Approach – Our MANET Model ........................................................................................ 18
VI. Experimental Evaluation ................................................................................................... 22
    A. Setup ................................................................................................................................ 22
    B. Results & Discussions ..................................................................................................... 22
    C. Limitation of our model ................................................................................................... 29
VII. Conclusion & Future work ............................................................................................... 30
Bibliography ............................................................................................................................ 32
I. Introduction

The use of Mobile ad hoc Networks (MANET) has been growing continuously for the past few years [1]. A mobile ad hoc network is an autonomous collection of mobile devices (laptops, smart phones, sensors, etc.) that communicate with each other over wireless links and cooperate in a distributed manner in order to provide the necessary network functionality in the absence of a fixed infrastructure. The mobile nature and the ability to configure it on the fly make MANETs useful for a number of applications such as emergency and rescue operations, local networks, etc. The development and deployment of wireless applications onto real networks can be really time consuming for large networks which typically consist of hundreds of nodes. This requires a large investment of time and money. To overcome this overhead designers make use of network simulators which let them test and validate each layer of a MANET node, from application layer to the physical layer. Designers typically use network simulators to simulate complex network topologies, plan protocols and balance traffic loads generated by applications. A lot of network simulators are available for both commercial and non-commercial use such as ns-3 [2], OPNET [3] and OMNeT++ [4]. These simulators have the capability of simulating wired networks, wireless network, and even vehicular networks including movement of components and communication channels. More advanced simulators [2][4] are even able to incorporate the usage of real devices during simulations, a technique called Hardware in loop (HIL) simulation.

Use of HIL simulator brings the application developers a step closer to real network scenarios. In HIL simulation, a real hardware device running the application is connected to a simulated network. This lets developers assess the behavior of their application and target network under different scenarios, which is essential to understand how the application will perform after deployment. Simulators such as ns-3 and OMNeT++ have support for HIL simulation. But these simulators tend to slow down and do not operate in real time constraint when networks with more than 20-30 nodes are being simulated. Both ns-3 and OMNeT++ are open source simulators and have ready to use MANET models available. However, to use these MANET models for HIL simulation we need to integrate several classes that act as a bridge between the real hardware and these MANET models. There are different ways to integrate real world applications into OMNeT++ namely, socket connection, source code integration and shared library integration. Use of socket connection is the fastest way to setup HIL simulation with OMNeT++ but it could raise CPU and timing issues [5].

Though it is possible to simulate the application and network in an existing (non-real time) simulator without using a HIL simulator, there is a drawback to this approach. The network simulators consider network behavior separately from the specific applications the network should accommodate. As a consequence, application designers seldom have immediate feedback from the network to change the behavior of the application or network. Also, the speed of execution of a real hardware is very fast compared to a simulator. HIL simulation offers developers to test their application instead of using a real network but there are some challenges in it as well. To enable HIL development, both the network simulator and the
application should run concurrently in real-time constraints. But this requirement is difficult

to meet as the processing speed of simulator depends on a number of factors such as the
detail level of number of simulated nodes, data rate and packet size, mobility of nodes,
routing protocol, etc. Modeling the network more accurately will make the simulator more
computational intensive. Also, slowing down the application at the real hardware creates
unnecessary complexity during the development of each application that uses our HIL
environment. Thus the simulation must match the real-time constraints of the applications
running on the physical hardware while preserving much of the accuracy. A simple real time
MANET model which provides same delay for all the messages in a simulation can be
realized with relatively less effort. However, this would result in a very inaccurate simulation
results. Therefore, we need our real time simulator to also be as accurate as possible. In
Chapter IV we discuss more about accuracy and how we compare different simulation
models.

This work is developed under the COMMIT [6] project SenSafety [7]. SenSafety is about
opportunistic sensing for safety in public spaces. Its goal is to collect, analyze and
disseminate data by means of a sensor network. SenSafety focuses on monitoring crowd rich
environments such as festivals, parades, and public events. In case of fights or fires unsafe
situations occur, which often hamper or block the normal means of communication.
Hampered communication occurs due to network overload or simply due to destruction of the
communication infrastructure. Typical applications within SenSafety rely on many devices
such as smartphones carried by individuals, sensor nodes installed in the infrastructure,
routers and unmanned aerial vehicles (UAVs). All these devices together create a MANET on
a temporary event site on which security related video of certain threatening situations can be
transported. With the collected information first responders, such as police or fire brigade,
can make a more informed decision on how to respond.

The benefits of a HIL capable real time MANET simulation for SenSafety are:

- It enables application developers to understand how their applications are likely to
  perform within the typical network conditions found in SenSafety, such as moving
  nodes, large number of nodes, frequently interrupting communication.

- It allows network architects and protocol designers to immediately understand the
  impact of their decisions, such as where to position gateways or which
  communication protocols to use, on real applications.

Figure 1 shows a typical ad-hoc network scenario applicable to SenSafety project. The
scenario consists of a number of mobile nodes in which a node from one end is sending data
packets to a fixed node located at another end of the field. Other nodes in the network
generate traffic and send it to random destinations. Throughout our work this scenario is used
as a base for a realistic use case scenario consisting of smartphones, laptops and video stream
traffic. This scenario forces the network to use routing for the traffic to reach the destination.
Additionally it avoids networks where the source and sink of the data are close to each other.
A typical application that we wanted to address is video streaming in a MANET. However,
by performing some experiments (discussed in Chapter IV) it was found that even a very abstract OMNeT++ MANET model is not capable to operate in real time for the high data rates required by video streaming. Thus, we try to simulate a scenario where audio streaming at a data rate of 8 kbps can be achieved within the real time constraint.

