P-3316

ENVIRONMENTAL MONITORING USING SENSOR NETWORKS

A PROJECT REPORT

Submitted by

R.MANJUPRIYA	**71206205025**
V.K.YAMINI	**71206205060**
V.SARANYA	**71206205304**

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI-600 025

APRIL 2010

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"ENVIRONMENTAL MONITORING USING SENSOR NETWORKS"** is the bonafide work of **R.MANJUPRIYA, V.K.YAMINI** and **V.SARANYA** who carried out the project work under my supervision.

SIGNATURE

Dr.L.S.Jayashree, Ph.D

HEAD OF THE DEPARTMENT

Department of Information Technology,
Kumaraguru College Of Technology,
Coimbatore-641006.

SIGNATURE

Dr.L.S.Jayashree, Ph.D

SUPERVISOR

Professor

Department of Information Technology,
Kumaraguru College Of Technology,
Coimbatore-641006.

The candidates with University Register Nos. **71206205025, 71206205060** and **71206205304** were examined by us in the project viva-voce examination held on 19.4.10

19/4/10

INTERNAL EXAMINER

19.4.10

EXTERNAL EXAMINER

ABSTRACT

The goal of sensor networks that monitor the environment is to detect and report the temporal and spatial dynamics of that environment and to run unattended for several months. This project proposes a novel data gathering protocol, ROPE, which takes account of the varying capacity and demand of environment monitoring applications.

Energy conservation is one of the primary requirements of WSN. Hence each node compresses its gathered data locally, transmitting a burst of data when the communication conditions are favorable. A Reactive Opportunistic Protocol, ROPE is a novel combination of following features:

Opportunistic transmission times are chosen to transmit bursts of data when conditions are good and otherwise wait, taking advantage of the generous latency requirements of environment monitoring applications.

Lightweight in-network data processing is used to minimize the amount of data transmitted, saving network energy, and aiding scientific analysis of the delivered data.

Minimal state information is stored by each node, making the network self-organizing and fault tolerant. For example, path information is only generated when needed, and data packets are not acknowledged individually.

ACKNOWLEDGEMENT

We wish to express sincere thanks and deep sense of gratitude to our learned and respected Director Dr.J.Shanmugam, Ph.D, for permitting us to undertake this project.

We are greatly indebted to our beloved Principal Dr.S.Ramachandran, Ph.D, who has been the backbone of all our deeds.

We earnestly express our gratitude and heartiest thanks to our Project Coordinator as well as our guide Dr.L.S.Jayashree, Ph.D, Head of the Department, Department of Information Technology, Kumaraguru College of Technology, for her consummate technical guidance, with constant encouragement and suggestions in carrying out this project successfully.

We thank all other staff members and other technicians for their co-operation and valuable guidance. We also thank our parents and friends who helped us to complete this project.

Our work will be incomplete without expressing our gratitude to various authors whose works were referred to carry out this project.

Finally, we thank our college library for providing us with many informative books that help us to enrich our knowledge to bring out the project successfully.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

WSN	- WIRELESS SENSOR NETWORKS
SN	- SENSOR NETWORKS
TCP	-TRANSMISSION CONTROL PROTOCOL
UDP	- USER DATAGRAM PROTOCOL
MAC	- MESSAGE AUTHENTICATION PROTOCOL
NAM	- NETWORK ANIMATOR
Tcl	- TOOL COMMAND LANGUAGE
NS2	- NETWORK SIMULATOR2
AODV	- ADHOC ON-DEMAND DISTANCE VECTOR ROUTING
ROPE	- REACTIVE OPPORTUNISTIC PROTOCOL FOR ENVIRONMENT

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 WIRELESS SENSOR NETWORKS - AN OVERVIEW

A wireless sensor network is a wireless network which consists of equally distributed autonomous devices using sensors capable of monitoring the physical or environmental conditions such as temperature, sound, vibration, pressure, motion or pollutants, at various locations. Sensor nodes are small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source. The transducer generates electrical signals based on sensed physical effects and phenomena. The microcomputer processes and stores the sensor output. The transceiver, which can be hard-wired or wireless, receives commands from a central computer and transmits data to that computer. The power for each sensor node is derived from the electric utility or from a battery.

Wireless Sensor Networks has been an active research area during recent years. Compared with traditional wireless networks, WSN has many unique features. WSN devices usually consume very low power, and they can withstand harsh environmental conditions. In addition, WSN devices are highly mobilized and they are mostly automated so that not many operations are needed. Due to these features WSN has been used in a broad variety of applications. Some typical applications are monitoring nature inhabitants, health system and eco-system, gathering data on temperature, sound and pressure, etc. The cost and size of different sensors vary a lot. Accordingly, energy, memory, computational speed and bandwidth of different sensors are very much alike.

1.2 WHAT IS SENSOR NETWORKS?

Sensor networks provide a web of interconnectivity: multiple sources of information that will allow decision-making processes to be more accurate and efficient. These processes can be complex and demanding however, and are often constrained in a number of possibly conflicting dimensions such as quality, responsiveness and cost.

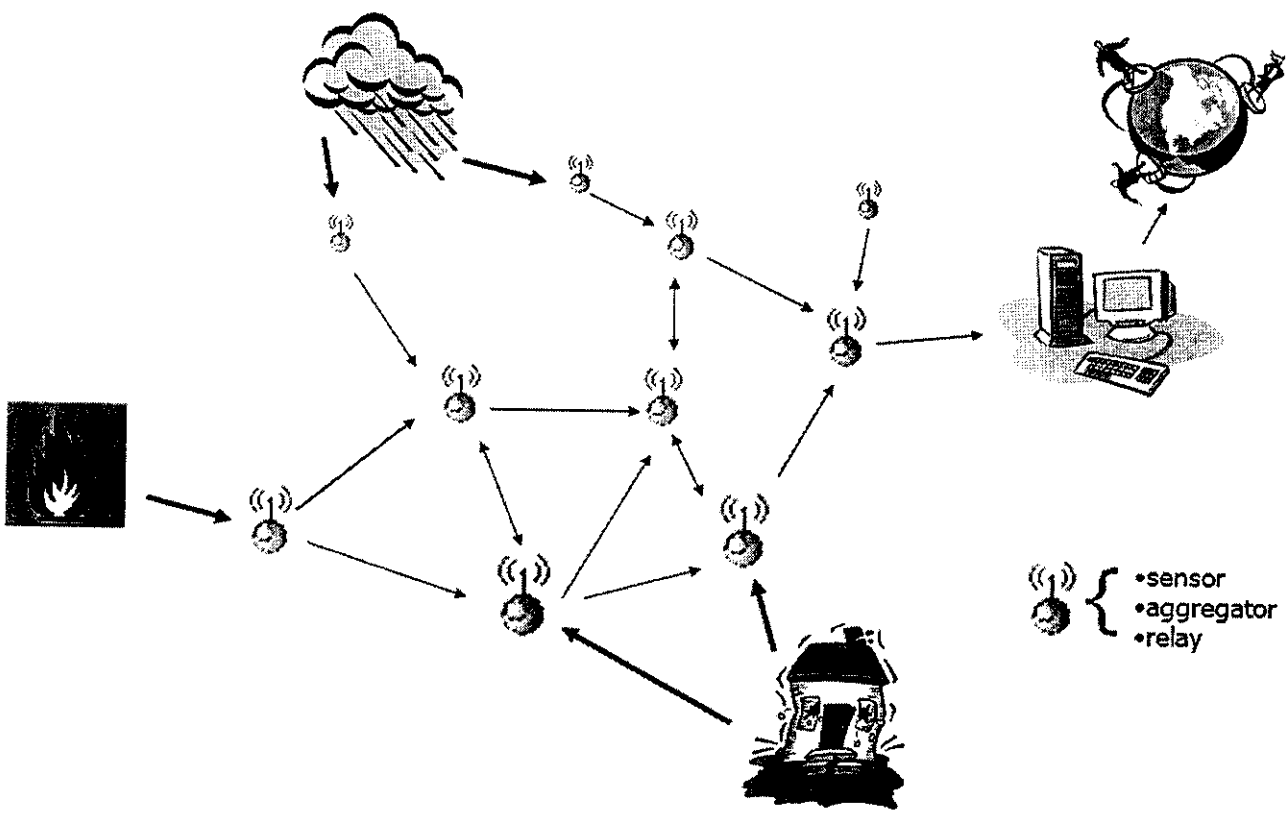

FIG 1.1 OVERVIEW OF SENSOR NETWORKS

1.3 COMPONENTS OF SENSOR NETWORKS

a. Sensing unit:

Sensing units are usually composed of two sub units: sensors and analog to digital converters. The analog signals produced by the sensors are converted to digital signals by the ADC, and fed into the processing unit.

b. **Processing unit:**

The processing unit which is generally associated with a small storage unit manages the procedure that makes the sensor nodes collaborate with the other nodes to carry out the assigned sensing tasks.

c. **Transceiver unit:**

A transceiver unit connects the nodes to the networks.

d. **Power unit:**

One of the most important components of sensor nodes is the power unit. Power units may be supported by a power scanning unit such as solar cells.

1.4 APPLICATIONS OF WSN

The applications for WSNs are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring, object tracking, nuclear reactor control, fire detection, and traffic monitoring. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes. The various applications are follows:

1. Military application
    i. Monitoring friendly forces and equipment.
    ii. Battlefield surveillance.
    iii. Nuclear, biological and chemical attack detection.
2. Environmental application
    i. Forest fire.
    ii. Bio complexity mapping of the environment.
    iii. Flood detection.
    iv. Precision agriculture.
3. Health application
    i. Tele monitoring of human physiological data.

ii. Tracking and monitoring doctors and patients inside a hospital

iii. Drug administration in hospital.

4. Home application

i. Home automation.

ii. Smart environment.

In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

The intrinsic properties of individual sensor nodes, pose additional challenges to the communication protocols in terms of energy consumption.

1.5 ENVIRONMENTAL MONITORING

Environmental monitoring is a significant driver for wireless sensor network research. Its potential to provide dynamic, real-time data about monitored variables of a landscape will enable scientists to measure properties that have not previously been observable.

Environmental monitoring describes the processes and activities that need to take place to characterize and monitor the quality of the environment. Environmental monitoring is used in the preparation of environmental impact assessments, as well as in many circumstances in which human activities carry a risk of harmful effects on the natural environment. All monitoring strategies and programs have reasons and justifications which are often designed to establish the current status of an environment or to establish trends in environmental

parameters. In all cases the results of monitoring will be reviewed, analyzed statistically and published.

The goal of sensor networks that monitor the environment is to detect and report the temporal and spatial dynamics of that environment and to run unattended for several months.

1.6 ISSUES

Considerable advances have been made in recent years in hardware and software for building wireless sensor networks. However, to ensure effective data gathering by sensor networks for monitoring remote outdoor environments, the following problems remain:

1. **Reactivity**: the ability of the network to react to its environment, and provide only relevant data to users.
2. **Robustness**: the ability of network nodes to function correctly in harsh outdoor environments.
3. **Network lifetime**: maximising the length of time the network is able to deliver data before batteries are exhausted.
4. **Communication failures**: The capacity of a sensor network to deliver sensed data can vary significantly over time as a result of communication failures.

Fortunately, most environment monitoring applications can tolerate delays of hours or even days to deliver gathered data. Furthermore, the demand from environment sensor nodes to report data is usually a very small percentage of the network's normal capacity.

1.7 EXISTING DATA GATHERING PROTOCOLS

1.7.1 GENERAL FEATURES

Existing data gathering protocols offer regular transmission slots for each node's data. They also assume that all processing of gathered data takes place centrally, after transmission, rather than locally on the gathering nodes. Most protocol maintains a routing tree from sensor nodes to one or more base stations. Sensor nodes either report their data immediately through the tree or data may be filtered and stored at each node, or then transmitted later. There is a growing body of experimental evidence that "the neighbour abstraction [of communication connectivity between nodes] is a poor approximation of reality" in outdoor wireless networks.

The standard mechanisms for overcoming communication failures between nodes include handshaking, redundant transmissions and redundant data paths.

