ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

Simulation of Underwater Channel Estimation Based on Different Node Placement Using NS2

C.Haripriya[1], S.Aarthi[2], S.Bhavani[3], A.Deepika[4]

Assistant Professor, Dept. of ECE, S.A Engineering College, Chennai, Tamilnadu, India[1]

UG Student, Dept. of ECE, S.A Engineering College, Chennai, Tamilnadu, India[2]

UG Student, Dept. of ECE, S.A Engineering College, Chennai, Tamilnadu, India[3]

UG Student, Dept. of ECE, S.A Engineering College, Chennai, Tamilnadu, India[4]

ABSTRACT: In this project we aim to communicate from under water to the surface and from the surface to under water. The nodes are used and the data packets are transferred between source node and destination node with the single base station. The malicious nodes are removed to remove the delay and so the delay is reduced in this project. The droptail mechanism used in this project so that it will just drop the extra packets of data and transmits the other data. This help to reduce the packet loss or data loss. The xgraphs are generated in this project for four parameters such as end to end delay, throughput, residual energy and packet loss ratio. This project is simulated using NS2 software to see the most actual result. The node strength are assigned, ring like coverage area is formed for transmission. UWSN is the underwater sensor network used for transmission in underwater and WSN wireless sensor network is used above the surface for communication. The nodes are placed in shallow water in the form of a cluster to reduce delay and efficient transmission. Thus the program results in improved throughput, residual energy, reduced end to end delay and packet loss ratio.

KEYWORDS: Delay, Droptail mechanism, UWSN

I.INTRODUCTION

In mission-critical applications, such as battlefield reconnaissance, fire detection in forests, and gas monitoring in coal mines, wireless sensor networks (WSNs) are deployed in a wide range of areas, with a large number of sensor nodes detecting and reporting some information of urgencies to the end-users. As there may be no communication infrastructure, users are usually equipped with communicating devices to communicate with sensor nodes. When a critical event (e.g., gas leak or fire) occurs in the monitoring area and is detected by a sensor node, an alarm needs to be broadcast to the other nodes as soon as possible,. Then, sensor nodes can warn users nearby to flee or take some response to the event.

A Wireless sensor network is spatially distributed autonomous sensors to *monitor* physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location.

A sensor node consumes battery power in the following four operations: sensing data, receiving data, sending data, and processing data. Generally, the most energy consuming component is the RF module that provides wireless communications. Consequently, out of all the sensor node operations, sending/receiving data consumes more energy than any other operations. The energy consumption for transmitting 1 bit of data on the wireless channel is equivalent to the energy required to execute thousands of cycles of CPU instructions . Therefore, efficient use of energy in WSN communication protocols extends the network lifetime. Hence, any MAC, network, and transport layer protocols designed for WSN should give due consideration to the efficient use of RF moduleby minimizing MAC collision,control message overhead in routing, efficient sleep/wake scheduling and so on. In addition, during protocol design, the limited resources of sensor nodes should also be considered, which includes low processing power, less memory, short-range communication, and low sensing power.

As sensor nodes for event monitoring are expected to work for a long time without recharging their batteries, sleep scheduling method is always used during the monitoring process. Obviously, sleep scheduling could cause transmission

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

delay because sender nodes should wait until receiver nodes are active and ready to receive the message. The delay could be significant as the network scale increases. Most of sleep scheduling method focus on minimizing the energy consumption. Actually, in the critical event monitoring, only a small number of packets need to be transmitted during most of the time. When a critical event is detected, the alarm packet should be broadcast to the entire network as soon as possible. Therefore, broadcasting delay is an important issue for the application of the critical event monitoring. To minimize the broadcasting delay, it is needed to minimize the time wasted for waiting during the broadcasting.