![Ad hoc network scenario for SenSafety](image)

**Figure 1: Ad hoc network scenario for SenSafety**

In our work

- We assess the standard deployment version of OMNeT++ and INET framework with respect to its capability for the real time HIL simulations.

- We develop a MANET model that operates in real time and can be used for HIL simulation with the OMNeT++ network simulator.

- We validate our model by performing a set of experiments and comparing it with the existing INET detailed model.

The existing INET [8] framework for OMNeT++ consists of two MANET models namely, the “detailed model” and the “abstract model”. In Chapter IV we discuss more on these models. Our MANET model is built on top of the existing abstract model. It is found through code profiling that in the detailed model, the physical layer largely contributes to the computational load in the simulation. Our model uses an abstract implementation of the physical layer (radio) and the wireless channel to reduce the computational load of the simulator. Our model calculates the end to end delay for packets based on the packet hop count and a set of derived polynomial equations. To validate our model we compare it to the
INET detailed model in terms of execution speed and accuracy. The experiments conducted on our model show that it is capable to operate in real time for scenarios and boundary conditions suitable for SenSafety with a relative error that falls within 2 to 8%. Our model can be used for real time simulations up to a data rate of 16 kbps. It is capable of simulating up to 100 nodes without losing its real time capabilities. Figure 2 shows a comparison of different MANET models in terms of execution speed and accuracy. We can see that the INET detailed model has high accuracy but is way outside the real time constraint. The INET abstract model though operates well within the real time constraint, its accuracy is not comparable to the INET detailed model. Thus, it is required to develop a MANET model that is as close as possible to INET detailed model in terms of accuracy and also operates in real time.

Figure 2: Comparison of different MANET models

The remainder of the report is organized as follows. In Chapter II, we provide a summary of the related work. The summary mentions the different techniques used to speed up network simulators. In Chapter III, we give an overview of the tools used in our model and reasons for their selection. Following that, in Chapter IV, we present a more detailed description of the problem statement and explain the scenario in which our model is applicable. Chapter V and VI describes our approach to the problem and the experimental results respectively. Finally, we conclude our report in Chapter VII.
II. Related Work

A literature survey in the field of network simulators reveals many ways to improve their performance. The work done by Jin et al. [9] proposes two speedup techniques that significantly improve the speed and scalability of distributed sensor network simulators by reducing the number of sensor node synchronizations during simulations. Even though, OMNeT++ simulator is capable of parallel and distributed processing, the INET framework running on top of OMNeT++ cannot be run in parallel. The reason behind this is that the MANET models within the INET framework use some important global components like IPv4Configurator (for assigning IP addresses) and ChannelControl (for sensing medium and radio control). These are central components that keep record of the status of all the nodes in the simulation. To parallelize these components we would have to make changes to them. This step requires a large investment in time because most these components have dependencies within the framework and it is possible that every component has to be re-implemented.

Walsh et al. [10] proposed the method of staged simulation to reduce the runtime of a discrete event simulator. The central idea of their work is to eliminate redundant or partially-redundant computations typical in simulations by caching and reusing the results of computations. In this work, the authors propose changes to the simulator rather than the models used within the simulator. This appears to be a promising technique to improve simulation speed. The authors test their work on the popular ns-2 simulator and could achieve 5 – 30x speedups for different network configurations. However, the data rate used by the authors in their experiments is 8 packets per second with each packet of size 512 bytes. The largest recommended data size (maximum transmission unit, MTU) in wireless communication is 100 bytes out of which 40 bytes are taken up by RTP and UDP headers. For 8 kbps audio streaming we will have to send approximately 17 packets per second which is double the packet rate used by authors in their experiments. Thus further experiments need to be done so that this method can be complementary to our approach and could further improve the simulation speeds for audio streaming.

Fluid model-based simulation [11] has been proposed as a solution to alleviate the computational overhead in large scale network simulation. In the fluid model based simulation approach, a group of closely-spaced packets are modeled as a fluid chunk at a specific time point. The behavior of these chunks is characterized by a set of mathematical models in the time domain. Though this method shows improvement in execution speed, it deviates from the packet level simulation approach. Also, to incorporate this method in our model we will have to redesign most of the components and possibly include new components for tracking these fluid chunks and their rate change.

As described earlier, the INET framework [8] consists of “detailed” and “abstract” MANET models. For our work, we wanted to make use of the existing abstract MANET model. Some of the works described above are different ways to improve simulator execution speed. Some of the methods can be used together with our work to give further improvement in the
OMNeT++ BASED MODEL FOR REAL-TIME MANET SIMULATIONS

simulator performance. The methods suitable for our work would require a considerable amount of changes to the MANET model and the simulator source code. As a result most of the techniques to improve the simulator speed could not be applied immediately to our work.
III. Background

A. Network Simulator

The network simulator is a tool that models the behavior of a network by calculating the interaction between the different network entities (nodes, data packets, etc.). To implement the HIL simulation approach we need to select a network simulator fulfilling certain criteria. The selection criteria depend on features such as accuracy, execution speed, scalability, etc. of the simulator and no single simulator is superior with respect to all the features. There are a number of network simulators available to choose from. We prefer to use a simulator that offers a wide variety of models out-of-the-box. Models are processes which emulate the hardware and run on top of the simulator.

