Design and Simulation of Wireless Sensor Network in NS2

Genita Gautam
Department of Computer Sc & Engineering
Sikkim Manipal Institute of Technology,
Sikkim Manipal University,Sikkim

Biswaraj Sen
Department of Computer Sc & Engineering
Sikkim Manipal Institute of Technology,
Sikkim Manipal University,Sikkim

ABSTRACT
This paper provides a study of how to design and implement Wireless Sensor Network (WSN) in NS2 (Network Simulator version 2).A wireless sensor network (WSN) consists of a number of sensors which are spatially distributed and are capable of computing, communicating and sensing.NS2 is an event-driven simulation tool that is useful in studying the dynamic nature of computer networks.NS2 provides users with executable command ns which take on input argument, which is the name of a Tool Command Language (TCL) simulation scripting file. Network Animator (NAM) is a TCL based animation tool for viewing network simulation traces and real world packet traces. Scripting languages such as AWK (Aho Weinberger Kernighan) script and PERL script can be used to calculate the performance metrics using these trace files. A simple simulation consisting of 17 nodes and calculation of average end-to-end delay and average energy consumption is shown.

Keywords
Wireless Sensor Network (WSN), Network Simulator Version2 (NS2), Tool Command Language (TCL), Network Animator (NAM), Aho Weinberger Kernighan (AWK).

1. INTRODUCTION
A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions such as temperature, sound, vibration, pressure and humidity [2]. There are a number of sensors which connect to the controllers and processing stations directly while there are other sensors who communicate the collected data wirelessly to a centralized processing station. It is for this reason that a sensor node is often not only responsible for collection of data, but also for in-network analysis, correlation, and aggregation of its own data and data from other sensor nodes. The capabilities of sensor nodes in a WSN can vary widely, that is, simple sensor nodes may monitor a single physical phenomenon, while more complex devices may combine many different sensing techniques (e.g., acoustic, optical, magnetic)[3]. They can also differ in their communication capabilities. While simple sensors may only collect and communicate information about the observed environment, more powerful sensors (i.e., sensors with large processing, energy, and storage capacities) can perform computation and aggregation of data. Wireless sensor networks have inspired many applications. Some of them are futuristic while a large number of them are practically useful. The diversity of applications in the latter category is remarkable – environment monitoring, target tracking, pipeline (water, oil, gas) monitoring, structural health monitoring, precision agriculture, health care, supply chain management, active volcano monitoring, transportation, human activity monitoring, and underground mining may also perform extensive processing and aggregation functions [2].

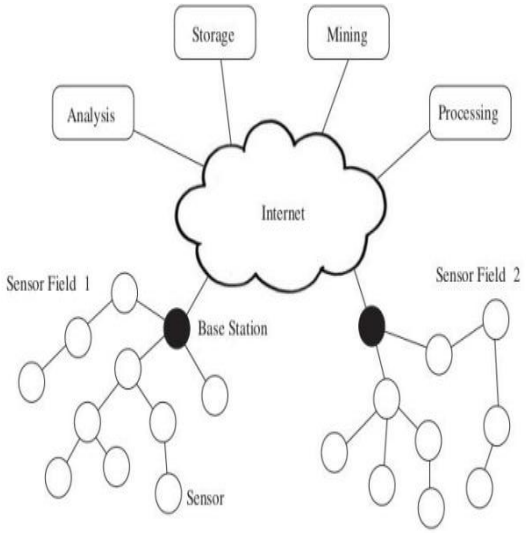


Fig 1[2]: Wireless sensor network

2. NS2 SIMULATORS
Network Simulator (Version 2), widely known as NS2, is an event- driven simulation tool that is useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989[6][10].

NS2 provides users with executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation.NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Network animator (Nam) is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces. It is mainly intended as a companion animator to the ns simulator [6][10].

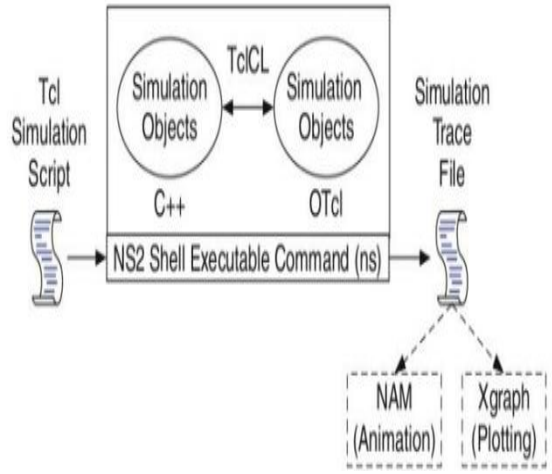


Fig 2[6]: NS2 Simulator

After the trace file is created Scripting languages such as AWK (Aho Weinberger Kernighan) script and PERL script can be used to calculate the performance metrics. Here PERL script is used to calculate the average end- to-end delay of packets from the source node to the sink node. PERL script is a general- purpose programming language originally developed for text manipulation and now used for wide range of tasks including system administration, web development, network programming and more.