1.7.2 STANDARD MECHANISMS USED IN EXISTING PROTOCOL

a. SMAC Protocol

This protocol does Handshaking with data acknowledgement and repeated transmission.

b. SECOAS firefly MAC Protocol

In this protocol nodes adapt to communication failure by ceasing their time division scheduled transmission when they cannot hear their neighbours and resuming when communication conditions improve.

c. Habitat Monitoring Network

This type of protocol transmits multiple times without acknowledgements.

1.7.3 DISADVANTAGES OF EXISTING PROTOCOL

In case of routing tree, the overhead for tree maintenance is high given that the volume of data to be delivered is low. Thus, maintaining a routing tree conflicts with the goal of maximising the lifetime of the sensor network. When data to be transmitted is low, overhead occurs. They assume that communication failures can be overcome by the repeated transmission of individual data items. However, this approach does not support long-lived networks, because nodes waste valuable energy on data retransmissions when communication conditions are poor or on transmitting redundant data. Lifetime of the network gets affected which leads to low data delivery rate and battery life. So Capacity and demand does not match.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 SOIL MOISTURE

Surface soil moisture is a variable that plays a crucial role in many processes occurring at the soil-atmosphere interface. The knowledge of the moisture content of the soil over a field or a catchment can be very helpful for hydrological, agronomical and meteorological applications.

Being such an extremely dynamic variable, the possibility of achieving its estimation by means of remote sensing observations is very interesting for many applications. At present, active microwave (radar) sensors represent the best alternative for a remote SM estimation for hydrologic and agronomical applications.

Monitoring soil moisture levels in the field helps to conserve water and energy, optimize crop yields, and avoid soil erosion and water pollution.

2.2 SOIL MOISTURE ANALYSIS USING SENSOR NETWORKS

In order to better manage surface water in soils, we must monitor properties such as soil moisture and temperature. Obtaining real-time, fine-grained data is critical for success, but not possible with current wired data-loggers which are both expensive, and not able to react to significant events (e.g. to increase sensing rate during a rain storm). Wireless sensor networks are a new technology that promises fine grain monitoring in time and space, and at a lower cost, than is currently possible.

2.3 TYPES OF SOIL

Soils are classified into one of about a dozen standard texture classes, based on the proportions of sand, silt, and clay particles. Sand particles are

larger than clay particles, with silt particles falling in between. For example, a soil that is 20 percent clay, 60 percent silt, and 20 percent sand (by weight) would be classified as silt loam.

Sand, loamy sand, sandy loam, loam, silt, sandy clay loam, clay loam, silty clay loam, sandy clay, silty clay, and clay are some types of soil.

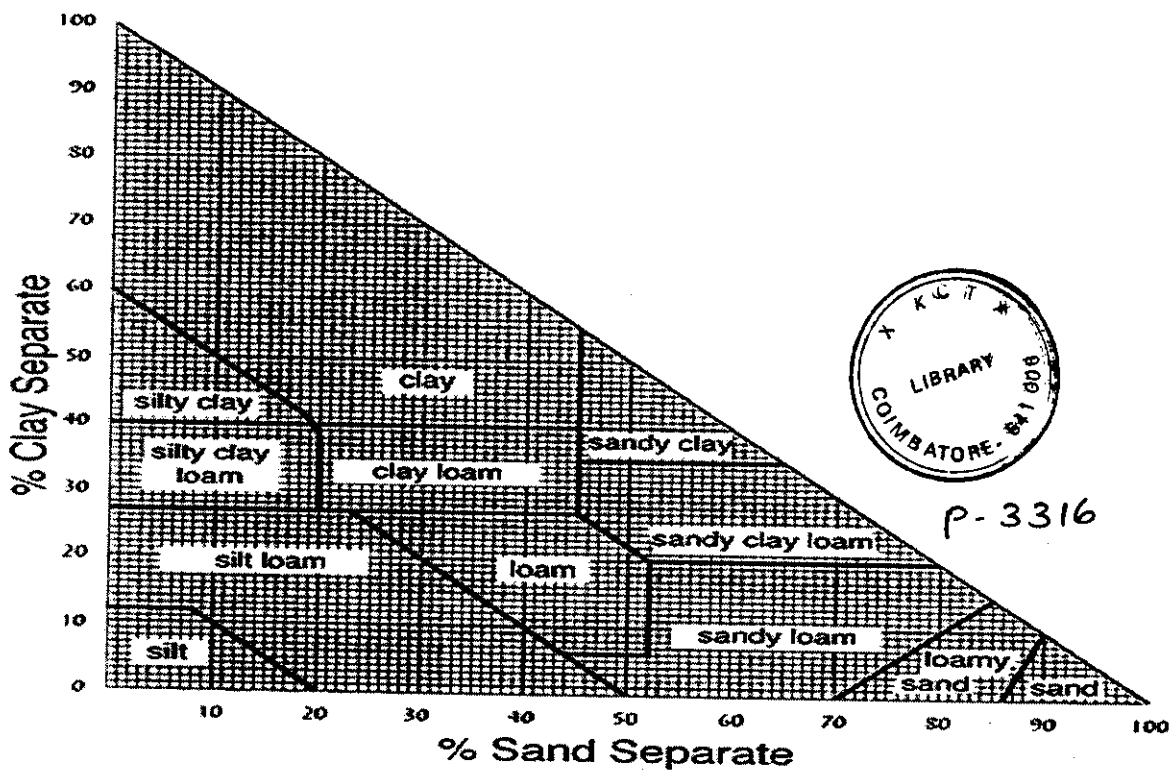

FIG 2.1 SOIL TEXTURAL TRIANGLE

2.4 HOW SOIL HOLDS WATER?

The water-holding capacity of a soil depends on its type, organic matter content, and past management practices, among other things. There are two major varieties of soils: Coarse-textured and fine-textured soils.

Coarse-textured soils have a high percentage of sand, and fine-textured soils have a high percentage of clay. Fine-textured soils generally hold more water than coarse-textured soils, although some medium-textured soils hold as much or more plant-available water than some clay soils.

Field capacity: When a balance is reached between gravitational and capillary force, water stops moving downward and is held by surface tension in the soil, a condition known as field capacity. **Permanent wilting point (PWP) or wilting point (WP)** is defined as the minimal point of soil moisture the plant requires not to wilt.

TABLE 2.1 SOIL MOITURE READINGS (CENTIBARS)

TYPE OF SOIL	MOISTURE LEVEL
Sand or loamy sand	40 – 50
Sandy loam	50 – 70
Loam	60 – 90
Clay loam or clay	90 – 120

2.5 SOIL MOISTURE MONITORING TECHNIQUES

a. Direct Inspection

The least expensive methods rely on digging up soil samples in the field and then inspecting, feeling, or weighing and drying them.

b. Feel and Appearance Method

Take walnut-sized soil samples from various locations and depths in the field, appropriate to crop's root zone. With practice and diligence, the feel and appearance method can be accurate enough for most irrigation management decisions. A soil probe, auger, or core sampler is far superior to a shovel, especially for retrieving deep soil samples.

c. Hand-Push Probe

A hand-push probe (sometimes called a Paul Brown Probe or Brown Moisture Probe) is used to determine the depth of wetted soil and also to retrieve soil samples. To determine the depth of wetted soil, push the probe vigorously into the soil by putting your weight on the handle without turning. The probe will stop abruptly when it reaches dry soil.

d. Gravimetric Weight Method

The gravimetric method involves weighing soil samples, drying them in an oven, weighing them again, and using the difference in weight to calculate the amount of water in the soil. While too time consuming to be used for day-to-day management decisions, this highly accurate and low-cost method is often used to calibrate other tools.

e. Meters and Sensors

More sophisticated devices measure some physical property that is correlated with soil moisture. Some portable sensing tools are pushed directly into the soil or into an access tube implanted in the soil. Other systems rely on buried sensors that are either hard-wired to a fixed meter or else have long attached wires (electrodes) that are left above-ground and hooked to a portable hand-held meter.

f. Data Loggers

Soil moisture data loggers are typically battery-operated devices, permanently mounted on a post and hard-wired to buried electrical resistance block sensors. At regular intervals (generally every several hours), the data logger sends a current through each sensor, measuring electrical resistance. The measurements

are converted into soil moisture readings and stored in memory. Data loggers with a graphical display show several days or weeks of readings in a bar graph.

g. Tensiometers

A tensiometer is an airtight, water-filled tube with a porous ceramic tip on the end that is placed in the soil, with a vacuum gauge on the other end that protrudes above the ground. Tensiometers measure soil water tension and display the reading on the vacuum gauge in centibars. These devices work best in the range of 0 to 80 centibars, making them better suited to coarse soils than fine soils.Tensiometers are fairly easy to use but must be serviced regularly by filling with water and using a pump to pull a vacuum. If the soil becomes too dry, tensiometers can lose soil contact, requiring re-installation.

TABLE 2.2 IRRIGATION GUIDELINES BASED ON CENTIBAR READINGS

Reading	Interpretation
0-10 centibars	Saturated soil
10-20 centibars	Most soils are at field capacity
30-40 centibars	Typical range of irrigation in many coarse soils
40-60 centibars	Typical range of irrigation in many medium soils
70-90 centibars	Typical range of irrigation in heavy clay soils
> 100 centibars	Crop water stress in most soils

2.6 ADVANTAGES OF SENSOR NETWORKS COMPARED TO OTHER TECHNIQUES

a. Adaptable to changeable environmental condition like heavy rain, dry period etc.
b. Robust transmission of collected data.
c. No need for external supervision.
d. Suitable for very large area.
e. Long life and less energy consumption.

PROPOSED METHODOLOGY

CHAPTER 3

PROPOSED METHODOLOGY

3.1 ROPE PROTOCOL

The ROPE protocol adapts to communication failures, using a different mechanism. In ROPE a node transmits a burst of data packets only if it receives feedback that the data path is reliable. In both cases, nodes do not waste energy transmitting packets when communication links are very poor. One of the most energy-expensive operations in a sensor network is transmitting a packet. Thus, the lifetime of a network can be improved significantly if each node reduces the number of packets that need be transmitted. Local data compression is particularly suitable for environment monitoring applications with generous latency requirements, because the more data a node gathers before it is required to report that data, the greater the chance for significant local compression of the data.

3.2 FEATURES OF ROPE

Opportunistic transmission times are chosen to transmit bursts of data when conditions are good, and otherwise wait taking advantage of the generous latency requirements of environment monitoring applications. **Lightweight in-network data processing** is used to minimise the amount of data transmitted, saving network energy, and aiding scientific analysis of the delivered data. **Minimal state information** is stored by each node, making the network self-organising and fault-tolerant.

3.3 THE MAIN FUNCTIONAL REQUIREMENT

1. Capture the temporal and spatial dynamics of environment variables as accurately as possible.

2. Each node monitors local environment variables, reporting significant changes to one or more gateway nodes or base stations. Base stations transfer the data to permanent storage.

3.4 NETWORK PARAMETERS

N (nodes): the number of sensing nodes in the network.

S (sensing interval): the temporal separation between readings of the monitored environment variable.

D (data compression): the effect of filtering on the size of a measured data series. Significant changes in the monitored variable need to be saved.

RP (readings per packet): the number of readings (minimum, maximum or average) in a data transmission packet.

L (latency): the maximum time that can elapse before a filtered series of readings need to be reported. The values of these parameters may change over time as a result of changes in the environment (such as increased temporal dynamics of monitored variables), or changes made by the protocol (such as altering the sensing interval or latency time).

3.5 MAXIMIZING NODE LIFE USING SLEEP WAKE CYCLE

In order to save energy, and thus maximise the lifetime of a sensor network, most network protocols synchronise their nodes in sleeping and waking cycles. The capacity of a sensor network for a given application is characterised by the following parameters. Again, the values of these parameters may change over time because of changes in the environment or they may be adjusted by the protocol.

W (waking cycle time): the temporal separation between waking periods for each node.

C (communication slot availability): the proportion of waking periods in which communication is feasible.

R (retransmission ratio): the average number of times each packet is transmitted in a good communication period, including retransmissions to overcome losses.

PW (maximum packets per cycle): the maximum number of packets that can be transmitted in a single waking period.

NW (nodes per cycle): the number of nodes allowed to transmit per waking period.

The conditions under which a specific sensor network application and environment is able to deliver its data successfully can now be expressed in terms of the demand for reporting, and the network capacity available for reporting. Capacity and demand with subscript **s** is measured in waking periods per latency period, while subscript **p** is packets per waking period.