Simulators like ns-3 [2] and OMNeT++ [4] are popular event based simulators that offer a wide variety of models to use. There is no dominant simulator when comparing OMNeT++ and ns-3, in terms of performance and scalability. Both, being open source, have an adequate selection of models available and provide easy additions to or modifications of the existing set. There are many articles comparing these different network simulators but they do not provide a clear choice on which suits best for our simulation scenario [13][14][15].

OMNeT++ is a good simulator for research purposes and has a well-designed simulation engine that supports hierarchical modeling [16]. In hierarchical modeling each component of a MANET node is modeled as a layer. These layers are connected by gates which transfer the data packets up and down these layers. As a result there is no tight coupling between layers and any layer can be replaced by a user defined implementation keeping the other layers intact. This aspect of hierarchical modeling is very important for our work where we want to modify the radio layer and keep the higher layers (from link layer to application layer) intact. Another important selection criterion is the availability of analysis tools. OMNeT++ has a rich graphical user interface with many possibilities such as step-by-step network debugging and results processing, while ns-3 relies on external tools such as WireShark [17]. Hence, OMNET++ Network simulator is preferred for our project.

B. Network Model

A network model is an abstraction of the functionality of a device or a group of interconnected devices. A detailed model for wireless networks emulates each layer [18] of a wireless node as close as possible to a real device. A network simulator often uses an event driven engine and a model describing what is processed after an event is triggered. Within the event it is possible to schedule one or more follow up events. These events can be scheduled with variable delay, thus the network simulator can incorporate processing time, traveling time, or delays independent of the time it takes to process the event. To avoid the tedious task of building all necessary models, there are several model sets available, which are called frameworks within OMNET++. For our project we need a basic mobile ad hoc network model that lets us configure it to use different routing protocols and mobility for various use
case scenarios. Also, it is preferred that the framework already has capabilities to communicate to external devices for HIL simulations.

There are three potential frameworks for OMNeT++ that fit our requirement namely, INET Framework [8], INETMANET framework [19], and MiXiM framework [20]. The INET Framework contains models for several wired and wireless networking protocols. This includes the internet stack, mobility, different MANET protocols, and several basic application models like UDP and TCP to generate traffic. INETMANET is a fork of INET and includes MANET specific models, experimental features and protocols. Most of the models in the INETMANET framework are added to the INET framework. The third, MiXiM, includes modules for the IEEE 802.11 standard. It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols, concentrating more on the lower layers of the OSI [18] model. MiXiM lacks implementation of various routing protocols. Hence, MiXiM is not suitable for our scenario. There is great similarity between INET and INETMANET framework. Also, INET 2.3.0 has an abstract implementation of wireless radio and channel which forms the basis of our model. Above points justify the use of INET framework for our scenario.
IV. Analysis of the existing OMNeT++ simulator

A. Profiling the OMNeT++ code

It was necessary to profile the OMNeT++ code together with the INET framework’s MANET model to check which parts of the code take up most of the execution time. If the overhead of OMNeT++ simulator itself would have been considerable then it would have been necessary to design a new simulator (our experimental results will however show that this is not necessary, allowing a re-use of many components already developed as part of OMNeT++). Valgrind (Callgrind), a profiling tool for software in Linux was used to profile the code [21]. This tool gives the time spent in each function as a percentage of the overall runtime. The results show that a large amount of time is spent in simulating the radio (54%) and the MAC layer (38%). But the percentage of time used is distributed across many functions. Table 1 shows the percentage of time spent in different parts of the simulator code. As the data from profiling is huge, we just list the important functions which take up most of the simulation time.

Table 1: Results of OMNeT++ and INET Framework code profiling

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Percentage of Time</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cmdenv::simulate()</td>
<td>1.92</td>
<td>This is the simulator overhead when run in command line mode.</td>
</tr>
<tr>
<td>Radio::handleMessage()</td>
<td>54.75</td>
<td>Implementation of the physical layer (radio)</td>
</tr>
<tr>
<td>WirelessMacBase::handleRequest()</td>
<td>38.34</td>
<td>Implementation of the MAC layer</td>
</tr>
<tr>
<td>MobilityBase::handleRequest()</td>
<td>1.97</td>
<td>Handles mobility of nodes</td>
</tr>
</tbody>
</table>

By profiling the OMNeT++ code it is clear that simulator infrastructure does not impose much overhead in computation and modifications are required in the physical layer of the MANET model of INET framework. This layer basically addresses the common functionality of the radio and abstracts out specific parts into two interfaces namely, ReceptionModel and RadioModel. The reception model is responsible for modelling path loss, interference and antenna gain. The radio model is responsible for calculating frame duration, modeling modulation scheme and possible forward error correction.

B. Real-time simulator

This section describes some of the important terms used in the remaining part of the thesis. To understand these terms it necessary to learn the behavior of messages in a real world and in a simulated environment. Consider a scenario where two smartphones named node A and node B communicate with each other. Suppose that in the real world a message packet takes 10µs to travel from node A to node B. This is the time from the instant the packet was created by node A till it is received at node B. We call this time real world end to end delay. When
the same scenario is simulated, the network simulator takes some time to execute the simulation code and give us the result of 10µs. Figure 3 shows two possibilities in a simulated environment. In case of a real time simulator the time it takes to process and give us the real world end to end delay, in this case 10µs is less than or equal to 10µs. However, a non-real time simulator takes more than 10µs to give us the same real world end to end delay.

**Simulated end to end delay**: It is the real world end to end delay of a single packet given by the simulator. This represents the actual amount of time a packet will take to travel from sender to receiver in the real world.