3. METHODOLOGY

Sensing can be defined as a technique that is used to gather information about a physical object or process, including the occurrence of events (i.e., changes in state in rise of temperature or pressure). A device performing such a sensing task is called a sensor. The constraint most often associated with sensor network design is that sensor nodes operate with limited energy budgets. Typically, they are powered through batteries, which must be either replaced or recharged (e.g., using solar power) when depleted. For some nodes, neither option is appropriate, that is, they will simply be discarded once their energy source is depleted. Whether the battery can be recharged or not significantly affects the strategy applied to energy consumption [2]. Thus when we create sensor nodes in NS2, an energy model needs to be defined which is the energy each node has at the beginning of simulation. The components required for creating an energy model includes initialEnergy, txPower, rxPower and idlePower. The initialEnergy represents the level of energy in the node at the beginning of simulation, txPower and rxPower represents the energy consumed for transmitting and receiving the packets. And the most important component the energy model of a sensor node must contain is called the "sensePower". It denotes the energy consumed by a sensor node during the sensing operation. Apart from these components it is important to specify the communication range "RXThresh" and the sensing range "CSThresh" of a node.
The following code will set the "RXThresh" and "CSThresh" to 40 meters.

Phy/WirelessPhy set CSThresh_ 40;

Phy/WirelessPhy set RXThresh_ 40;

The energy model can be created using the following code:

 # Energy model

*$ns node-config -energyModel EnergyModel *

*-initialEnergy 50 *

-txPower 0.75

*-rxPower 0.25 *

- idlePower 0.04

-sensePower 0. 10

The next step in simulation is tracing. Tracing is a necessary technique for each discrete-event simulator, because it tells us what has happened inside the model during its running and the output analysis is also based on the tracing results. Normally for a simulation tool, trace data can be either displayed directly during execution of the simulation, or stored in a file to be post- processed and analyzed. Ns-2 supports the latter one better, though Nam (an animation tool designed for working with ns-2) can implement the first one to a certain extent.Ns-2 is able to trace all packets that are received, dropped and sent by agents, routers[6]. Before the simulation starts running, we should tell ns 2 what events we want to trace:

set tracef [open simple.tr w]

This command creates an object tracef and opens the file simple.tr in write mode. The format of the trace file is presented below:

Type Identifier	Time	Source Node	Destination Node	Packet Name	Packet Size	Flags	Flow ID	Source Address	Destination Address	Sequence Number	Packet Unique ID

Fig 3[6]: Trace file format

4. RESULT AND ANALYSIS

4.1 Simulation Scenario

Sixteen wireless sensor nodes are created. Node 17 is labeled as the sink node. Communication between the nodes is achieved using UDP. A CBR session is created between each node and the sink node. The simulation has been carried out for 100 milliseconds. The routing protocol used is DSDV and the MAC protocol used is 802.11.

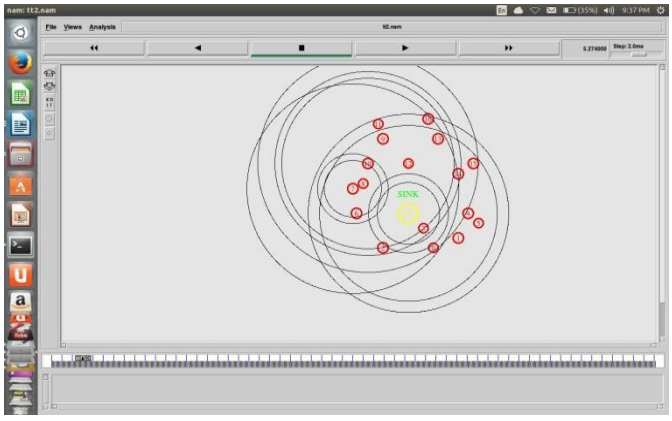


Fig 4: Nam output showing the sink node and the other nodes

4.2 Calculation of Average End-to-End Delay and Total Energy Consumed

The trace file simple.tr is obtained is shown in figure 5. In addition to the packets that are received, dropped and sent by

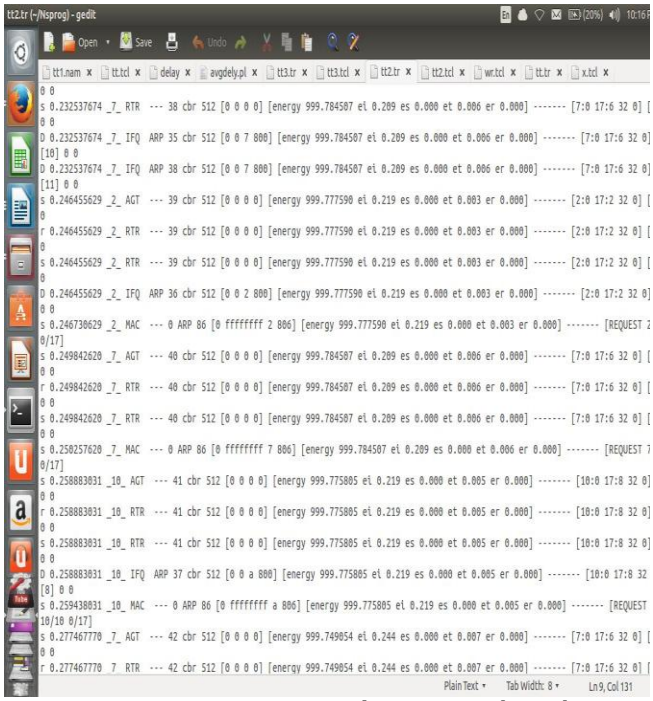
the nodes, the trace file also shows the energy model. [Energy e_i e_s e_t e_r]. Where,