Capacitys = C:L=W

Demands = N=NW

Capacityp = PW

Demandp = NW:R:*d*(D:(L=S)=RP*e*

The requirement for successful environment monitoring is determined by a simple relationship: capacity must be greater than or equal to demand for both **s** and **p** measures.

3.6 DATA COLLECTION IN A REACTIVE ENVIRONMENT

Each of the network parameters may have a different value under normal and extreme conditions. For example, 8 hours, is the normally acceptable

latency to report a set of readings, but latency of 36 or more hours may be acceptable in extreme conditions.

The dynamics of monitored variables changes over time and thus so does the compression factor. Communication slot availability also changes over time. In response to extreme conditions, robust data gathering protocols adapt by increasing capacity or reducing demand. Additionally, long-lived environment sensor networks need to be self-stabilising in that when nodes are added or removed from the network, the network automatically re-configures itself. For e.g. the soil moisture sensor network reacts to rain storms: frequent soil moisture readings are collected during rain (say, every 10 minutes), but less frequent readings (say, once a day) are collected when it is not raining.

Achieving reactivity with this configuration is challenging because the node monitoring rain is separated from the nodes monitoring soil moisture, and yet these nodes need to share information, whilst minimising the time spent sending, receiving and listening to messages making wireless sensor networks more reactive.

A Reactive Soil Moisture Sensor Network is an important step towards improving their effectiveness as environmental monitors. In particular, reactive monitoring reduces the amount of energy spent by each node in gathering and transmitting data, and so increases the lifetime of the network. Maximising the untethered operating time of sensor networks is an important goal for environmental monitoring in the remote locations typical of many Australian applications.

3.7 ROPE PROTOCOL DESIGN

The ROPE protocol is designed for clusters of monitoring nodes with low power resources (batteries and limited solar recharge) and short range radios connected to the base station and GSM gateway. These nodes process data locally, reporting readings of interest to a base station. The base station has

higher power resources (larger batteries) than the nodes and a long range GSM mobile phone link to a persistent database store. Network nodes perform coordinated sleep-wake cycles, synchronised by a high power broadcast by the base node.

Sensing nodes wake each cycle and listen for a synchronisation signal from the base. The network is self-stabilising since if a node misses a synchronisation signal, it estimates the time and increases its listening interval for the next round, until it is again synchronised with the base. When new nodes are added to the network, they listen for a full cycle to achieve synchronisation with the network's sleep cycle. If no signal is heard, a new node continues gathering data, but waits for several hours before trying to resynchronise.

At the beginning of each waking cycle, any node which has sufficient data to report, or any node whose reporting period has expired, contends for a transmission path to the base station. After a few packet times, any nodes that does not require data delivery they return to sleep until the next global reporting round. If more than one node tries to transmit in the same slot, the first requestor is chosen and the rest defer to other slots.

3.8 SOIL MOISTURE DEMAND AND CAPACITY

In normal conditions a node generates a minimum of 2 packets in a single slot per latency period, typically 13 packets, and in the worst case 120 packets. The maximum packet per slot capacity of ROPE is much larger than required: 1800 packets in 90 seconds during awake time. However, energy is not wasted since ROPE nodes only participate during the time they are transmitting their data. In extreme conditions, the availability of communication slots can fall to 15% or lower. No protocol can satisfy delivery constraints when communication is infeasible for an extended period. However, ROPE extends the envelope of successful operation as far as possible by first attempting to increase capacity and then decreasing demand.

Capacity and demand for slots can be adjusted by increasing latency, decreasing the wake cycle rate or reducing the number of participating nodes. Changes to the waking rate are initiated by the base station, and broadcast to all nodes in its synchronisation message. The new wake cycle should always be a divisor of the normal cycle so that any node that misses the cycle change, or is added later, is still able to find a synchronisation message and so join the network. We can reduce the number of nodes participating by increasing the allowed latency for a proportion (but not all) of the nodes to a much higher value than their neighbours.

For example, if a node hears its neighbour transmitting in extreme conditions, then it can double its own latency. That node may also slow down its sampling rate if necessary, to reduce the amount of data it must store before transmission. Although such nodes will eventually transmit more data, there is plenty of capacity for doing this. Capacity must always be greater than demand.

3.9 IN-NETWORK DATA PROCESSING

Generally data are processed in central location. But in ROPE the goal of local processing of gathered data at each node is to minimise the amount of data to be transmitted, thus minimizing the energy wastage. The generous latency requirements typical of environment monitoring applications increase the opportunity for effective compression of the data gathered during that period.

3.10 FAULT TOLERANT REPORTING

Most of the communication links are noisy, and so a percentage of packets transmitted, even in good conditions, will be lost. When communication is very poor because of conditions such as bad weather or low batteries waiting for solar recharge, it is better to delay transmission and try again later.

Some trade-offs are to be made here. Acknowledging every packet transmitted by a sensor node potentially doubles the energy cost of delivering

that data, depending on the size of ack packets. Acknowledging every packet is also susceptible to failure because of asymmetric links. However, if no acknowledgements are used, or nodes always transmit multiple times, then significant energy may be wasted in useless transmissions.

So solution for this problem is to use one feedback packet per 10 data packets transmitted. Each feedback packet contains the number of packets received in the last burst, and the sequence numbers of the missing packets. If the loss rate is too high then the sensor node abandons its connection and tries in another slot. Otherwise, a node can choose whether or not to retransmit the lost packets, allowing for data redundancy, and then continues to deliver the rest of its data. A similar feedback scheme is used when a node requests to send its data, but this time only 3 copies of the request are transmitted for feedback from the base node. A node that has no new data to transmit except its latency period health message sends that packet during the request phase of the protocol.

3.11 DATA TRANSMISSION PERFORMANCE

It is the ability of the nodes that how well they are able to identify the quality of their communication links, and how much that quality changes over time. The trials measure:

1. The temporal pattern of single hop packet transmission losses over an extended period;
2. Designed to minimise the energy used by each node, in order to maximise the overall lifetime of a network.

RESULT AND ANALYSIS

CHAPTER 4

RESULT AND ANALYSIS

4.1 THE SIMULATION OF NS2

The AODV protocol is stimulated in NS-2 network simulator, a discrete event simulator that can model and simulate single hop ad hoc wireless networks. The operating system used is Fedora 8 linux. The Distribution Coordination Function of the IEEE standard 802.11 for wireless LANs is used as a MAC layer. The simulation is carried out with Constant Bit Rate traffic. Each sender sends data packet of 512 bytes long. The simulation used a send buffer of 64 data packets, containing the data packet waiting for a root. Packets send by a routing layer are queued at the interface queue till the MAC layer can transmit them. The size of the interface queue used is 50 packets long.

4.2 RESULT AND ANALYSIS USING NAM

NAM stands for Network Animator. This tool animates the network elements as described in the tcl script. A complete visualization is available to the user which depicts the networking concepts in the project. The animator takes the tcl file as a input and creates a nam and a trace file as a output. NAM consists of tools for editing the network topology, navigation bar and a step size controller for time. With all these tools it is possible to vividly view how actually the project works with finer resolution times. The tools for editing include zoomer and controls for interfacing. A status bar at the bottom of the animator indicates the current status of the network elements.

Trace and traffic-pattern files are to be created for simulations. The pause time is at 1.0 second and packet size at 512 bytes. The application traffic pattern consists of 24 CBR nodes running on UDP within 500m*500m area. The traffic of each of CBR connections begins as per schedule from the beginning of

simulation and the simulation stops every 0.10, 0.20 or 0.25 sec depending on the region.

TABLE 4.1 SIMULATION ENVIRONMENT

Network space	500*500 metres
Simulation time	6 seconds
Number of nodes	25
Physical/MAC layer	IEEE 802.11 at 1 Mbps
Transmission range	250 metres
Interface queue	Queue/Drop Tail/Priqueue
IFQ length	50 packets
Traffic type	CBR
Load	CBR:25 to 150 kbps
Bandwidth	2e6
Routing protocol	AODV

4.3 PHYSICAL/MAC/NETWORK LAYERS SPECIFICATIONS

Channel type	Wireless Channel
Antenna type	Omni Antenna
Transmission range	250 metres
Energy Propagation model	Energy
MAC protocol	Mac 802_11
Queue type	Priority Queue
Max queue length	50
Ad hoc Routing protocol	AODV