**CPU time**: This represents the processing time taken by the simulator to calculate the simulated end to end delay for a packet. To measure this we set the real world time stamp to a packet when it is created. Once the packet is processed and its simulated end to end delay is calculated, we get the attached time stamp from the packet and subtract it from the current time. We can say that CPU time is the elapsed “wall clock” time from the instance a packet is created at sender till it is received at the application layer of the receiver in a simulator.

**Real-time simulator**: For real time operation of the simulator, the CPU time should be less than the simulated end to end delay for every packet sent in the simulated network. We call such a simulator a real time simulator. For our work we relax this criteria a bit. There are some specific network operations which tend to be slow and occur sporadically. For example, during the route discovery process there could be a situation when the CPU time for certain packets increase and thus they deviate from our real time definition. We ignore such cases as they form just a small fraction of the total packets sent. Thus a real time simulator for us is one that operates as per our real time definition for most of the packets sent in the network.
C. INET Detailed & INET Abstract model

INET 2.3.0 comes with a highly abstract implementation of the MAC and radio layer. These modules can be used to test the routing protocols in MANETs. Though this speeds up the simulation considerably, accuracy is lost. Several experiments were performed to test the OMNeT++ network simulator (along with the INET models) and evaluate if it can be used for real-time simulation in the context of video streaming in mobile ad hoc networks. In these experiments we compare the real time performance of the INET abstract model and the INET detailed model. To measure the real time performance we use the metric CPU time/simulated end to end delay. As long as this factor is less than 1 we say that the simulator along with the MANET model can work in real time. As the simulations of the INET detailed and INET abstract model run separately, it is not possible to compare CPU time/simulated end to end delay for every packet one to one. Thus we take the average of this value across the whole simulation time and then compare.

In these experiments we set up our network configuration suitable for the SenSafety scenario (Figure 1). We have the sender and receiver nodes on the corner of the field such that they are diagonally opposite to each other. The field consists of a number of mobile nodes that receive and forward the packets to next destination node. Table 2 lists the configuration used for these experiments to setup a network similar to SenSafety scenario. Experiments were conducted by varying different parameters in the network (and keeping other parameters constant as mentioned in Table 2) such as

- Number of nodes
- Mobility of nodes
- Data rate and packet size
- Type of routing protocol
- Size of the field
- Transmitter power

<table>
<thead>
<tr>
<th>Network Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field size</td>
<td>500m X 500m</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>30</td>
</tr>
<tr>
<td>Mobility of nodes</td>
<td>Random mobility using INET frameworks “Mass Mobility” class</td>
</tr>
<tr>
<td>Routing Protocol</td>
<td>AODV*</td>
</tr>
<tr>
<td>Data rate and packet size</td>
<td>8 kbps (17 RTP packets of 100 bytes)</td>
</tr>
</tbody>
</table>

*Even though AODV has more latency compared to OLSR, it is more suitable in a resource critical environment. The OLSR protocol is more efficient in networks with high density and highly sporadic data traffic [23].

All these experiments were conducted on a machine with an Intel core i7 2.4Ghz, 8GB RAM and Ubuntu 12.04 (64 Bit). OMNeT++ version 4.4 and the INET 2.3.0 framework were used
for network simulation. To speed up OMNeT++ scalar and vector logging was disabled and the simulator was run in command line express mode (Cmdenv). The command line user interface is a small, portable and fast user interface that compiles and runs on all platforms. Cmdenv does not make use of the graphical user interface and is designed primarily for batch execution. All the projects were built in release mode using the GCC compiler. Each experiment was run for a simulation time of 800s and repeated 30 times with 30 different seeds. The average value and standard deviation for all the runs are calculated and plotted in the graphs discussed below.

Figure 4: Real time performance comparison of MANET models with increasing number of nodes and 8 kbps data rate

Figure 4 shows the real time performance of the INET abstract and INET detailed models with respect to increasing the number of mobile nodes. The numbers next to the data points indicate the ratio of CPU time and simulated end to end delay. We can see that as the number of nodes increase the computational load also increases. Hence, there is a steady increase in the ratio of CPU time and simulated end to end delay. Also, it can be seen that the INET abstract model performs much better compared to the INET detailed model in terms of simulation speed. INET abstract model is able to operate in real time for up to 60 nodes whereas the INET detailed model becomes non-real time for just 20 nodes. This shows that number of nodes is an important bottleneck for real time operation of a network simulator.

Figure 5 shows the real time performance of the INET abstract and INET detailed models with increasing data rate. An increase in data packets means more events to process which results in an increase in execution time. Also, more traffic in the network results in more packet collisions and re-transmissions which increase the end to end delay. But this increase
in end to end delay is less compared to increase in *CPU time* as seen from the graph. We can see that for data rates less than 16 kbps, the INET abstract model operates within the real time constraint whereas the INET detailed model is already non-real time for an 8 kbps data rate.

![Real time performance comparison of MANET models with 30 nodes and increasing data rate](image)

Figure 5: Real time performance comparison of MANET models with 30 nodes and increasing data rate

For streaming data the standard protocol to use is RTP (Real-time Transport Protocol) with UDP. So, for a QCIF quality video transmission (16FPS, 176x144, 1 byte per pixel) we need to send 405504 bytes per second. With 100x compression it reduces to 4055 bytes/s. The largest recommended data size in wireless communication is 100 bytes of which the RTP and UDP headers take 40 bytes. Therefore the required throughput is 4055/60 = 67.5 packets/s. The INET abstract model operates in real time constraint only up to 16 kbps which is close to 34 packets per second. Hence, even with the INET abstract model the required throughput for video streaming is not met. However, with a data rate of 8-16 kbps we can have a real time simulation for audio streaming. With same RTP with UDP protocol for audio streaming, we now have to send just 16 packets per second for 8 kbps data rate.