II.RELATED WORK

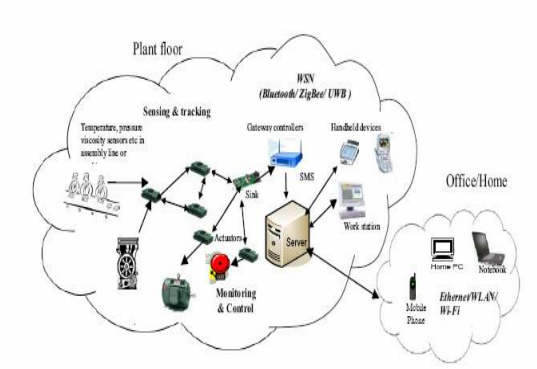

Fig.1.WSN Industrial Applications

- Distributed Intelligent Sensing System for
- Factory automation
- Process Control
- Real-time monitoring of machinery's health
- Detection of liquid areas
- Real time inventory management gas leakage
- Remote monitoring of contaminated

In this program, five segments are used. The five segments being, Initialization, file creation, configuration, traffic control and final procedure.theawk is used to store the formula used in the program. The MAC 802.11 which is wireless sensor Network is initialized. The datas are transmitted serially.

The drop tail mechanism is used with the advantage that it will just drop the extra bytes in the packet. The antenna used is an omnidirectional antenna. The Q length is designed to be 50 bits per transmission. The maximum of 59 nodes are created in this program. The AoDV routing protocol with the MOSTR supporting protocol is used for transmission of data packets.

Graph creation:
The x and y axis are created with the value of 1000 and 600 respectively and the z axis is assigned to zero. The energy node is created as 100 Jules initial, transmission power be 0.175 Jules and the received power is also 0.175 Jules.

File creation:
The topography command creates x and y and that checks the nodes space. The trace files are created which gives the result at the end.

Now the programs are linked together. The nam file is created which is the executable file. It is designed in the write mode so that it will write every information for execution. Now again the program is linked with it. The God file is created to support the program.

All the files are now configured and then the node creation is made. The node size can be assigned. When the size increases the coverage area increases ans so the drain will also be more.The size we deisgned is 20 and 35 for different nodes. The nodes are labeled and the speed is assigned.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

UDP:
The UDP protocol is used as it is connectionless protocol. So, the destination address will not be given and the acknowledgement will not arrive in return. But this helps in faster communication. The source agent and destination agent are connected through the channel. The data are transmitted to the channel. The source agent will use a antenna to send data and the antenna used in the destination agent will receive the data that is transmitted. Now the source agents are activated for that purpose.

Timing:
The interval timing between two transmission is 0.05 sec. The start and stop time are assigned by us. The bits transmitted per second can be 10 to 20 bits for the Q length of 20.

Finish:
All the files are called using calling function and all the files are linked together. The .add command is used to add all the program functions. Now the xgraph linking is made to generate xgraph for throughput, end to end delay, Residual energy.

III.PROPOSED SYSTEM

This method focus the channel estimation parameters such as end to end delay, throughput, residual energy and packet loss ratio of underwater node placement in shallow water. All data (bathymetry, altimetry, temperature, salinity, depth) are real and have been sourced from different satellite databases.

IV.APPLICATION

Area monitoring:
Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detect enemy intrusion; a civilian example is the geo-fencing of gas or oil pipelines. Area monitoring is most important part.

Health Care Monitoring:
The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.

Air Pollution Monitoring:
Wireless sensor networks have been deployed in several cities (Stockholm, London and Brisbane) to monitor the concentration of dangerous gases for citizens. These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

Forest Fire Detection:
A network of Sensor Nodes can be installed in a forest to detect when a fire has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

Landslide detection:
A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

Water quality monitoring:

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

Water quality monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.

Natural disasters prevention:
Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

Machine health monitoring:
Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionality. In wired systems, the installation of enough sensors is often limited by the cost of wiring. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors.

Data Logging:
Wireless sensor networks are also used for the collection of data for monitoring of environmental information, this can be as simple as the monitoring of the temperature in a fridge to the level of water in overflow tanks in nuclear power plants. The advantage of WSNs over conventional loggers is the "live" data feed that is possible.