4.4 COMPARISON GRAPH

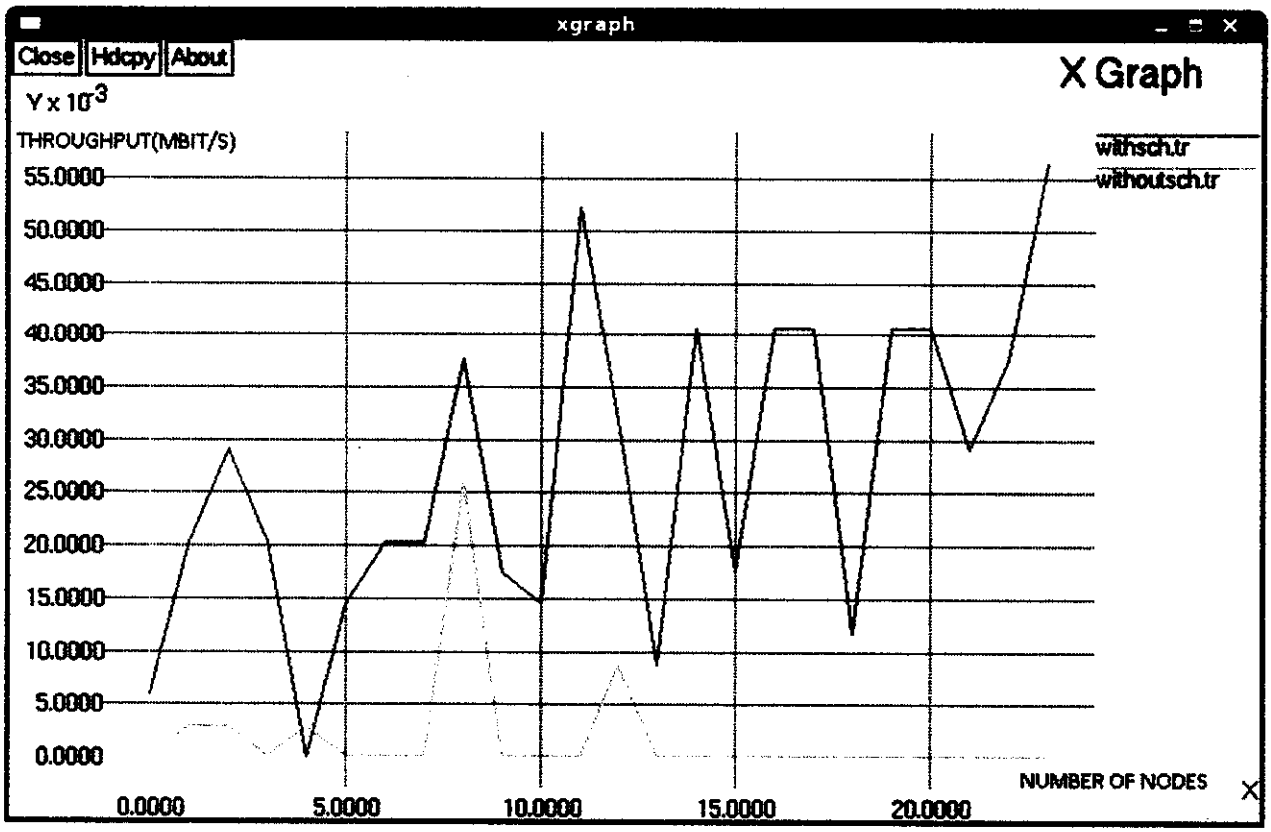

FIG 4.1 COMPARISON OF THROUGHPUT OF EACH NODE WITH SCHEDULING AND WITHOUT SCHEDULING

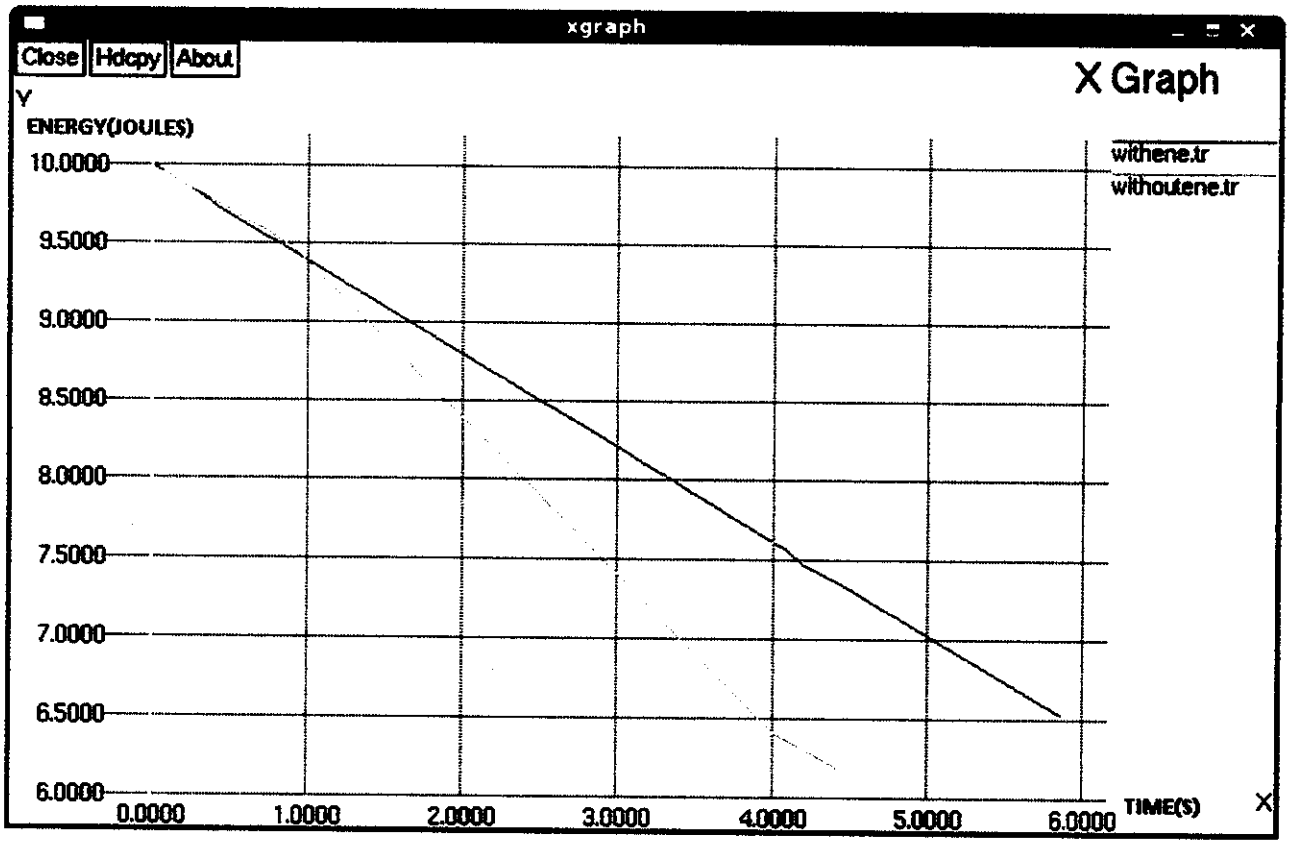

FIG 4.2 COMPARISON OF TIME VS ENERGY LEVEL OF NODES WITH SCHEDULING AND WITHOUT SCHEDULING

CONCLUSION AND OUTLOOK

CHAPTER 5

CONCLUSION AND FUTURE OUTLOOK

5.1 CONCLUSION

This protocol is used for outdoor, data-gathering sensor networks. The ROPE protocol adapts the data reporting demands of its environment dynamics to the communication capacity of its environment. The protocol is opportunistic in choosing good transmission times, and making effective use of in-network processing to reduce the network's overall transmission load. Thus, in changeable environments, ROPE is more energy efficient than existing protocols which maintain permanent routing trees and schedules, and also more robust.

5.2 FUTURE OUTLOOK

Currently the ROPE protocol is implemented for a single hop network in which the transmissions of each sensor node are able to reach the base node. Further, a multi-hop version of the protocol can be developed.