From Figure 6 we can see that the OLSR routing protocol gives a high value to the ratio of *CPU time* and *simulated end to end delay*. OLSR protocol requires more processing as it is a proactive routing mechanism (Routes are established before they are actually needed). Nodes periodically broadcast topological information updates to all other nodes in the network. This increases the CPU time as more messages are to be processed. Also, as routes are already established in OLSR protocol, the data packets get delivered early resulting in reduced end to end delay. We can see from the graph that INET abstract model operates well within real time constraint for both AODV and DSR protocols.
OMNeT++ BASED MODEL FOR REAL-TIME MANET SIMULATIONS

Figure 6: Real time performance comparison of MANET models with various routing protocols (8 kbps data rate and 30 nodes)

In Figure 7 we compare the real time performance of both the models with respect to the field size. As the field size increases the end to end delay of packets also increases. As a result of this the ratio of CPU time and simulated end to end delay starts reducing. Hence, for large field sizes the INET models operate in real time. However, very large field sizes do not suit our SenSafety scenario. A field size of 500m X 500m is a reasonable choice.

Figure 7: Real time performance comparison of MANET models with varying field size (8 kbps data rate and 30 nodes)
Figure 8 shows the real time performance of both the models with respect to the transmission range. In case of INET detailed model, we have to specify transmitter power based on which the transmission distance is calculated using the Friis transmission equation given by

\[ r = \frac{10^{(p_t+g_t+g_r-p_r)/20}}{41.88 X f} \]

where
- \( f \) is frequency,
- \( p_t \) is transmitter power,
- \( g_t \) is transmitter antenna gain,
- \( g_r \) is receiver antenna gain and
- \( p_r \) is receiver sensitivity

Figure 8: Real time performance comparison of MANET models with varying transmitter range (8 kbps data rate and 30 nodes)

In the INET detailed model, with increasing transmitter power the number of hops for a packet is reduced. Because of the fewer number of hops both the CPU time and simulated end to end delay decrease. However, this effect is more pronounced on the simulated end to end delay which in turn increases the ratio of CPU time and simulated end to end delay.

From the experimental results described above it is clear that in terms of real time performance, the INET abstract model is better than the INET detailed model. In certain cases the simulator along with the INET abstract model operates within the real time
constraint. Using the INET abstract model we are able to achieve our real time requirement for 16 kbps data rate with 60 nodes. Data rate of 16 kbps is not sufficient for video streaming applications but we can still make use of INET abstract model for audio streaming applications. Though INET abstract model is able to run within real time constraints, it does so by losing the accuracy. In the next section we discuss our notion of accuracy and compare the accuracy of INET abstract model with the INET detailed one.

D. Accuracy of the INET Abstract Model

To test the accuracy of the INET abstract model we compare it with the INET detailed model. Similar to the experiments in the previous section, we run another set of experiments using both the models by varying parameters such as number of nodes, data rate, routing protocol, etc. For each simulation setup we calculate the average end to end delay of packets. We compare the average end to end delay of the packets for both the models to check for accuracy as it is not possible to map INET detailed and INET abstract packets one to one. A comparison on the basis of individual packets is not possible as the experiments of INET abstract and INET detailed model execute as separate simulation runs. In each set of the experiments it is found that the average end to end delay given by INET abstract model is quite less than the one given by INET detailed model. In Figure 9 we show one such scenario where a comparison of end to end delays of all the packets for INET detailed and INET abstract model is done. This simulation was run for 800s at 8 kbps data rate keeping other parameters the same as in Table 2. As the graph is too big we just show the first 250s of the simulation.

![Figure 9: Comparison of accuracy of the INET detailed and INET abstract model](image-url)
Table 3 shows the average delay and the number of sent and received packets for this particular experiment. The average end to end delay is much smaller for INET abstract model compared to the INET detailed model. This is obvious as there is no detailed implementation of physical layer in the INET abstract model. The physical layer in the INET abstract model does not calculate the interference range and can also send and receive packets simultaneously. These factors reduce the end to end delay for packets considerably as shown.

<table>
<thead>
<tr>
<th>8 kbps data rate</th>
<th>Sent</th>
<th>Received</th>
<th>Average end to end delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>INET abstract</td>
<td>14220</td>
<td>14220</td>
<td>0.129 ms</td>
</tr>
<tr>
<td>INET detailed</td>
<td>14220</td>
<td>14219</td>
<td>1.115 ms</td>
</tr>
</tbody>
</table>

From the above experiment it can be seen that the end to end delay estimate of INET abstract model is inaccurate as compared to the INET detailed model. Thus, we need to model this difference in delay without affecting the real time capabilities of the INET abstract model. In the next chapter we discuss how we approach this problem.
V. Approach – Our MANET Model

In the previous chapter it was experimentally shown that the INET abstract model is capable of working within the real-time requirements for certain conditions. However, the model is quite inaccurate compared to the INET detailed model. Our aim is to build a new model that lies in between INET detailed and INET abstract such that it is as close as possible to the detailed model in terms of accuracy and still operates under the real-time requirements. Figure 2 shows this graphically. Figure 10 shows a quick comparison between the INET detailed and the INET abstract model. The INET abstract model consists of an ideal radio and ideal channel. An ideal radio layer is capable of sending and receiving packets simultaneously which does not happen in reality. All the layers above the radio (Physical layer) are same for both the INET detailed and INET abstract model. The key idea behind our approach is to use a set of polynomial functions to model the physical layer of a MANET node. These polynomial functions derived from a number of parameters such as bitrate, number of nodes, hop count compute the difference in the end to end delay given by the INET abstract and INET detailed model.