Water/Waste Water Monitoring:
Monitoring the quality and level of water includes many activities such as checking the quality of underground or surface water and ensuring a country's infrastructure for the benefit of both human and animal

V.TESTING

Testing Principles:

Before applying method to design effective test cases, a software engineer must understand the basic principles that guide software testing. Davis (DAV95) suggests a set of testing principles which have been adapted for use in this book.

All tests should be traceable to customer requirements.
Test should be planned long before testing begins.
Test pare to principle applets to software testing.
Testing should begin "in the small" and progress towards testing
Exhaustive testing is not possible.

Unit testing:

Unit testing focuses on verification errors on the smallest unit of software design-the module. Using the procedural design description as a guide, important control paths are tested to uncover errors within the boundary of the module.

The module interface is tested to ensure that the information properly flows into and out of the program unit under test. Boundary conditions are tested to ensure that the module operates properly at the boundaries established to limit of restrict processing.

Integration testing:

Integration testing is a systematic technique for constructing the program structure while conducting test to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design.

White box testing:

White box testing is some time is called glass box testing, is a test case design that uses a control structure of the procedural design to drive the test cases. Using white-box testing methods, the software engineer can drive test cases that

Guarantee that logical decisions are on the true and false sides

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

Exercise all logical decisions are on the true and false sides
Execute all loops at their boundaries and within their operational
Exercise internal data structure to assure the validity

Acceptance testing:

Finally when the software is completely built, a series of acceptance tests are conducted to enable the client to validate all requirements. The user conducts these tests rather than the system developer, which can range from informal test drive to a planned and systematical executed series of tests.

These acceptance tests are conducted over a period of weeks or months, there by uncovering cumulative errors that might degrade the system order time. In this process alpha testing and beta testing are used to uncover the errors that only the end user seems able to find.

Alpha testing:

The customer conducts the alpha test at the developer's site. The client notes the errors and usage problems and gives report to the developer. Alpha tests are conducted in a control environment.

Beta testing:

The beta testing is conducted at one or more customer's sites by the end users of the software. Unlike the alpha testing, the developer is not present. Therefore a beta test is a "live" application of the software in the environment that cannot be developed by the developer. The customer records all the problems encountered during the beta testing and reports these to the developers at regular intervals.

Black box testing:

Black box testing focuses on the functional requirements of the software. That is black box testing enables the software engineer to drive a set of input conditions that will fully exercise the requirements for a program.

Black box testing is not an alternative for white box testing techniques. Rather, it is a complementary approach that is likely to uncover different class of errors.

Black box testing attempts to find errors in the following

- Interface errors.
- Performances in data structures or external
- Performance errors.
- Initialization and termination errors.
- Incorrect or missing functions.
- All the above-mentioned errors were checked

VI.SYSTEM DESIGN

Input Design:

Input Screen must be design in such a way to give an easy navigation throughout the screen without the violation of the input validation.

Input design is the process of converting the user-originated data into a computer-based format. Inaccurate input data are the most common cause of error in data processing. The goal of an input data are collected and organized into a group and error free. Input data are collected and organized into a group of similar data. Once identified, appropriated input media are selected for processing.

The design was done with six major objectives in mind

- Effectiveness
- Accuracy
- Ease of Use
- Consistency
- Simplicity
- Attractiveness

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

Output Design:

Designing computer output should proceed in an organized, well throughout manner; the right output must be developed while ensuring that each output element is designed so that candidates will find the system easy to use effectively. The term output refers to any effect produced by a system whether displayed or executed. When we design an output we must identify the specific output that is needed to meet the system. The usefulness of the new system is evaluated on the basis of their output. The output from the computer systems is required primarily to communicate the results of processing to users.

An output generally refers to the result that is generated by the system. An application is successful only when it can produce efficient and effective reports. The reports generated must be useful for the management and for the future reference.

VII.NS2 SOFTWARE

NS2 is an object oriented simulator, written in C++, with a Tcl interpreter as a front-end. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy), and a similar class hierarchy within the Tcl interpreter (also called the interpreted hierarchy).

