Comparison of OMNET++ and Other Simulator for WSN Simulation

Xiaodong Xian, Weiren Shi and He Huang
College of Automation, Chongqing University , Chongqing, 400044, China
Email: xxd@ccee.cqu.edu.cn, wrs@cqu.edu.cn, dick_hh@163.com

***Abstract*-Wireless sensor networks have gained considerable attention in the past few years. In this article we present a WSN simulator — OMNET++. Through compare with some well-known simulator proves OMNET++ has better performance than NS2 and OPNET. We demonstrate the use of the WSN simulation by implementing Directed Diffusion protocols, and perform performance comparisons (in terms of the execution time incurred and memory usage) in simulating WSN in OMNET++ and NS2. The simulation study indicates the WSN in OMNET++ is much more scalable than NS2. It showed that OMNET++ is better than other simulator in large-scale WSN simulation.**

I. INTRODUCTION

Wireless Sensor Networks[1-3](WSN) is consist by lots of sensor nodes which densely deployed in a region. It is a multi-hop ad hoc network system which through the wireless communication to sense, collect and process the objects in sensor region and send the messages to users. There are many potential applications of sensor networks including military, environmental and health related areas. Because of this, academe and industry pay more and more attention to it.

Nowadays WSN network project become more complex and the scale become larger, researchers must grasp the network simulation technology. The simulation researchers can study the wireless sensor network in the environment which can be controlled, they can also observe the nodes mutual function caused by the disturbance and the noise. Researchers can gain nodes detail to improve the network success ratio and reduce network maintenance works.

WSN has a large number of nodes. In order to get the excellent effect in the simulation, these requirements should be followed:

(1) WSN has a very large number of nodes, so simulation programs usually have a long debugging periods. Thus, the simulation software should place large emphasis on easy traceability and debuggability of simulation models to enhance researcher's working efficiency and the error correction ability.

(2) Usually, WSN use hierarchical structure, so we need a large number of modules to construct models, the module relations must defined in hierarchical to built reusable components .It can reduce researcher's work time and the usage of memory.

(3) WSN needs to cooperate with other software to achieve the target. Thus, software itself needs definition, customizable and open data interfaces to generate and process input and output files with commonly available software tools and allows embedding simulation models into larger applications. Embedding brings additional requirements about the memory management and reusability.

II. INTRODUCTION OF OMNET++

A. Model Structure

OMNET++[4][5] (Objective Modular Network Testbed in C++) is an object-oriented modular discrete event network simulator. In this section, we will introduce Model Structure and Logical Architecture in OMNET++.

OMNET++ model consists of simple modules and compound modules(Fig. 1). Simple modules are atomic elements in the module hierarchy: they can not be divided any further, it's the most frequent task is sending and receiving messages. Messages can be sent either via output gates, or directly to another module. Gates are the input and output interfaces of modules, they can be linked with connections. Connection can be assigned properties (such as Propagation delay , Data rate and Bit error rate). Gate via one of the several variations of the receive call to receive messages or directly transmit deliver messages from the simulation kernel.

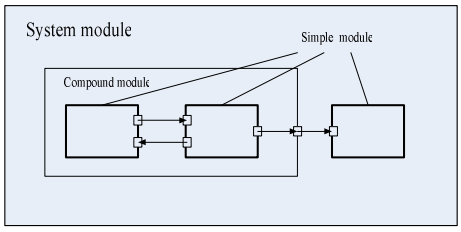

Fig. 1. Model Structure in OMNET++
(Arrows connecting and small boxes represent connections and gates.)

Compound module is consists of simple modules, compound modules and simple modules can consist another compound module, compound module hierarchy levels is unlimited. Nodes in WSN is compound modules which are consists of simple modules(Fig. 2(a)). In Fig. 2(a), layer0 module is physic layer, Application module is application layer, Coordinator modules send messages from other compound module to the right module to deal with. Energy module calculate the energy usage, Sensor module is collect the sense data. These structure is define by NED language which can edit by both text and graphical user interface(GUI)

978-1-4244-1718-6/08/$25.00 ©2008 IEEE

(Fig. 2(b)).