![Figure 10: Comparison of the INET detailed and INET abstract model](image)

From the experiments in Chapter IV we can see that the simulated end to end delay varies based on a number of factors such as data rate, transmission range, field size and number of nodes sending data. We performed a set of experiments to identify the most important parameters on which the difference in average end to end delay of packets in INET abstract and INET detailed model depends.
The list of parameters is

- Bitrate
- Transmission range
- Number of nodes sending data
- Placement of nodes
- Field size

In each set of experiments we vary one of these parameters to check how the difference in average end to end delay of packets in the INET abstract and INET detailed model relate to these parameters. Transmission range determines the hop count for the packets. A smaller transmission range leads to a larger number of hops required for the packets to reach their destination. The number of nodes, bitrate and field size are modeled as a parameter in terms of $bits/s/m^2$. We can say that the difference in the average end to end delay between the INET detailed and INET abstract model can be modeled as

$$\Delta E2E = f(\text{hop count, bits/s/m}^2)$$

Figure 11 shows a block diagram of our proposed model. In the proposed model the user has the option to specify number of additional nodes and the rate at which they send the data. These nodes and their generated traffic are modeled into the parameter $bits/s/m^2$. So, in our model these nodes are actually not simulated but their effect on end to end delay of packets is taken into account. This relieves the simulator from the computational overload as more number of nodes to simulate slows down the simulator (see Figure 4).

![Block diagram of proposed MANET model](image)

Figure 11: Block diagram of proposed MANET model.

Assuming the traffic is spread uniformly in the simulated network, we plot the variation of difference in average end to end delay for the INET abstract and the INET detail model with respect to increasing traffic (see Figure 12). For each hop count we get a curve which we approximate using a polynomial function such that the coefficient of determination denoted $R^2$ is close to 1. A $4^{th}$ degree polynomial function traces these curves closely.
Though we can use higher degree polynomial functions to trace these curves more accurately, it was found from experiments that a 4th order function gives us accurate enough results. For higher traffic in the network a 3rd degree polynomial function gives inaccurate results whereas a 4th and 5th degree functions give us comparable results. Figure 13 shows a comparison of average end to end delay given by INET detailed model and our model with 3rd, 4th and 5th degree polynomial functions for one such case. It can be seen that the result given by using 3rd degree polynomial functions deviates too much when compared to INET Detailed model. A 4th and 5th degree polynomial functions give us quite similar results.

Table 4 lists the polynomial functions for curves shown in Figure 12. Once the packets reach the application layer of the receiver we check its packet information to get the hop count. Based on the hop count of the packet we select an equation from Table 4. Using the equation and plugging in the value for \( \text{bits/s/m}^2 \) we get the difference in average end to end delay of the INET abstract and the INET detail model. This delta component is then added to the packet delay given by the simulator.
Figure 13: Comparison of INET detailed and our model with 3rd, 4th and 5th degree polynomial functions

Table 4: Equations for our MANET model

<table>
<thead>
<tr>
<th>Hop</th>
<th>Equation ((y = \Delta E2E, x = \theta to 20 \text{ bits/s/m}^2))</th>
<th>Coefficient of determination ((R^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(y = 8E\cdot13x^4 - 1E\cdot10x^3 + 7E\cdot09x^2 - 9E\cdot08x + 0.0005)</td>
<td>0.2901</td>
</tr>
<tr>
<td>2</td>
<td>(y = 1E\cdot11x^4 - 2E\cdot09x^3 + 1E\cdot07x^2 - 3E\cdot06x + 0.0007)</td>
<td>0.9449</td>
</tr>
<tr>
<td>3</td>
<td>(y = 2E\cdot10x^4 - 3E\cdot08x^3 + 2E\cdot06x^2 - 2E\cdot05x + 0.0012)</td>
<td>0.9779</td>
</tr>
<tr>
<td>4</td>
<td>(y = 2E\cdot10x^4 - 3E\cdot08x^3 + 1E\cdot06x^2 - 3E\cdot05x + 0.0017)</td>
<td>0.9910</td>
</tr>
<tr>
<td>5</td>
<td>(y = 2E\cdot10x^4 - 4E\cdot08x^3 + 2E\cdot06x^2 - 3E\cdot05x + 0.0022)</td>
<td>0.9840</td>
</tr>
<tr>
<td>6</td>
<td>(y = 2E\cdot10x^4 - 3E\cdot08x^3 + 1E\cdot06x^2 - 1E\cdot05x + 0.0026)</td>
<td>0.9102</td>
</tr>
</tbody>
</table>
VI. Experimental Evaluation

In the previous chapter we presented our MANET model. To validate our model we compare it with the INET detailed model in terms of accuracy. It is not possible to compare the messages of two models one to one. So, to measure the accuracy we compare the average end to end delay of packets of our MANET model with the INET detail model. We also test our model for real time performance.

A. Setup

The setup used to test the accuracy of our model is same as for the previous experiments. All these experiments were conducted on a machine with an Intel core i7 2.4Ghz, 8GB RAM and Ubuntu 12.04 (64 Bit). OMNeT++ version 4.4 and the INET 2.3.0 framework were used for network simulation. All projects were built in release mode using the GCC compiler. Each experiment was run for a simulation time of 100 seconds.