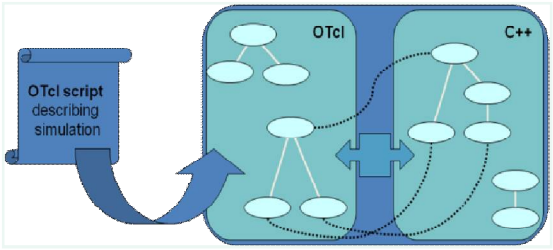

Figure 1: NS2 internal schematic diagram

NS2 uses two languages because it has two different kinds of things it needs to do: Detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation.

A large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. Tcl runs slower than C++ but can be changed very quickly (and interactively), making it ideal for simulation configuration.

Users create new simulator objects through the Tcl interpreter. These objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy.

Class Tcl Object is the base class for most of the other classes in the interpreted and compiled hierarchies. Every object in the class Tcl Object is created by the user from within the interpreter. An equivalent shadow object is created in the compiled hierarchy. The two objects are closely associated with each other.

The interpreted class hierarchy is automatically established through methods defined in the class Tcl Class. User instantiated objects are mirrored through methods defined in the class Tcl Object.

Tcl / C++ variable binding:

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

Class InstVar defines the methods and mechanisms to bind a C++ member variable in the compiled shadow object to a specified Tcl instance variable in the equivalent interpreted object. The binding is set up such that the value of the variable can be set or accessed either from within the interpreter, or from within the compiled code at all times.

Whenever the variable is read through the interpreter, the trap routine is invoked just prior to the occurrence of the read. The routine invokes the appropriate get function that returns the current value of the variable. This value is then used to set the value of the interpreted variable that is then read by the interpreter. Likewise, whenever the variable is set through the interpreter, the trap routine is invoked just after to the write is completed.

The routine gets the current value set by the interpreter, and invokes the appropriate set function that sets the value of the compiled member to the current value set within the interpreter.

The basic primitive for creating a node is:
set ns [new Simulator] $ns node
The instance procedure node constructs a node out of simpler classifier objects (to be discussed later). The Node itself is a standalone class in Tcl. However, most of the components of the node are themselves TclObjects.

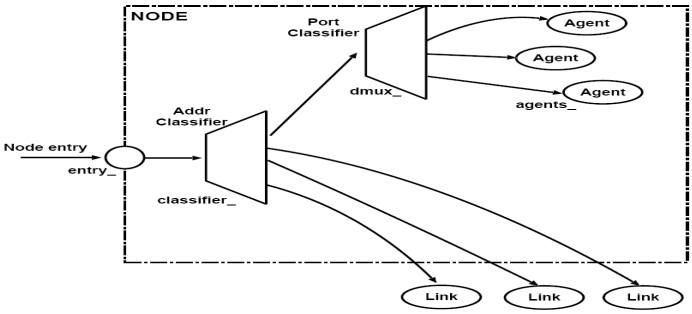

Figure2: Node structure

This simple structure consists of two Tcl Objects: an address classifier (classifier_) and a port classifier (dmux_). The function of these classifiers is to distribute incoming packets to the correct agent or to correct outgoing link.
Trace and Monitoring Support

There are a number of ways of collecting output or trace data on a simulation. Generally, trace data is either displayed directly during execution of the simulation, or (more commonly) stored in a file to be post-processed and analyzed. There are two primary but distinct types of monitoring capabilities currently supported by the simulator. The first, called *traces*, record each individual packet as it arrives, departs, or is dropped at a link or queue. Trace objects are configured into a simulation as nodes in the network topology, usually with a Tcl "Channel" object hooked to them, representing the destination of collected data (typically a trace file in the current directory). The other types of objects, called *monitors*, record counts of various interesting quantities such as packet and byte arrivals, departures, etc.