APPENDICES

SOURCE CODE

APPENDICES

APPENDIX 1

SOURCE CODE

```
#=====================================================================
# Define options
#=====================================================================
set val(chan)           Channel/WirelessChannel     ;#Channel Type
set val(prop)           Propagation/TwoRayGround    ;# radio-propagation model
set val(netif)          Phy/WirelessPhy             ;# network interface type
set val(mac)            Mac/802_11                  ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue     ;# interface queue type
set val(ll)             LL                          ;# link layer type
set val(ant)            Antenna/OmniAntenna         ;# antenna model
set val(ifqlen)         50                          ;# max packet in ifq
set val(nn)             25                          ;# number of mobilenodes
set val(rp)             AODV                        ;# routing protocol
set val(ie)             10 ;
set val(rx)             0.5 ;
set val(tx)             0.5 ;
set val(bie)            200 ;
set val(brx)            1.0 ;
set val(btx)            1.0 ;
set val(x)              500 ;
set val(y)              500 ;
set val(energymodel)    EnergyModel;
set val(radiomodel)     RadioModel;
Phy/WirelessPhy set Pt_ 0.2818;
```

```
Phy/WirelessPhy set bandwidth_ 2e6;
Mac/802_11 set dataRate_ 2.0e6;
#===================================================================
#Main Program
#===================================================================
# create simulator instance
set ns_   [new Simulator]

set g1 [open withs.tr w]
set tracefd  [open wireless-node-out.tr w]
$ns_ trace-all $tracefd
set namtrace [open wireless-node-out.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $val(x) $val(y)

#Create GOD
create-god $val(nn)

#create channel
set chan_1_ [new $val(chan)]

# configure node
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
```

```
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -energyModel "EnergyModel" \
                -idlePower 1.0 \
                -rxPower $val(rx) \
                -txPower $val(tx) \
                -sleepPower 0.001 \
                -transitionPower 0.01 \
                -transitionTime 0.001 \
               -nitialEnergy $val(ie) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace ON \
             -channel $chan_1_

#configure for mobilenodes
      for {set j 0} {$j < $val(nn)} {incr j} {
        global node_
        set node_($j) [$ns_ node]
        $node_($j) random-motion 0   ;#disable random motion
    }
```

```
# configure for base-station node
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -energyModel "EnergyModel" \
                 -idlePower 1.0 \
                 -rxPower $val(brx) \
                 -txPower $val(btx) \
                 -sleepPower 0.001 \
                 -transitionPower 0.01 \
                 -transitionTime 0.001 \
                 -initialEnergy $val(bie) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
               -channel $chan_1_

# Create Base Station node
set BS [$ns_ node]
$BS random-motion 0
```

```
# Position (fixed) for base-station node
$BS set X_ 250.0
$BS set Y_ 250.0
$BS set Z_ 0.0
#color index
$ns_ color 0 green
$ns_ color 1 red
$ns_ color 2 yellow
$ns_ color 3 black
$ns_ color 4 pink
$ns_ color 5 red
$ns_ color 6 blue
$ns_ color 7 chocolate
$ns_ color 8 orange
#Provide initial(X,Y, for now Z=0) co-ordinates for nodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 280.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 100.0
$node_(1) set Y_ 420.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 120.0
```

```
$node_(2) set Y_ 460.0
$node_(2) set Z_ 0.0
$node_(3) set X_ 190.0
$node_(3) set Y_ 380.0
$node_(3) set Z_ 0.0
$node_(4) set X_ 140.0
$node_(4) set Y_ 320.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 90.0
$node_(5) set Y_ 350.0
$node_(5) set Z_ 0.0
$node_(6) set X_ 150.0
$node_(6) set Y_ 275.0
$node_(6) set Z_ 0.0
$node_(7) set X_ 60.0
$node_(7) set Y_ 255.0
$node_(7) set Z_ 0.0
$node_(8) set X_ 350.0
$node_(8) set Y_ 140.0
$node_(8) set Z_ 0.0
$node_(9) set X_ 450.0
```

```
$node_(9) set Y_ 200.0
$node_(9) set Z_ 0.0
$node_(10) set X_ 480.0
$node_(10) set Y_ 70.0
$node_(10) set Z_ 0.0
$node_(11) set X_ 400.0
$node_(11) set Y_ 60.0
$node_(11) set Z_ 0.0
$node_(12) set X_ 350.0
$node_(12) set Y_ 380.0
$node_(12) set Z_ 0.0
$node_(13) set X_ 370.0
$node_(13) set Y_ 450.0
$node_(13) set Z_ 0.0
$node_(14) set X_ 420.0
$node_(14) set Y_ 400.0
$node_(14) set Z_ 0.0
$node_(15) set X_ 480.0
$node_(15) set Y_ 370.0
$node_(15) set Z_ 0.0
$node_(16) set X_ 430.0
```

```
$node_(16) set Y_ 300.0
$node_(16) set Z_ 0.0
$node_(17) set X_ 350.0
$node_(17) set Y_ 280.0
$node_(17) set Z_ 0.0
$node_(18) set X_ 50.0
$node_(18) set Y_ 50.0
$node_(18) set Z_ 0.0
$node_(19) set X_ 150.0
$node_(19) set Y_ 150.0
$node_(19) set Z_ 0.0
$node_(20) set X_ 200.0
$node_(20) set Y_ 100.0
$node_(20) set Z_ 0.0
$node_(21) set X_ 160.0
$node_(21) set Y_ 30.0
$node_(21) set Z_ 0.0
$node_(22) set X_ 120.0
$node_(22) set Y_ 50.0
$node_(22) set Z_ 0.0
$node_(23) set X_ 75.0
```

```
$node_(23) set Y_ 100.0

$node_(23) set Z_ 0.0


$ns_ at 0.01 "$node_(0) color red"

$ns_ at 0.01 "$node_(0) label 0"

$ns_ at 0.01 "$BS color red"

$ns_ at 0.01 "$BS label BS"

$ns_ at 0.45 "$node_(1) color red"

$ns_ at 0.45 "$node_(1) label 1"

$ns_ at 0.45 "$BS color red"

$ns_ at 0.45 "$BS label BS"

$ns_ at 1.01 "$node_(2) color red"

$ns_ at 1.01 "$node_(2) label 2"

$ns_ at 1.01 "$BS color red"

$ns_ at 1.01 "$BS label BS"

$ns_ at 1.39 "$node_(3) color red"

$ns_ at 1.39 "$node_(3) label 3"

$ns_ at 1.39 "$BS color red"

$ns_ at 1.39 "$BS label BS"

$ns_ at 2.01 "$node_(4) color red"

$ns_ at 2.01 "$node_(4) label 4"

$ns_ at 2.01 "$BS color red"
```

```
$ns_ at 2.01 "$BS label BS"

$ns_ at 2.39 "$node_(5) color red"

$ns_ at 2.39 "$node_(5) label 5"

$ns_ at 2.39 "$BS color red"

$ns_ at 2.39 "$BS label BS"

$ns_ at 3.01 "$node_(6) color red"

$ns_ at 3.01 "$node_(6) label 6"

$ns_ at 3.01 "$BS color red"

$ns_ at 3.01 "$BS label BS"

$ns_ at 3.39 "$node_(7) color red"

$ns_ at 3.39 "$node_(7) label 7"

$ns_ at 3.39 "$BS color red"

$ns_ at 3.39 "$BS label BS"

$ns_ at 0.76 "$node_(8) color red"

$ns_ at 0.76 "$node_(8) label 8"

$ns_ at 0.76 "$BS color red"

$ns_ at 0.76 "$BS label BS"

$ns_ at 1.78 "$node_(9) color red"

$ns_ at 1.78 "$node_(9) label 9"

$ns_ at 1.78 "$BS color red"

$ns_ at 1.78 "$BS label BS"
```

```
$ns_ at 2.76 "$node_(10) color red"

$ns_ at 2.76 "$node_(10) label 10"

$ns_ at 2.76 "$BS color red"

$ns_ at 2.76 "$BS label BS"

$ns_ at 3.78 "$node_(11) color red"

$ns_ at 3.78 "$node_(11) label 11"

$ns_ at 3.78 "$BS color red"

$ns_ at 3.78 "$BS label BS"

$ns_ at 0.17 "$node_(12) color red"

$ns_ at 0.17 "$node_(12) label 12"

$ns_ at 0.17 "$BS color  red"

$ns_ at 0.17 "$BS label BS"

$ns_ at 0.55 "$node_(18) color red"

$ns_ at 0.55 "$node_(18) label 18"

$ns_ at 0.55 "$BS color red"

$ns_ at 0.55 "$BS label BS"

$ns_ at 1.17 "$node_(13) color red"

$ns_ at 1.17 "$node_(13) label 13"

$ns_ at 1.17 "$BS color red"

$ns_ at 1.17 "$BS label BS"

$ns_ at 1.55 "$node_(19) color red"

$ns_ at 1.55 "$node_(19) label 19"
```

```
$ns_ at 1.55 "$BS color red"

$ns_ at 1.55 "$BS label BS"

$ns_ at 2.17 "$node_(14) color red"

$ns_ at 2.17 "$node_(14) label 14"

$ns_ at 2.17 "$BS color red"

$ns_ at 2.17 "$BS label BS"

$ns_ at 2.55 "$node_(20) color red"

$ns_ at 2.55 "$node_(20) label 20"

$ns_ at 2.55 "$BS color red"

$ns_ at 2.55 "$BS label BS"

$ns_ at 3.17 "$node_(15) color red"

$ns_ at 3.17 "$node_(15) label 15"

$ns_ at 3.17 "$BS color red"

$ns_ at 3.17 "$BS label BS"

$ns_ at 3.55 "$node_(21) color red"

$ns_ at 3.55 "$node_(21) label 21"

$ns_ at 3.55 "$BS color red"

$ns_ at 3.55 "$BS label BS"

$ns_ at 4.17 "$node_(16) color red"

$ns_ at 4.17 "$node_(16) label 16"

$ns_ at 4.17 "$BS color red"

$ns_ at 4.17 "$BS label BS"
```

```
$ns_ at 4.55 "$node_(22) color red"

$ns_ at 4.55 "$node_(22) label 22"

$ns_ at 4.55 "$BS color red"

$ns_ at 4.55 "$BS label BS"

$ns_ at 5.17 "$node_(17) color red"

$ns_ at 5.17 "$node_(17) label 17"

$ns_ at 5.17 "$BS color red"

$ns_ at 5.17 "$BS label BS"

$ns_ at 5.55 "$node_(23) color red"

$ns_ at 5.55 "$node_(23) label 23"

$ns_ at 5.55 "$BS color red"

$ns_ at 5.55 "$BS label BS"

#Provide X,Y,Z co-ordinates for nodes

$ns_ at 0.0 "$node_(0) setdest 50.0 280.0 0"

$ns_ at 0.0 "$node_(1) setdest 50.0 420.0 0"

$ns_ at 0.0 "$node_(2) setdest 120.0 460.0 0"

$ns_ at 0.0 "$node_(3) setdest 190.0 380.0 0"

$ns_ at 0.0 "$node_(4) setdest 140.0 320.0 0"

$ns_ at 0.0 "$node_(5) setdest 90.0 350.0 0"

$ns_ at 0.0 "$node_(6) setdest 150.0 275.0 0"

$ns_ at 0.0 "$node_(7) setdest 60.0 255.0 0"
```

```
$ns_ at 0.0 "$node_(8) setdest 350.0 140.0 0"
$ns_ at 0.0 "$node_(9) setdest 450.0 200.0 0"
$ns_ at 0.0 "$node_(10) setdest 480.0 70.0 0"
$ns_ at 0.0 "$node_(11) setdest 400.0 60.0 0"
$ns_ at 0.0 "$node_(12) setdest 350.0 380.0 0"
$ns_ at 0.0 "$node_(13) setdest 370.0 450.0 0"
$ns_ at 0.0 "$node_(14) setdest 420.0 400.0 0"
$ns_ at 0.0 "$node_(15) setdest 480.0 370.0 0"
$ns_ at 0.0 "$node_(16) setdest 430.0 300.0 0"
$ns_ at 0.0 "$node_(17) setdest 350.0 280.0 0"
$ns_ at 0.0 "$node_(18) setdest 50.0 50.0 0"
$ns_ at 0.0 "$node_(19) setdest 100.0 200.0 0"
$ns_ at 0.0 "$node_(20) setdest 200.0 100.0 0"
$ns_ at 0.0 "$node_(21) setdest 160.0 30.0 0"
$ns_ at 0.0 "$node_(22) setdest 120.0 50.0 0"
$ns_ at 0.0 "$node_(23) setdest 75.0 75.0 0"
$ns_ at 0.0 "$BS setdest 250.0 250.0 0"
set tcp0 [new Agent/TCP]
set sink0 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp0
$ns_ attach-agent $BS $sink0
```

```
$ns_ connect $tcp0 $sink0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns_ at 0.01 "$ftp0 start"

$ns_ at 0.10 "$ftp0 stop"

$ns_ at 0.15 "$node_(0) color blue"


set tcp1 [new Agent/TCP]

set sink1 [new Agent/TCPSink]

$ns_ attach-agent $node_(1) $tcp1

$ns_ attach-agent $BS $sink1

$ns_ connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns_ at 0.45 "$ftp1 start"

$ns_ at 0.53 "$ftp1 stop"

$ns_ at 0.54 "$node_(1) color blue"


set tcp2 [new Agent/TCP]

set sink2 [new Agent/TCPSink]

$ns_ attach-agent $node_(2) $tcp2

$ns_ attach-agent $BS $sink2
```

```
$ns_ connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns_ at 1.01 "$ftp2 start"
$ns_ at 1.10 "$ftp2 stop"
$ns_ at 1.15 "$node_(2) color blue"

set tcp3 [new Agent/TCP]
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(3) $tcp3
$ns_ attach-agent $BS $sink3
$ns_ connect $tcp3 $sink3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ns_ at 1.39 "$ftp3 start"
$ns_ at 1.48 "$ftp3 stop"
$ns_ at 1.54 "$node_(3) color blue"

set tcp4 [new Agent/TCP]
set sink4 [new Agent/TCPSink]
$ns_ attach-agent $node_(4) $tcp4
$ns_ attach-agent $BS $sink4
```

```
$ns_ connect $tcp4 $sink4
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ns_ at 2.01 "$ftp4 start"
$ns_ at 2.10 "$ftp4 stop"
$ns_ at 2.15 "$node_(4) color blue"

set tcp5 [new Agent/TCP]
set sink5 [new Agent/TCPSink]
$ns_ attach-agent $node_(5) $tcp5
$ns_ attach-agent $BS $sink5
$ns_ connect $tcp5 $sink5
set ftp5 [new Application/FTP]
$ftp5 attach-agent $tcp5
$ns_ at 2.39 "$ftp5 start"
$ns_ at 2.48 "$ftp5 stop"
$ns_ at 2.54 "$node_(5) color blue"

set tcp6 [new Agent/TCP]
set sink6 [new Agent/TCPSink]
$ns_ attach-agent $node_(6) $tcp6
$ns_ attach-agent $BS $sink6
```

```
$ns_ connect $tcp6 $sink6
set ftp6 [new Application/FTP]