B. Results & Discussions

Table 5 shows the comparison of average end to end delay of our MANET model and INET detailed model with variation in transmission range. The values in brackets represent the standard deviation. For each transmission range we perform multiple experiments with different seed values and then take an average of all the end to end delay given by each experiment. It can be seen from the table that for higher transmission range the loss in accuracy of our model is less than 1%. However, for small transmission range the loss in accuracy is more than 20%. For smaller transmission range the number of hops for a packet increases and as there is no interference factor in INET abstract model the resulting number of hops could be less compared to the INET detailed model. This results in mismatch in hop count of our MANET model with the INET detailed model and hence deviations in calculated end to end delay value.

Table 5: Average end to end delay comparison for different transmission ranges (40 nodes, 8 kbps)

<table>
<thead>
<tr>
<th>Range (m)</th>
<th>Average INET detailed (in seconds) (SD)</th>
<th>Average Our model (in seconds) (SD)</th>
<th>% change in average</th>
</tr>
</thead>
<tbody>
<tr>
<td>550</td>
<td>0.001029780 (0.000619098)</td>
<td>0.001031699 (0.000629412)</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>340</td>
<td>0.001116788 (0.002307949)</td>
<td>0.001207197 (0.001599930)</td>
<td>8%</td>
</tr>
<tr>
<td>230</td>
<td>0.003523162 (0.026935402)</td>
<td>0.003135614 (0.010961301)</td>
<td>11%</td>
</tr>
<tr>
<td>175</td>
<td>0.010669281 (0.095508926)</td>
<td>0.008114163 (0.050339391)</td>
<td>23%</td>
</tr>
</tbody>
</table>

Figure 14, 15 and 16 show the plot of all the packets sent during the simulation time for both the INET detailed model and our MANET model with variation in transmission range for selected seed values. It can be seen that a lot of packets overlap in both the models.
Table 6 shows the comparison of the average end to end delay of our MANET model and the INET detailed model with variations in the positions of sending and receiving nodes. In previous experiments the sending and receiving nodes were positioned on the diagonally opposite ends of the field. The seed value is used to generate randomness in the positioning of the nodes. It can be seen from the table that for different seed values the loss in accuracy is within 2%. But there could be scenarios where the positioning of nodes is such that the loss in accuracy is high. We discuss more on this in the next section on limitation of our model.
OMNeT++ BASED MODEL FOR REAL-TIME MANET SIMULATIONS

Figure 16: Comparison of INET detailed and our MANET model for transmission range of 175m (40 nodes, 8 kbps)

Table 6: Average end to end delay comparison for different position of nodes (40 nodes, 8 kbps)

<table>
<thead>
<tr>
<th>Seed</th>
<th>Average INET Detailed (in seconds) (SD)</th>
<th>Average Our Model (in seconds) (SD)</th>
<th>% change in average</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.001029780 (0.000619098)</td>
<td>0.001031699 (0.000624912)</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>23</td>
<td>0.001023263 (0.000179483)</td>
<td>0.001014714 (0.000085731)</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>42</td>
<td>0.001326069 (0.012268557)</td>
<td>0.001298180 (0.010935646)</td>
<td>2.1%</td>
</tr>
<tr>
<td>12</td>
<td>0.001399777 (0.013330908)</td>
<td>0.001230075 (0.000519464)</td>
<td>12%</td>
</tr>
</tbody>
</table>

Figure 17 and 18 show plot of all the packets sent during the simulation time for both the INET detailed and our MANET model with variation in the positions of sending and receiving nodes.

Figure 17: Comparison of INET detailed and our MANET model for a seed value of 7 (40 nodes, 8 kbps)
In Figure 18 we show the real time capabilities of our model. It can be seen that for around 50 nodes our model is capable of running within real time constraint which is very close to the performance of INET abstract model in terms of speed of execution. It can also be seen that INET detailed model crosses the real time line for 20 nodes itself as discussed earlier also.

Figure 19: Comparison of INET detailed, INET abstract and our MANET model for real time performance (8 kbps data rate)
Table 7 shows the comparison of the average end to end delay of our MANET model and the INET detailed model with variations in the number of nodes. As seen in Figure 19 our model works in real time up to 40 nodes. Nodes greater than 40, we model it using the \( \text{bits/s/m}^2 \) parameter. In the configuration file user can specify these nodes separately by using the parameter name \textit{unsimulatednode}. These nodes are not actually simulated in the simulation environment, however the traffic generated by them is taken into account.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Average INET detailed (in seconds)</th>
<th>Average Our model (in seconds)</th>
<th>% change in average</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.001006810</td>
<td>0.001007092</td>
<td>&lt; 1%</td>
</tr>
<tr>
<td>40</td>
<td>0.001025376</td>
<td>0.001005871</td>
<td>2%</td>
</tr>
<tr>
<td>60</td>
<td>0.001076773</td>
<td>0.001006940</td>
<td>6.4%</td>
</tr>
<tr>
<td>80</td>
<td>0.001107941</td>
<td>0.001006098</td>
<td>9%</td>
</tr>
<tr>
<td>100</td>
<td>0.001809568</td>
<td>0.001076868</td>
<td>40%</td>
</tr>
</tbody>
</table>

It can be seen that for higher number of nodes (more than 40) also our model is able give results with not much drop in accuracy. The difference in average value is more when compared to INET detailed model but the higher average value in INET detailed model is due to certain packets which have very high latency. Figure 20, 21 and 22 show a plot of all the packets sent for the whole simulation time for selected seed values. The plots show a lot of overlap between packets of INET detailed and our model. Hence, comparing the average end to end delay may not give a clear indication of the accuracy of our model.

