A Review on Wireless Sensor Network Simulator

Kartik N. Shah∗, Himanshu Ghetia, Shantanu Santoki, Jaishankar N.
School of Computing Science,
VIT University, Vellore, India

Abstract—Wireless Sensor Network (WSN) is one of the most favorable areas for research work. WSNs are composed of large number of sensor nodes. So, it is very complex to model WSN for analysis purpose. To implement and evaluation of algorithms and protocols in WSN, simulators and emulators are used. There is always a concern that simulation result may not match with the original behavior of the system. So, we must be aware about the strengths and weakness of the simulators. There are lot many simulators and emulators available for WSNs. These simulators can be changed based on requirements. In this paper, we will discuss some of the famous simulators and emulators. Also we will compare those simulators based upon their merits and demerits. In this paper, we will consider simulators like NS2, OMNET++, TOSSIM and EmStar.

Keywords—Wireless Sensor Network, Simulation Tools, Review, Survey, WSN.

I. INTRODUCTION

Network simulation is an evolutionary area of computer networks. They are widely used for development of protocols and network architectures. An arbitrary model that specifies behavior of the nodes and communication channels are called as Network Simulators. As an example, if we want to investigate the characteristics of a new routing protocol then it is usually implemented in network simulator. Most of the network simulation toolkits are based on discrete event-based simulation (DES) paradigm [1]. To analyze the performance of wired and wireless networks, we can have methods called analytical methods, physical measurements and computer simulation. On the other hand, for wireless sensor networks cannot be analyzed using above methods easily as it can have energy problems, limited battery life etc. So, discrete-event simulation can be used for WSNs. It can simulate many jobs running on all the nodes [2]. Trace-driven simulation is commonly used in real system and the results are more accurate [2].

At the beginning stage of a protocol, simulators are used to develop and test. It can finish execution in very short time. It can simulate thousands of nodes in very less cost and less amount of time [3]. In the case of emulator, it uses firmware as well as hardware for simulation [3]. Emulator uses real nodes, so it can provide better performance compared to simulators. Many architectures can be combined and made a ‘tiered architecture’ which can have different capabilities, costs and battery backup [11][12]. Wireless sensor nodes are very small in size and can be used in many fields as they are cheap in cost. It is having less complex functionality. So, if there are thousands of sensor nodes in the field then we cannot observe the energy consumption, network topology or new algorithm. So, we need to go for simulation environment. Fig. 1 shows the simple wireless sensor node.

Figure 2 shows the Model for wireless sensor network. We can see the functionality that Agent will pass the information to nodes via environment and nodes will pass the information to sink nodes using radio channel.
In the following subsection we will take the overview of different simulation tools.

II. SIMULATION TOOLS

A. NS2 (Network Simulator)

Network Simulator (NS) is also a discrete event simulator which was developed at UC Berkeley. It was developed for simulation of IP networks at packet level [6]. It is generally used to investigate the protocols such as UDP, TCP, ECN, RED, CBQ etc [7]. Following figure shows the simplified view of network simulator.

NS contains network simulation object libraries in Object-oriented Tcl (OTcl) script interpreter. NS2 architecture is having following five parts.
1. Event Scheduler
2. Network Components
3. Tclcl
4. OTcl library
5. Tcl 8.0 script language.
Figure 4 shows the architecture of NS2.
Visualization tool for NS is Network AniMator (NAM). Following Figure 5 shows the screenshot of NAM.

**B. TOSSIM (For TinyOS Networks)**

Different simulators have different focuses like NS simulators are simulation for packet level. TOSSIM was developed for TinyOS sensor networks. It is discrete event simulator developed at UC Berkeley. It deals with the interaction and behaviour of the networks rather than the packet level. A small sensing and communication device called as Motes. They are less expensive and limited energy devices. TinyOS is an OS used in sensors and the sensors that use this OS are called as Motes. Following Figure 6 shows the TOSSIM architecture.

TOSSIM architecture is made up of following parts [5]
1. Frames (TinyOS component graphs)
2. Events (Execution model)
3. Models (Radio and ADC)
4. Hardware abstraction components
5. Communication Services

It comes with Java visualization and actuation environment called TinyViz. [5]. The following figure 7 shows the TinyViz for TOSSIM.

![TinyViz connected to TOSSIM](image)

C. OMNET++

OMNET++ is a C++ based discrete event simulator for simulation of communication networks, distributed and parallel networks. It is widely used in simulation for complex IT systems and hardware architecture as well. It is free for academics (non-commercial) use. The modules of OMNET++ modules can communicate using message passing model. Active modules are also called as simple modules. These simple modules can be combined to a single module and called as composite module [9]. Figure 8 shows the logical architecture of OMNET++.

![OMNET++ logical architecture](image)

The components of OMNET++ are as follows: [10]
1. Simulation kernel library
2. NED topology description lang
3. IDE (based upon Eclipse)
4. Tkenv
5. Cmdeenv
6. Utilities
7. Documentation
Following figure shows the screenshot of OMNET++.

**Fig 9: Screenshot of Tkenv of OMNET++ [9]**

**D. EmStar**

EmStar is an emulator. It was designed for WSN built in C [13-14]. It was developed by University of California, Los Angeles. It is trace driven emulator for real time [14]. As it uses C language, we can do modular programming in it. So, we can have more flexibility of making different modules for different functionalities. We can also reuse the modules whenever needed. EmStar is GUI based emulator. So, it is easy to use. But the drawback is limited scalability. We can have limited number of nodes in this emulator. The layered framework of EmStar can be shown in figure 10.

**Fig10 Layered Framework for EmStar[15]**

Emstar environment is consists of libraries, tools, services and microkernel extension. The most widely used tools are:

- EmSim : Simulator for microserver
- EmCee: Interface for real low-power radios
- EmTOS : Extension of Emstar for TinyOS applications. The visualization of EmStar (EmView) can be shown as in following figure 11.
III. CONCLUSION

We can conclude from the above discussion that every simulator is made with a special purpose. Each simulator is having its merits as well as its demerits. So, based upon requirements we can select a proper simulator. Network Simulators (NS) are made for packet level analysis of data while TOSSIM was made for the study of behavior and interaction of networks. We can summarize as follows.

<table>
<thead>
<tr>
<th>Simulator/Emulator</th>
<th>NS2</th>
<th>TOSSIM</th>
<th>OMNET++</th>
<th>EmStar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete-Event (DE)/ Trace driven(TD)</td>
<td>DE</td>
<td>DE</td>
<td>TD</td>
<td>DE</td>
</tr>
<tr>
<td>Open Source</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GUI</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>General Purpose</td>
<td>Packet Level</td>
<td>TinyOS sensor</td>
<td>Communication networks</td>
<td>Real-time</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENT

The authors would like to thank the School of Computer Science and Engineering, VIT University, for giving them the opportunity to carry out this project and also for providing them with the requisite resources and infrastructure for carrying out the survey.

REFERENCES