$ftp6 attach-agent $tcp6
$ns_ at 3.01 "$ftp6 start"
$ns_ at 3.10 "$ftp6 stop"
$ns_ at 3.15 "$node_(6) color blue"

set tcp7 [new Agent/TCP]
set sink7 [new Agent/TCPSink]
$ns_ attach-agent $node_(7) $tcp7
$ns_ attach-agent $BS $sink7
$ns_ connect $tcp7 $sink7
set ftp7 [new Application/FTP]
$ftp7 attach-agent $tcp7
$ns_ at 3.39 "$ftp7 start"
$ns_ at 3.48 "$ftp7 stop"
$ns_ at 3.54 "$node_(7) color blue"

set tcp8 [new Agent/TCP]
set sink8 [new Agent/TCPSink]
$ns_ attach-agent $node_(8) $tcp8
$ns_ attach-agent $BS $sink8
```

```
$ns_ connect $tcp8 $sink8
set ftp8 [new Application/FTP]
$ftp8 attach-agent $tcp8
$ns_ at 0.76 "$ftp8 start"
$ns_ at 0.95 "$ftp8 stop"
$ns_ at 0.96 "$node_(8) color blue"

set tcp9 [new Agent/TCP]
set sink9 [new Agent/TCPSink]
$ns_ attach-agent $node_(9) $tcp9
$ns_ attach-agent $BS $sink9
$ns_ connect $tcp9 $sink9
set ftp9 [new Application/FTP]
$ftp9 attach-agent $tcp9
$ns_ at 1.78 "$ftp9 start"
$ns_ at 1.90 "$ftp9 stop"
$ns_ at 1.96 "$node_(9) color blue"

set tcp10 [new Agent/TCP]
set sink10 [new Agent/TCPSink]
$ns_ attach-agent $node_(10) $tcp10
$ns_ attach-agent $BS $sink10
```

```
$ns_ connect $tcp10 $sink10
set ftp10 [new Application/FTP]
$ftp10 attach-agent $tcp10
$ns_ at 2.76 "$ftp10 start"
$ns_ at 2.95 "$ftp10 stop"
$ns_ at 2.96 "$node_(10) color blue"

set tcp11 [new Agent/TCP]
set sink11 [new Agent/TCPSink]
$ns_ attach-agent $node_(11) $tcp11
$ns_ attach-agent $BS $sink11
$ns_ connect $tcp11 $sink11
set ftp11 [new Application/FTP]
$ftp11 attach-agent $tcp11
$ns_ at 3.78 "$ftp11 start"
$ns_ at 3.95 "$ftp11 stop"
$ns_ at 3.96 "$node_(11) color blue"

set tcp12 [new Agent/TCP]
set sink12 [new Agent/TCPSink]
$ns_ attach-agent $node_(12) $tcp12
$ns_ attach-agent $BS $sink12
```

```
$ns_ connect $tcp12 $sink12
set ftp12 [new Application/FTP]
$ftp12 attach-agent $tcp12
$ns_ at 0.17 "$ftp12 start"
$ns_ at 0.30 "$ftp12 stop"
$ns_ at 0.38 "$node_(12) color blue"

set tcp18 [new Agent/TCP]
set sink18 [new Agent/TCPSink]
$ns_ attach-agent $node_(18) $tcp18
$ns_ attach-agent $BS $sink18
$ns_ connect $tcp18 $sink18
set ftp18 [new Application/FTP]
$ftp18 attach-agent $tcp18
$ns_ at 0.57 "$ftp18 start"
$ns_ at 0.70 "$ftp18 stop"
$ns_ at 0.74 "$node_(18) color blue"

set tcp13 [new Agent/TCP]
set sink13 [new Agent/TCPSink]
$ns_ attach-agent $node_(13) $tcp13
$ns_ attach-agent $BS $sink13
```

```
$ns_ connect $tcp13 $sink13
set ftp13 [new Application/FTP]
$ftp13 attach-agent $tcp13
$ns_ at 1.17 "$ftp13 start"
$ns_ at 1.30 "$ftp13 stop"
$ns_ at 1.38 "$node_(13) color blue"

set tcp19 [new Agent/TCP]
set sink19 [new Agent/TCPSink]
$ns_ attach-agent $node_(19) $tcp19
$ns_ attach-agent $BS $sink19
$ns_ connect $tcp19 $sink19
set ftp19 [new Application/FTP]
$ftp19 attach-agent $tcp19
$ns_ at 1.57 "$ftp19 start"
$ns_ at 1.70 "$ftp19 stop"
$ns_ at 1.74 "$node_(19) color blue"

set tcp14 [new Agent/TCP]
set sink14 [new Agent/TCPSink]
$ns_ attach-agent $node_(14) $tcp14
$ns_ attach-agent $BS $sink14
```

```
$ns_ connect $tcp14 $sink14
set ftp14 [new Application/FTP]
$ftp14 attach-agent $tcp14
$ns_ at 2.17 "$ftp14 start"
$ns_ at 2.30 "$ftp14 stop"
$ns_ at 2.38 "$node_(14) color blue"

set tcp20 [new Agent/TCP]
set sink20 [new Agent/TCPSink]
$ns_ attach-agent $node_(20) $tcp20
$ns_ attach-agent $BS $sink20
$ns_ connect $tcp20 $sink20
set ftp20 [new Application/FTP]
$ftp20 attach-agent $tcp20
$ns_ at 2.57 "$ftp20 start"
$ns_ at 2.70 "$ftp20 stop"
$ns_ at 2.74 "$node_(20) color blue"

set tcp15 [new Agent/TCP]
set sink15 [new Agent/TCPSink]
$ns_ attach-agent $node_(15) $tcp15
$ns_ attach-agent $BS $sink15
```

```
$ns_ connect $tcp15 $sink15
set ftp15 [new Application/FTP]
$ftp15 attach-agent $tcp15
$ns_ at 3.17 "$ftp15 start"
$ns_ at 3.30 "$ftp15 stop"
$ns_ at 3.38 "$node_(15) color blue"

set tcp21 [new Agent/TCP]
set sink21 [new Agent/TCPSink]
$ns_ attach-agent $node_(21) $tcp21
$ns_ attach-agent $BS $sink21
$ns_ connect $tcp21 $sink21
set ftp21 [new Application/FTP]
$ftp21 attach-agent $tcp21
$ns_ at 3.57 "$ftp21 start"
$ns_ at 3.70 "$ftp21 stop"
$ns_ at 3.74 "$node_(21) color blue"

set tcp16 [new Agent/TCP]
set sink16 [new Agent/TCPSink]
$ns_ attach-agent $node_(16) $tcp16
$ns_ attach-agent $BS $sink16
```

```
$ns_ connect $tcp16 $sink16
set ftp16 [new Application/FTP]
$ftp16 attach-agent $tcp16
$ns_ at 4.17 "$ftp16 start"
$ns_ at 4.30 "$ftp16 stop"
$ns_ at 4.38 "$node_(16) color blue"

set tcp22 [new Agent/TCP]
set sink22 [new Agent/TCPSink]
$ns_ attach-agent $node_(22) $tcp22
$ns_ attach-agent $BS $sink22
$ns_ connect $tcp22 $sink22
set ftp22 [new Application/FTP]
$ftp22 attach-agent $tcp22
$ns_ at 4.57 "$ftp22 start"
$ns_ at 4.70 "$ftp22 stop"
$ns_ at 4.74 "$node_(22) color blue"

set tcp17 [new Agent/TCP]
set sink17 [new Agent/TCPSink]
$ns_ attach-agent $node_(17) $tcp17
$ns_ attach-agent $BS $sink17
```

```
$ns_ connect $tcp17 $sink17
set ftp17 [new Application/FTP]
$ftp17 attach-agent $tcp17
$ns_ at 5.17 "$ftp17 start"
$ns_ at 5.34 "$ftp17 stop"
$ns_ at 5.38 "$node_(17) color blue"

set tcp23 [new Agent/TCP]
set sink23 [new Agent/TCPSink]
$ns_ attach-agent $node_(23) $tcp23
$ns_ attach-agent $BS $sink23
$ns_ connect $tcp23 $sink23
set ftp23 [new Application/FTP]
$ftp23 attach-agent $tcp23
$ns_ at 5.55 "$ftp23 start"
$ns_ at 5.75 "$ftp23 stop"
$ns_ at 5.76 "$node_(23) color blue"


$ns_ at 4.01 "$node_(0) color red"
$ns_ at 4.01 "$node_(0) label 0"
$ns_ at 4.01 "$BS color red"
$ns_ at 4.01 "$BS label BS"
```

```
$ns_ at 4.39 "$node_(1) color red"

$ns_ at 4.39 "$node_(1) label 1"

$ns_ at 4.39 "$BS color red"

$ns_ at 4.39 "$BS label BS"

$ns_ at 5.01 "$node_(2) color red"

$ns_ at 5.01 "$node_(2) label 2"

$ns_ at 5.01 "$BS color red"

$ns_ at 5.01 "$BS label BS"

$ns_ at 5.39 "$node_(3) color red"

$ns_ at 5.39 "$node_(3) label 3"

$ns_ at 5.39 "$BS color red"

$ns_ at 5.39 "$BS label BS"

$ns_ at 6.01 "$node_(4) color red"

$ns_ at 6.01 "$node_(4) label 4"

$ns_ at 6.01 "$BS color red"

$ns_ at 6.01 "$BS label BS"

$ns_ at 6.39 "$node_(5) color red"

$ns_ at 6.39 "$node_(5) label 5"

$ns_ at 6.39 "$BS color red"
```

```
$ns_ at 6.39 "$BS label BS"

$ns_ at 7.01 "$node_(6) color red"
$ns_ at 7.01 "$node_(6) label 6"
$ns_ at 7.01 "$BS color red"
$ns_ at 7.01 "$BS label BS"

$ns_ at 7.39 "$node_(7) color red"
$ns_ at 7.39 "$node_(7) label 7"
$ns_ at 7.39 "$BS color red"
$ns_ at 7.39 "$BS label BS"

set tcp0 [new Agent/TCP]
set sink0 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp0
$ns_ attach-agent $BS $sink0
$ns_ connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns_ at 4.01 "$ftp0 start"
$ns_ at 4.10 "$ftp0 stop"
$ns_ at 4.15 "$node_(0) color blue"
set tcp1 [new Agent/TCP]
```

```
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(1) $tcp1
$ns_ attach-agent $BS $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at 4.39 "$ftp1 start"
$ns_ at 4.48 "$ftp1 stop"
$ns_ at 4.54 "$node_(1) color blue"

set tcp2 [new Agent/TCP]
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $tcp2
$ns_ attach-agent $BS $sink2
$ns_ connect $tcp2 $sink2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns_ at 5.01 "$ftp2 start"
$ns_ at 5.10 "$ftp2 stop"
$ns_ at 5.15 "$node_(2) color blue"

set tcp3 [new Agent/TCP]
```

```
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(3) $tcp3
$ns_ attach-agent $BS $sink3
$ns_ connect $tcp3 $sink3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ns_ at 5.39 "$ftp3 start"
$ns_ at 5.48 "$ftp3 stop"
$ns_ at 5.54 "$node_(3) color blue"

set tcp4 [new Agent/TCP]
set sink4 [new Agent/TCPSink]
$ns_ attach-agent $node_(4) $tcp4
$ns_ attach-agent $BS $sink4
$ns_ connect $tcp4 $sink4

set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ns_ at 6.01 "$ftp4 start"
$ns_ at 6.10 "$ftp4 stop"
$ns_ at 6.15 "$node_(4) color blue"
$ns_ at 4.78 "$node_(8) color red"
```

```
$ns_ at 4.78 "$node_(8) label 8"

$ns_ at 4.78 "$BS color red"

$ns_ at 4.78 "$BS label BS"


set tcp8 [new Agent/TCP]

set sink8 [new Agent/TCPSink]

$ns_ attach-agent $node_(8) $tcp8

$ns_ attach-agent $BS $sink8

$ns_ connect $tcp8 $sink8

set ftp8 [new Application/FTP]

$ftp8 attach-agent $tcp8

$ns_ at 4.78 "$ftp8 start"

$ns_ at 4.90 "$ftp8 stop"

$ns_ at 4.96 "$node_(8) color blue"

#Throughput comparison graph

proc throughput {} {

global ns_ sink0 sink1 sink2 sink3 sink4 sink5 sink6 sink7 g1 g1 g1 g1 g1 g1 g1
g1

global ns_ sink8 sink9 sink10 sink11 g1 g1 g1 g1

global ns_ sink12 sink13 sink14 sink15 sink16 sink17 g1 g1 g1 g1 g1 g1

global ns_ sink18 sink19 sink20 sink21 sink22 sink23 g1 g1 g1 g1 g1 g1

#Get the number of bytes that have been received by the traffic sinks
```

```
set rb0 [$sink0 set bytes_]
set rb1 [$sink1 set bytes_]
set rb2 [$sink2 set bytes_]
set rb3 [$sink3 set bytes_]
set rb4 [$sink4 set bytes_]
set rb5 [$sink5 set bytes_]
set rb6 [$sink6 set bytes_]
set rb7 [$sink7 set bytes_]
set rb8 [$sink8 set bytes_]
set rb9 [$sink9 set bytes_]
set rb10 [$sink10 set bytes_]
set rb11 [$sink11 set bytes_]
set rb12 [$sink12 set bytes_]
set rb13 [$sink13 set bytes_]
set rb14 [$sink14 set bytes_]
set rb15 [$sink15 set bytes_]
set rb16 [$sink16 set bytes_]
set rb17 [$sink17 set bytes_]
set rb18 [$sink18 set bytes_]
set rb19 [$sink19 set bytes_]
set rb20 [$sink20 set bytes_]
```

```
set rb21 [$sink21 set bytes_]

set rb22 [$sink22 set bytes_]

set rb23 [$sink23 set bytes_]

#Get the current time

set now [$ns_ now]

#Calculate the throughput in (MBit/s) and write it to the files

puts $g1 "0 [expr $rb0/$now*8/1000000]"

puts $g1 "1 [expr $rb1/$now*8/1000000]"

puts $g1 "2 [expr $rb2/$now*8/1000000]"

puts $g1 "3 [expr $rb3/$now*8/1000000]"

puts $g1 "4 [expr $rb4/$now*8/1000000]"

puts $g1 "5 [expr $rb5/$now*8/1000000]"

puts $g1 "6 [expr $rb6/$now*8/1000000]"

puts $g1 "7 [expr $rb7/$now*8/1000000]"

puts $g1 "8 [expr $rb8/$now*8/1000000]"

puts $g1 "9 [expr $rb9/$now*8/1000000]"

puts $g1 "10 [expr $rb10/$now*8/1000000]"

puts $g1 "11 [expr $rb11/$now*8/1000000]"

puts $g1 "12 [expr $rb12/$now*8/1000000]"

puts $g1 "13 [expr $rb13/$now*8/1000000]"

puts $g1 "14 [expr $rb14/$now*8/1000000]"
```

```
puts $g1 "15 [expr $rb15/$now*8/1000000]"
puts $g1 "16 [expr $rb16/$now*8/1000000]"
puts $g1 "17 [expr $rb17/$now*8/1000000]"
puts $g1 "18 [expr $rb18/$now*8/1000000]"
puts $g1 "19 [expr $rb19/$now*8/1000000]"
puts $g1 "20 [expr $rb20/$now*8/1000000]"
puts $g1 "21 [expr $rb21/$now*8/1000000]"
puts $g1 "22 [expr $rb22/$now*8/1000000]"
puts $g1 "23 [expr $rb23/$now*8/1000000]"
#Reset the bytes_ values on the traffic sinks
$sink0 set bytes_ 0
$sink1 set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0
$sink4 set bytes_ 0
$sink5 set bytes_ 0
$sink6 set bytes_ 0
$sink7 set bytes_ 0
$sink8 set bytes_ 0
$sink9 set bytes_ 0
$sink10 set bytes_ 0
```

```
$sink11 set bytes_ 0

$sink12 set bytes_ 0

$sink13 set bytes_ 0

$sink14 set bytes_ 0

$sink15 set bytes_ 0

$sink16 set bytes_ 0

$sink17 set bytes_ 0

$sink18 set bytes_ 0

$sink19 set bytes_ 0

$sink20 set bytes_ 0

$sink21 set bytes_ 0

$sink22 set bytes_ 0

$sink23 set bytes_ 0
}

$ns_ at 5.86 "throughput"

# Tell all nodes when the simulation ends

$ns_ at 5.85 "$BS color pink"

$ns_ at 5.86 "stop"

$ns_ at 5.87 "puts \"NS EXISTING...\";$ns_ halt"

proc stop {} {

  global ns_ tracefd g1

  $ns_ flush-trace
```

```
    close $tracefd
    close $g1
    puts "running nam..."
    exec nam wireless-node-out &
    exec xgraph withsch.tr withoutsch.tr -geometry 800x500 &
    exec xgraph withene.tr withoutene.tr -geometry 800x500 &
}

puts "Starting Simulation..."
$ns_ run
```