Simulator:

The simulator is an event-driven simulator. The scheduler runs by selecting the next earliest event, executing it to completion, and returning to execute the next event. Unit of time used by scheduler is seconds. Presently, the simulator is single-threaded and only one event in execution at any given time.

If more than one event is scheduled to execute at the same time, their execution is performed on the FIFO manner (first scheduled – first dispatched). No partial execution of events or pre-emption is supported.

An event generally comprises an event time, event id and a handler function. Two types of objects are derived from the base class Event - packets events and "at-events". Packets events will be discussed later in detail.

An "at-event" is a Tcl procedure execution scheduled to occur at a particular time. This is frequently used in simulation scripts. A simple example of how it is used is as follows:
set ns [new Simulator]
$ns use-scheduler Heap

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 4, April 2016

$ns at 300.5 "finish"

This Tcl code first creates a simulation object, then changes the default scheduler implementation to be heap-based, and finally schedules the function "finish" to be executed at time 300.5 (in seconds).

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as network emulation.

Network simulator:

A network simulator is a software program that imitates the working of a computer network. In simulators, the computer network is typically modelled with devices, traffic etc and the performance is analysed. Typically, users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in use today, such as WLAN, Wi-Max, UDP, and TCP.

REFERENCES

[1] S.Chen, S.Tang, M.Huang, and Y.Wang, "Capacity of data collection in arbitrary wireless sensor networks," *IEEE Trans. Parallel Distrib.Syst.*, vol. 23, no. 1, pp. 52–60, Jan. 2012.
[2] K.Akkaya and M.Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 325–349, 2005.
[3] Q.Mamun, "A qualitative comparison of different logical topologies for wireless sensor networks," *Sensors*, vol. 12, no. 11, pp. 14887–14913,2012.
[4] W.B. Heinzelman, A.P.Chandrasekasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.
[5] K.W.Fan, S.Liu, and P.Sinha, "Structure-free data aggregation in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 8,pp. 929–942, Aug. 2007.
[6] J.Kulik, W.R.Heinzelman, and H.Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc.5thAnnu. ACM/IEEE Int. Conf. Mobile Comput.Netw. (MobiCom)*, Seattle,WA, USA, Aug. 1999, pp. 174–185.
[7] C.Intanagonwiwat, R.Govindan, and D.Estrin, "Directed diffusion: A scalable and robust communicatioDDn paradigm for sensor networks,"in*Proc.6th Annu. Int. Conf. Mobile Comput.Netw. (MobiCom)*,New York, NY, USA, 2000, pp. 56–67.
[8] R.C.Shah and J.M.Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. IEEE Wireless Commun. Netw.Conf. (WCNC)*, vol. 1, Orlando, FL, USA, Mar. 2002, pp. 350–355.
[9] D.Braginsky and D.Estrin, "Rumor routing algorithm for sensor networks," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw.Appl. (WSNA)*, Atlanta, GA, USA, 2002, pp. 22–31.
[10] J.Yang, B.Bai, and H.Li, "A cluster-tree based data gathering algorithm for wireless sensor networks," in *Proc. Int. Conf. Autom. Control Artif.Intell. (ACAI)*, Xiamen, China, Mar. 2012, pp. 22–25.
[11] R.Velmani and B.Kaarthick, "An energy efficient data gathering in dense mobile wireless sensor networks," *ISRN Sensor Netw.*, Apr. 2014, Art. ID 518268. [Online]. Available:
[12] H.Li, H.Yu, and A.Liu, "A tree based data collection scheme for wireless sensor network," in *Proc. Int. Conf. Netw., Int. Conf. Syst., Int. Conf. Mobile Commun. Learn. Technol. (ICNICONSMCL)*, Morne,Mauritius, Apr. 2006, p. 119.
[13] Y.X. Jin, F.Z.Chen, G.F.Che, and W.Hu, "Energy-efficient data collection protocol for wireless sensor network based on tree," in *Proc.Asia-Pacific Conf. PWearableComput. Syst. (APWCS)*, Shenzhen, China, 2010, pp. 82–85.