Energy represents the energy remaining,

e_i = energy in idle mode,

e_s= energy used in sensing

e_t= energy used in transmitting the packets



e_r = energy used in receiving the packets

Fig 5: Trace file simple.tr

A PERL script Avgdelay.pl is written to calculate the average end-to-end delay of packets from the source nodes to the sink node. The command used to execute it in the terminal is given by:

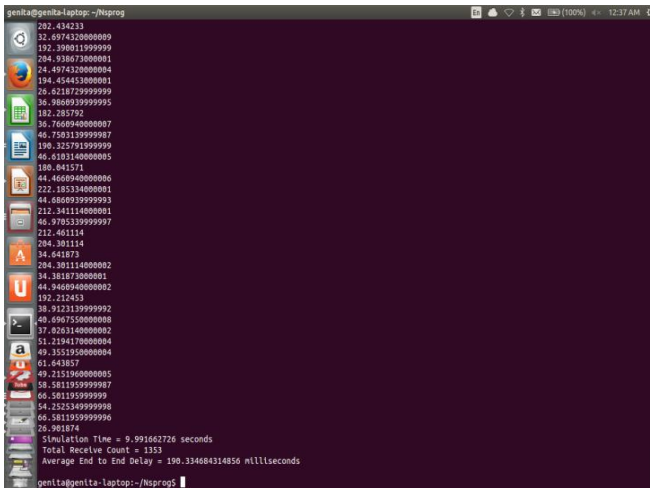


Fig 6: Calculation of average end-to-end delay using PERL script

An AWK Script energy.awk is written to calculate the average energy consumed in the network .The command to execute in the terminal is given by:

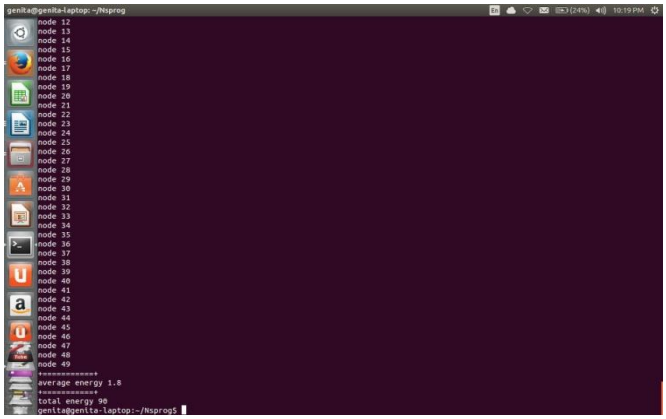


Fig 6: Calculation of average energy consumed using AWK script

5. CONCLUSION

This paper provides an overview of implementation of WSN in NS2. A simple simulation detail of creating a WSN is also presented. Tracing is done to capture all packets that are received, dropped and sent. The trace file obtained is simple.tr. The trace file has then been used to calculate the average end -to-end delay of the packets during the simulation via a PERL script Avgdelay.pl. Also the trace file is used to calculate the average energy consumed in the network via an AWK script energy.awk. In future the calculation of average energy consumption and average end-to-end delay can be represented using XGraph.

6. REFERENCES

[1] Jianliang Zheng and Myung J. Lee,"A Comprehensive Performance Study of IEEE 802.15.4",IEEE, Aug. 1999.

[2] Waltenegus Dargie and Christian Poellabauer," FUNDAMENTALS OF WIRELESS SENSOR NETWORKS", John Wiley & Sons, Ltd, pp.25-30, 2010

[3] SUHAS J. PATIL and B. R. CHANDAVARKAR,"HOMOGENEOUS SIMULTI-INTERFACE MOBILE NODE SUPPORT IN NS2"International Journal of Communication Network Security ISSN: 2231 – 1882, Volume-1, Issue-4, 2012

[4] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems

[5] Security Model Using NS2 "International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 4 Issue 1 May 2014.

[6] Introduction-to-network-simulator 2"ns2blogger.blogspot.in/",2014/04.

[7] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.

[8] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.

[9] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender.

[10] Teerawat Issariyakul, Ekram Hossain,"Introduction to Network Simulator 2", Springer US,2008, pp 1-18.