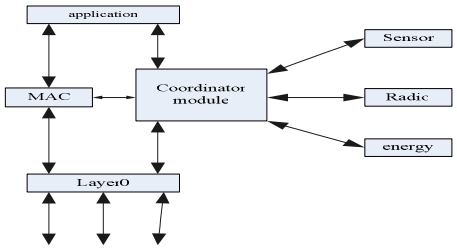

Fig. 2(a). modules in a node

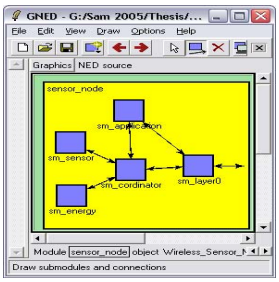

Fig. 2(b). graphical user interface

NED language can define the modules and their connection, NED description contains simple module declarations, compound module definitions and network definitions.

B. Internal Architecture

OMNET++ simulation logical architecture[6] is shown Fig. 3(a):

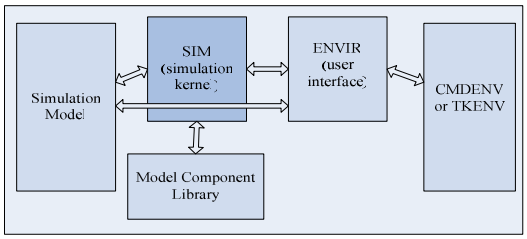

Fig. 3(a). logical architecture of OMNET++

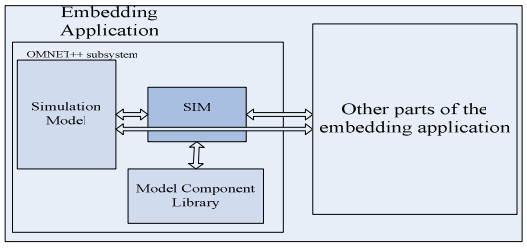

Fig. 3(b). Embedding OMNET++

SIM is the simulation kernel of OMNET++.there is an interface between SIM and ENVIR. Users can customize the full environment in which the simulation runs, and even embed an OMNET++ simulation into a larger application (Fig. 3(b)). The Model Component Library consists of the code of compiled simple and compound modules. Nowadays along with the OMNET++ widespread application more and more reusable model, network protocol and communicate models (in Fig. 3, they are simulation models) are contained in model library.

OMNET++ provides two user interfaces: TKENV and CMDENV. TKENV(Fig. 4) is OMNET++'s GUI. It provides three methods: automatic animation, module output windows and object inspectors. Automatic animation in OMNET++ is capable of animating the flow of messages on network charts, it increase the execution speed. Module out put windows is possible to open separate windows for the output of individual modules or module groups. Object inspector is a GUI window associated with a simulation object. Object inspectors can be used to display the state or contents of an object in the most appropriate way (i.e. a histogram object is displayed graphically, with a histogram chart), as well as to manually modify the object. It is automatically possible to inspect every simulation objects, there is no need to write additional code in the simple modules to make use of inspectors. These three methods made OMNET++ have easy traceability and debuggability.

CMDENV is a pure command-line interface. This feature is useful for batched simulation runs.