SCREEN SHOTS

APPENDIX 2

SCREEN SHOTS

NODES BEFORE TRANSMITTING

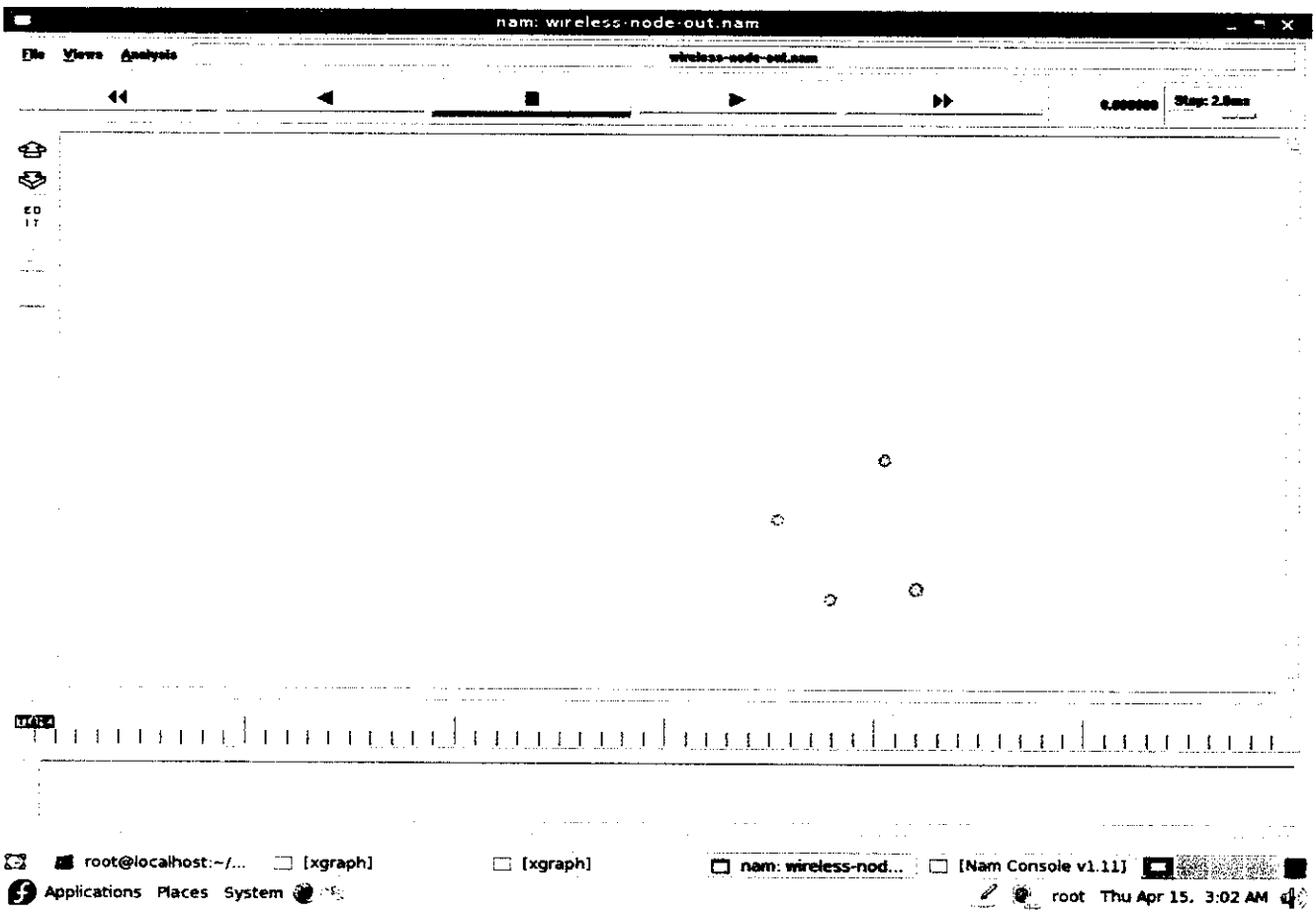

FIG 7.1 NODES BEFORE TRANSMITTING

TRANSMISSION OF NODE IN RAINY REGION

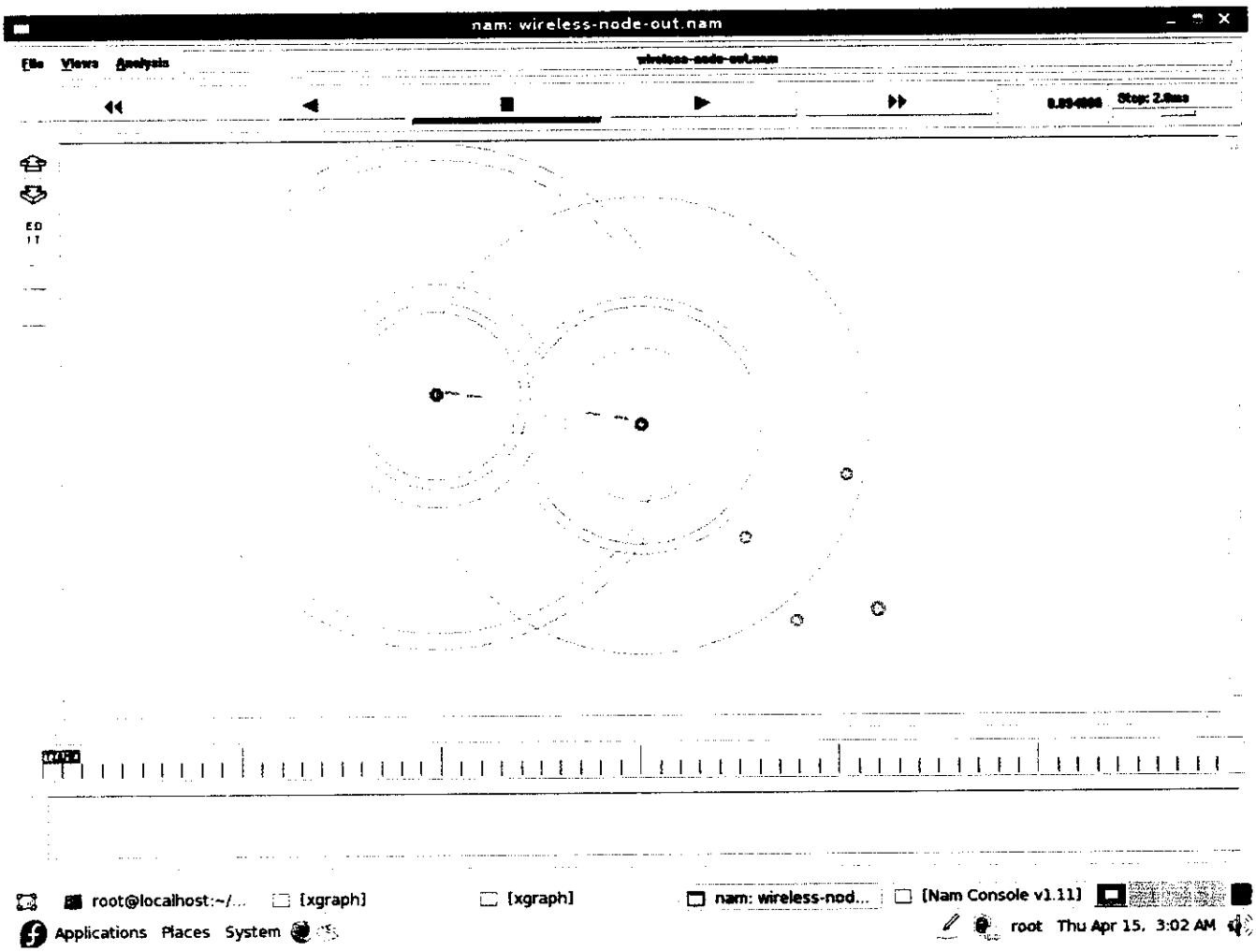

FIG 7.2 TRANSMISSION OF NODE IN RAINY REGION

TRANSMISSION OF A NODE IN DRY REGION

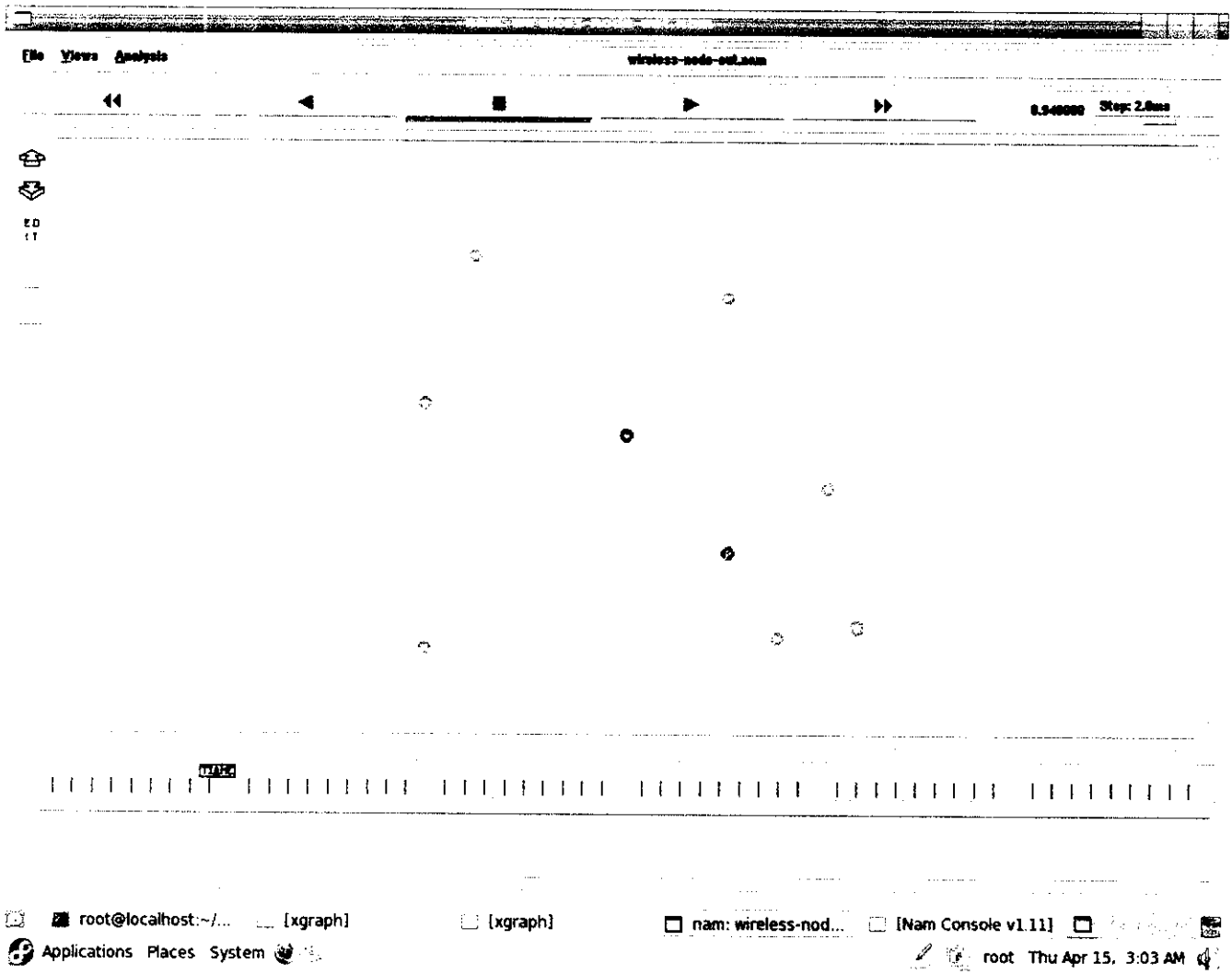

FIG 7.3 TRANSMISSION OF A NODE IN DRY REGION

TRANSMISSION OF A NODE IN MODERATE REGION

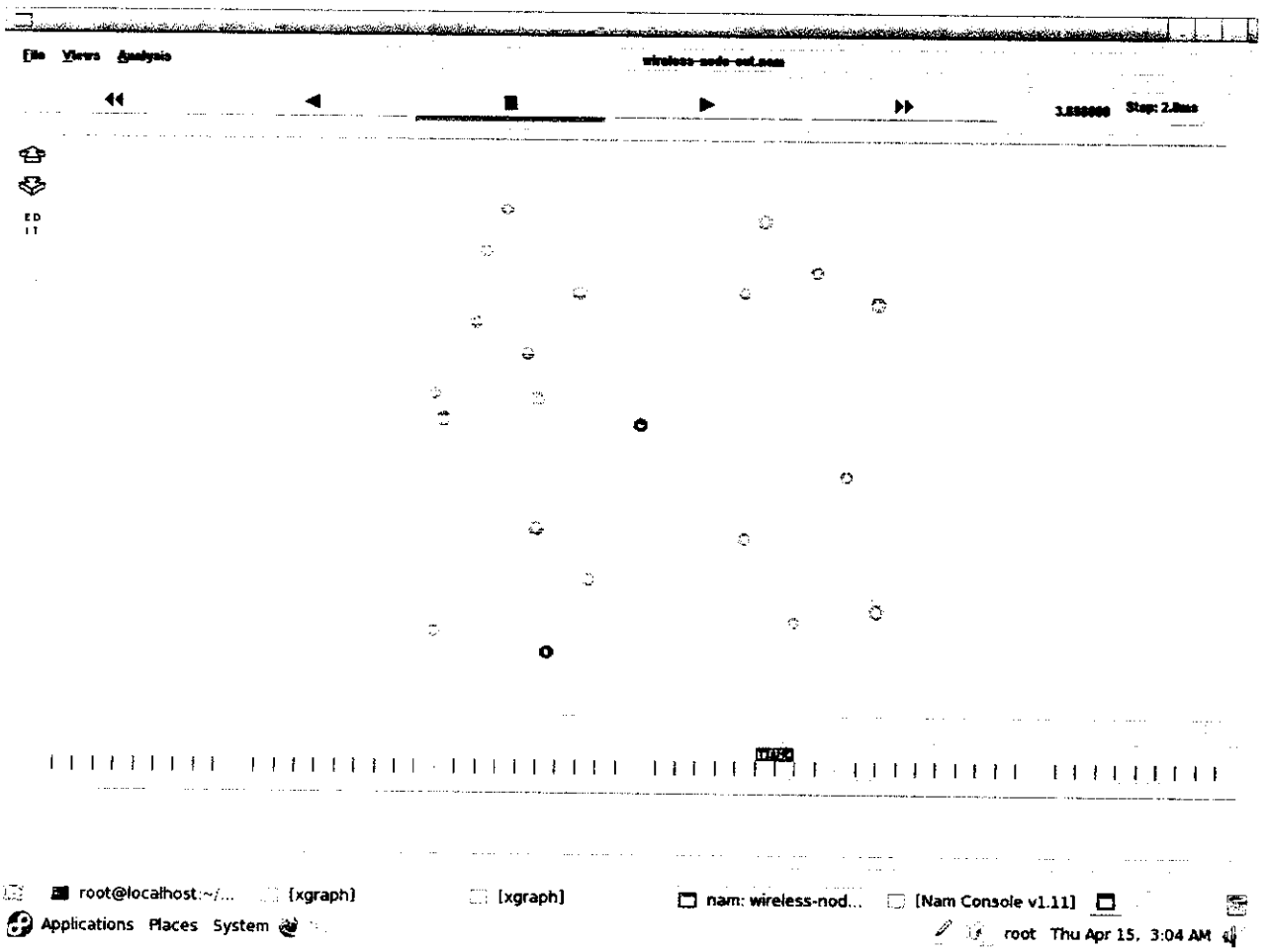

FIG 7.4 TRANSMISSION OF A NODE IN MODERATE REGION

TRANSMISSION OF A NODE IN MODERATE REGION

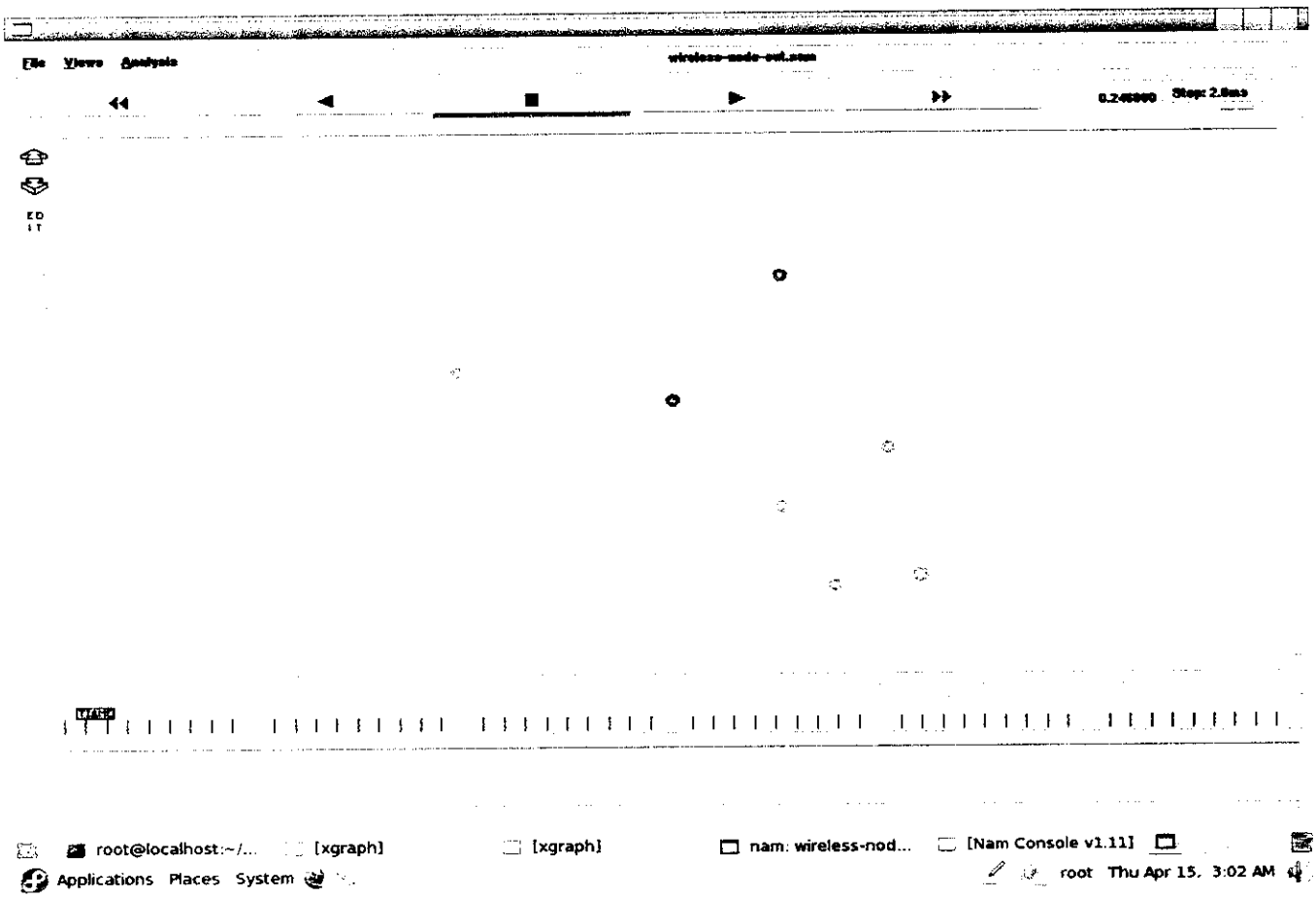

FIG 7.5 TRANSMISSION OF A NODE IN MODERATE REGION

NODES AFTER TRANSMISSION

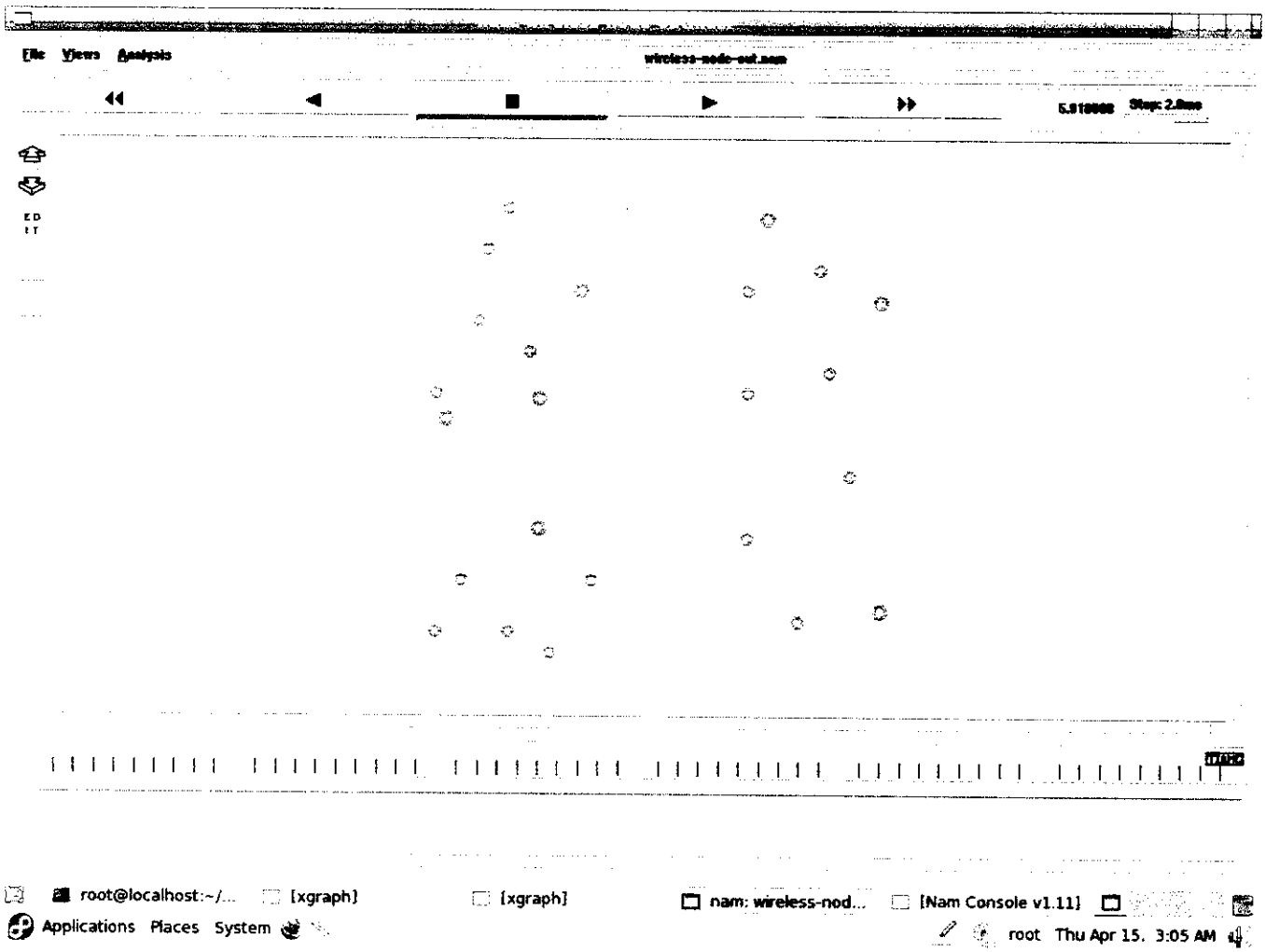

FIG 7.6 NODES AFTER TRANSMISSION

REFERENCES

REFERENCES

1. Keith Smettem, Kevin Mayer, Mark Kranz and Rachel Cardell-Oliver (2004) 'Field testing a wireless sensor network for reactive environmental monitoring' - International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne.

2. Antonio Gonzalez, Ian W Marshall, and Lionel Sacks (2004) 'A self-synchronised scheme for automated communication in wireless sensor networks' - International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne.

3. Alan Mainwaring, David Culler, John Anderson, Joseph Polastre and Robert Szewczyk (2004) 'An analysis of a large scale habitat monitoring application' - Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pages 214–226. ACM Press.

4. Chris Roadknight, Ian W Marshall, Laura Parrott and Nathan Boyd (2004) 'A layered approach to in situ data management on a wireless sensor network. In International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne.

5. ns-2, http://www.isi.edu/nsnam/ns/