Figure 20: Comparison of INET detailed and our MANET model for 60 nodes (8 kbps data rate)
Figure 21: Comparison of INET detailed and our MANET model for 80 nodes (8 kbps data rate)

Figure 22: Comparison of INET detailed and our MANET model for 100 nodes (8 kbps data rate)

Figure 23 show the cumulative distribution function (CDF) graphs for different number of nodes in the network. It can be seen that in each of the cases end to end delay of 80-90% of the packets sent in simulation overlap for INET detailed and our model.

We could not test the accuracy of our model for mobile nodes due to a bug in the INET abstract model of current INET version [22]. The IdealWirelessNIC (radio) of INET abstract model currently does not implement the LINK BREAK notification. This means that once a packet is lost (because the hosts get out of communication distance) the upper layers are not notified about the link break which means that older routes are not dropped. So all the data packets go the wrong way and no new routes are formed. Figure 24 shows one such case with mobile nodes where we can see that a lot of packets are getting dropped in INET abstract model compared to the INET Detailed model.
Figure 23: Comparison of INET detailed and our MANET model for different number of nodes. (a) 20 nodes, (b) 40 nodes, (c) 60 nodes, (d) 100 nodes

Figure 24: Comparison of INET detailed and INET abstract model with mobile nodes
C. Limitation of our model

So far we have compared the accuracy of our model with variation in different parameters. In the experiments where we change the network configuration by using different values for seed we see that there is not much loss in accuracy. But there are cases where our model could possibly fail. Table 8 shows the comparison of the average end to end delay of our MANET model and the INET detailed model depicting one such case where our model loses too much accuracy. In this example, we use the same configuration as for previous experiments, only difference being we position majority of the nodes in one corner of the field such that the traffic generated by them is concentrated to a particular area in the field as shown in Figure 25. We see that the average end to end delay given by our model is more compared to the INET detailed model. This is because in our model we assume the traffic to be distributed uniformly over the entire field. This is not the case with the INET detailed model. Hence, in INET detailed model the packets reach the destination through a less traffic path and have less end to end delay compared to our model. We discuss further on this limitation in Chapter VII.

Table 8: Average end to end delay comparison for a the network setup shown in Figure 25

<table>
<thead>
<tr>
<th>Average INET Detailed (in seconds)</th>
<th>Average Our Model (in seconds)</th>
<th>% change in average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001009323</td>
<td>0.001701645</td>
<td>&lt; 68.5%</td>
</tr>
</tbody>
</table>

Figure 25: Network setup to test limitation of our model
VII. Conclusion & Future work

In this work we have developed OMNeT++ Network simulator based MANET model that can be used for real time simulations. Our model uses an abstract implementation of the physical layer to reduce the computational load of the simulator. In our model we calculate the end to end delay for packets based on the hop count and a set of derived polynomial equations. In the previous chapter we described the implementation of our model and performed a set of experiments to compare its accuracy with the INET detailed model. We checked the accuracy of our model with the variations in transmission range of nodes, number of nodes and the network configuration. We also compared the real time capabilities of our model with the INET detailed and INET abstract models.

Our focus was to build a real time HIL simulator for the SenSafety scenario described in the introduction chapter. Experimental results show that the OMNeT++ simulator along with our MANET model is able to work in real time for up-to 40 nodes instead of just 10 nodes for the detailed model. Even for higher number of nodes the simulator is able to deliver results without considerable loss in accuracy when compared to the INET abstract model. Our model is capable of working in real time for data rates up to 16 kbps which is suitable for audio streaming. The user has the option to specify the number of nodes, data rate, transmission range and various MAC and routing settings. This work showed that it is possible to use polynomial functions to model the physical (radio) layer of a MANET node and improve the execution speed of the simulator. However, there is still a lot work that can be done to improve the simulator performance, both in terms of execution speed and accuracy. The staged simulation technique by Walsh et al [10] discussed in related work chapter looks very promising and can be used as a complementary approach to our work.

Our model, based on the INET abstract model, currently does not calculate the interference due to other nodes before sending a packet. This results in a data packet getting delivered to node which in reality should consider it as noise. Due to this there is sometimes mismatch in hop count for a packet when compared to the INET detailed model. In our model we can add a parameter where a user can specify the interference range so that packets are treated as noise when they are within the interference range. This method is different as compared to INET detailed model where the interference due to each node is calculated during runtime and hence it will not add much to the simulator execution time. Adding the option for interference range could possibly reduce the hop count mismatch between packets in our model and the INET detailed model and therefore will give more accurate results.

Our model currently assumes a uniform distribution of traffic across the field. This approach gives inaccurate results for certain network configurations. We showed one such example in the previous chapter (see Figure 25). To improve our model further, instead of considering a uniform distribution of the traffic in network we can divide the field in to smaller grids and compute the $\text{bits/s/m}^2$ in each grid. We then use the same approach of polynomial equations for each grid to compute the end to end delay. Also, we can further improve our model by
using different sets of polynomial functions for each field size. This would take the effect of field size more explicitly into account in our model.

Our aim was to build a real-time hardware in the loop simulator for the SenSafety scenario and test it with audio streaming application. We were able to develop a real time simulator for audio streaming but due to time constraints we could not add the hardware in loop functionality to our model. Therefore, one of the important works for future would be to add the hardware in loop functionality to our MANET model and test it for audio streaming applications.
Bibliography