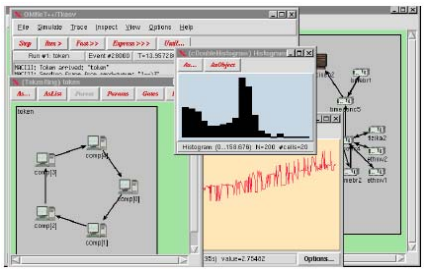

Fig. 4. Screenshot of the Tkenv User Interface of OMNeT++

III. Simulation Software Comparison

WSN has two kinds of simulation software: commercial and non-commercial. Non-commercial software has Parsec(Bagrodia et al. 1998),SMURPH (Gburzynski 1996), NS2 (Bajaj et al. 2000), Ptolemy(Davis et al. 1999), NetSim++ (Maranda et al. 1996), C++Sim(Little and McCue 1993), CLASS (Marsan et al. 1994), and commercial software has OPNET (OPNET Technologies, Inc.), EcoPREDICTOR (formerly COMNET III; Compuware Corp.).In these software NS2 is the most famous one in non-commercial software and OPNET is well-known in commercial software. This section will make a comparison of OMNET++ vs NS2[7][8] vs OPNET[9].

A. Programmability

OMNET++, NS2 and OPNET both have strong Programmability.

B. Available Protocols and Models

OPNET has lots of protocol models, including TCP/IP,

ATM, Ethernet, etc., but it is so expensive to get the licence.NS2 also has a large number of protocol models, but mostly centred around TCP/IP, that limit NS2 in WSN simulation. OMNET++ currently has TCP/IP, SCSI and FDDI models. Along with the fast increase of users, the model library also rapidly consummates and it can satisfy the large-scale sensor network simulation the demand.

OPNET is so expensive and NS2's protocol model is excessively unitary, so OMNET++ has the big advantage in the model library and the available model aspects.

C. Network Topology and Hierarchical Models

NS2 uses Tcl define network topology. This allows significant flexibility in building the topology, but it also impossible to create a graphical editor, made it inconvenience, especially for beginners. In addition, NS2 does not allow hierarchical model, it greatly limits NS2 use in WSN.

OPNET allows hierarchical models with arbitrarily deep nesting, like OMNET++.A significant difference between OMNET++ and OPNET is OPNET models always use fixed topology, while OMNET++'s NED and its graphical editor allow customize topology and parameterized topologies.

In network topology and hierarchical models, OMNET++ is better than NS2 and OPNET, NS2 is insufficient in this kind of ability and OPNET has too many limits. As WSN did not have the recognition of superior analysis topology and the network protocol, usually different environments need different topology and protocol, so the disadvantages of inflexible have a serious influence on simulation in WSN.

D. Programming Model and Simulation Library

Programming model typically uses either a thread/coroutine-based programming model, or FSMs built upon a message-receiving function(OPNET and NS2). OMNET++ is the only one which supports both programming models.

In simulation library, The NS2 simulation library provides less function. The OPNET simulation library is based on C, while the one in OMNET++ is a C++ class library.

In this aspect, OMNET++'s functions are much more powerful than NS2, C++ is better than C in WSN simulation.

E. Debugging and Tracing

Efficient debugging and tracing is an important issue in WSN. OMNET++ offers three tools for debugging purposes: automatic animation, module output windows and object inspectors. Automatic animation is supported by most of simulation software, but, other programmable simulation tools didn't have module output windows and object inspectors, partly due to the lack of a graphical runtime environment. OPNET has a powerful command-line simulation debugger, but it did not have a graphical runtime environment. NS2 is very slow in WSN simulation which limits its use.

NS2's performance of debugging and Tracing in network simulation is not very good, it needs a long time and lots of memories to simulate. OMNET++ and OPNET have good performance, but because of graphical runtime environment, OMNET++ is more convenient than OPNET.

F. Source Opening

The opening of the source code is not only necessary for embedding or modifying the simulation engine, but also provides significant help in debugging simulation models. Commercial software, and some non-commercial software do not provide source code of the simulation kernel, OPNET only provides the sources of the protocol models, OMNET++ and NS2 is fully open-sourced.

OMNET++ and NS2 are not paid for the license, and users can learn these logical architecture, thus, OMNET++ and NS2 is better than OPNET in this aspect.

Based on the facts mentioned above, simulation software in WSN, NS2's performance is not very good, OPNET has good performance (like OMNET++), but it is too expensive. So, OMNET++ is the better than other simulators in WSN simulation.

IV. Simulation Experiment Result and Analysis

OPNET is a expensive commercial software, we can not carry on the experiment to analyses it. In this section, we'll compare OMNET++ with NS2 in WSN simulation[10].

Experiment simulate Directed Diffusion under simpleMAC and IEEE 802.11 MAC[11][12] in both OMNET++ and NS2. Choose delivery ratio, runtime and memory usage to compare. Delivery ratio is node received message divides the message source node have been send, it can reflect network performance. Use the same protocol in different simulation software compare their delivery ratio can reflect OMNET++ and NS2's performance. Runtime and memory usage reflect the time and the memory usage in simulation, it is another side of software's performance. The experiment divides into 3 aspects: first, simulate Directed Diffusion[13] under simpleMAC in OMNET++ and NS2, compare its delivery ratio; than, in the same environment compare the runtime and memory usage; at last, simulate Directed Diffusion under IEEE 802.11 MAC in OMNET++ and NS2 compare the runtime and memory usage. Through comparing in OMNET++ and NS2 can prove OMNET++ have excellent performance in WSN simulation.

Following experiments use 500, 700, 900, 1100, 1300, 1500, 1750 and 2000 nodes, sensor field is 500×500m, 300s simulation time, 10 and 100 query generating nodes.

A. Comparison of Delivery Ratio in Directed Diffusion with SimpleMAC

Directed Diffusion under simpleMAC, N sensor nodes are randomly placed in a grid. Randomly a few nodes send REQ queries towards a region of interest, REQ saves the path when first sending. When a node receives an interest message, it first checks if it has the property list of its neighbors. The property list that the node maintains is the distance from the neighboring node to the final destination and the energy level's of the neighboring nodes. If the node has this neighbor

list it checks the last updated time of the neighbor list. If this time is with in the permissible limit, this information is used to decide the next hop neighbor. If the neighbor list does not exist or the last updated time is over the desired time, then the beacon messages are sent out. All the neighboring nodes will receive this beacon message. The neighboring nodes then send back beacon reply messages, which update these properties in the neighbor list. The next hop neighbor decision is based on the GEAR[14] protocol. After the REQ query reaches a region of interest, it is flooded to all the nodes in this region. A visited node list is maintained to avoid going into a loop. When a node in the region of interest receives an interest it sends back an REP message to the source of the REQ. The REP message follows the reverse path taken by the REQ. It gets the reverse path information from the nodes. When this REP message reaches the source node, the source node reinforces the path by sending back reinforcements. On the arrival of reinforcements, the nodes in the region of interest start sending back data messages. At regular intervals some data messages are marked as REP. When the source receives a data message marked as REP it sends reinforcements to rebuild the path. This would repair any holes that might have formed in the path.

In experiment we setup with queries generated by 10 nodes at random locations in the network. Fig. 5 is delivery ratio in NS2 and OMNET++ when change the number of nodes under the same simulation environment.

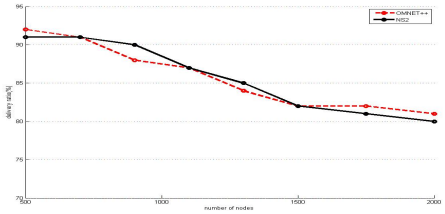

Fig. 5. Comparison of Delivery Ratio in same environment

In this figure, both OMNET++ and NS2's delivery ratio reduce when number of nodes increase, they almost have the same delivery ratio. We need to use execution time and memory usage to test the performance of simulation software.

B. Comparison of execution time and memory usage in DirectedDiffusion with SimpleMAC

Fig. 6 and Fig. 7 give the execution time when setup 10 and 100 queries generate nodes. It showed similar results at less number of nodes in the network. As the number of nodes in the network increases, OMNET++ incurs a smaller execution time. We did not test number of nodes more than 2000, because NS2 ran out of memory when number of nodes is more than 2000.

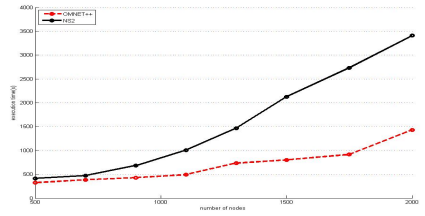

Fig. 6. execution time of 10 queries generate nodes (SimpleMAC)

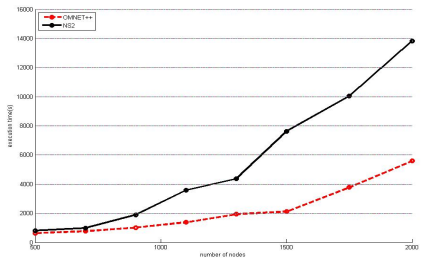

Fig. 7. execution time of 100 queries generate nodes (SimpleMAC)

We also measure the amount of memory allocated before the end of the simulation. The memory usage before the end of the simulation represents the amount of memory allocated to complete the 300 s simulation. As shown in Fig. 8 and Fig. 9, OMNET++ use less memory than NS2. This demonstrates that the data structures are used in a more scalable manner in OMNET++ to represent different classes and their interaction in the WSN framework. and the better garbage collection mechanism used in OMNET++ to reclaim unused memory.

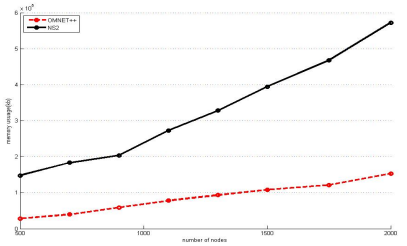

Fig. 8. memory usage of 10 queries generate nodes (SimpleMAC)

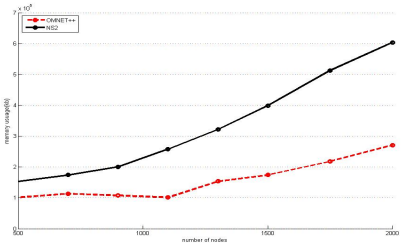

Fig. 9. memory usage of 100 queries generate nodes (SimpleMAC)

C. Comparison of execution time and memory usage in DirectedDiffusion with IEEE 802.11 MAC

Fig. 10 and Fig. 11 shown the execution time when setup 10 and 100 queries generate nodes. Curve in this figure is almost the same with curve in simpleMAC, have similar results at less number of nodes in the network, but when number of nodes increase, NS2 use more execution time than OMNET++.

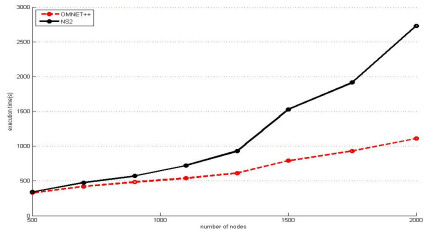

Fig. 10. execution time of 10 queries generate nodes (IEEE 802.11MAC)

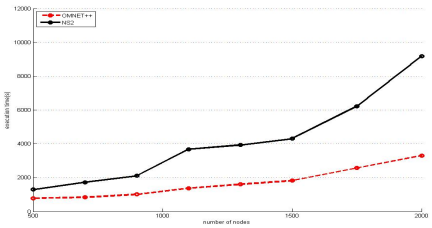

Fig. 11. execution time of 100 queries generate nodes (IEEE 802.11 MAC)

Fig. 12 and Fig. 13 give the memory usage under 802.11b MAC. Result is the same, OMNET++ use less memory than NS2. It also reflects the advantage of OMNET++ in memory usage under IEEE 802.11 MAC.

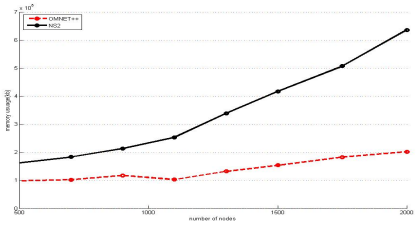

Fig. 12. memory usage of 10 queries generate nodes (IEEE 802.11 MAC)

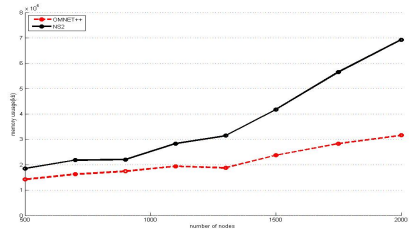

Fig. 13. memory usage of 100 queries generate nodes (IEEE 802.11 MAC)

In summary, the ability of OMNET++ to carry out simulation at the cost of a reasonable amount of memory and execution time, demonstrates the scalability of the WSN simulation framework in OMNET++. It make the OMNET++ be a very attractive candidate in evaluating newly emerging protocols for WSNs and their composite performance under various simulation scenarios.

V. CONCLUSIONS

In this article we proves that OMNET++ is an excellent WSN simulation software, its functions follows requirements of WSN simulation. Compare with NS2, it reflects that OMNET++ have better performance than NS2, also have some advantages when comparing with OPNET which is an expensive commercial software.

Directed Diffusion simulation under simpleMAC and IEEE 802.11 MAC in both OMNET++ and NS2 proved OMNET++ use less execution time and memory usage in WSN simulation. Execution time and memory usage's performance indicates that OMNET++ is much more scalable. Its excellent performance, animating graphical user interface, convenience topology describing language and easy operation obtains more and more user's favor.

In future, we will define and implement in OMNET++ generic classes for several newly emerging WSN functions, such as coverage, time indexing, and power management.

ACKNOWLEDGMENT

This work is supported by the PhD. Programs Foundation of Ministry of Education of China under Grant No. 20060611010, and Natural Science Foundation of Chongqing, China under Grant No. CSTC 2006BB2191.

REFERENCES

[1] Sun Limin, Li Jianzhong, Chen Yu, "Wireless Sensor Networks", *Tsinghua University Press*,2005.

[2] Youssef, Mohamed, El-Sheimy, Naser, "Wireless Sensor Network: Research vs. Reality Design and Deployment Issues", *Communication Networks and Services Research, 2007. CNSR '07. Fifth Annual Conference* on May 2007 Page(s):8 – 9.

[3] Li Shihan, Bai Yuebin, Qian Depei, "Research on Software Technology for Wireless Sensor Networks", *Application Research of Computers*, 2007, 01.

[4] OMNET++,http://www.omnetpp.org/

[5] András Varga. "OMNET++—Discrete Event Simulation System Version 3.2 User Manual".

[6] András Varga. "The OMNET++ Discrete Event Simulation System".

[7] NS-2, http://www.isi.edu/nsnam/ns/

[8] Xu Leiming, Pang Bo, Zhao yue, "NS and Network Simulation" [M].*Post & Telecom*, 2003.

[9] Wu Junhong, Yang Yang, etc., "Network Simulation Method and OPNET's Simulation", *Technology Computer Engineering*, 2004, 30(5): 106-108.

[10] Sam Phu Manh Tran,T, Andrew Yang, "OCO: Optimized Communication & Organization for Target Tracking in Wireless Sensor Networks", *IEEE International Conference on Sensor Networks*, Ubiquitous, and Trustworthy Computing, Vol 1 (SUTC'06) pp. 428-435.

[11] "IEEE Std. 802. 11", Wireless LAN Media Access Control (MAC)and Physical Layer (PHY) Specifications, 1999.

[12] Li Chunshi， Wang Guangxing, "A Survey of the IEEE 802.11 MAC Protocol in Wireless Ad Hoc Networks", *Computer Science* 2007 Vol.34 No.1 P.26-28.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. ACM Int'l. Conf. Mobile Comp. and Networking*, Aug. 2000.

[14] Zhao Haixia, "Secure geographical and energy aware routing protocol in wireless sensor networks", *Information Technology*, 2006, 09 .