Volume 5, Issue 10, October-2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

WSN Energy Efficient Mote Transfer for Voice Applications

Vivek Dubey*, Rajkumar Jarial
Department of Electrical Engineering,
NIT Hamirpur, Himachal Pradesh, India

Abstract— A wireless sensor network is a very major area of research due to the recent advances in MEMS technology, wireless communications and digital electronics. The interest in multimedia applications made it important to address the Quality of Service (QoS) issues related to such applications. It is difficult to deal with the tradeoff between energy efficiency and quality of service (QoS) for multimedia applications in wireless sensor networks. The aggregation scheme is applied to reduce overhead for high throughput and energy saving in wireless sensor networks. However, it may increase delay. In this paper, voice quality evaluation and energy analysis is done to determine the effectiveness of the voice transmission using firefly nodes.

Keywords— Wireless Sensor Network, TelosB, TinyOS, Quality of Service

I. INTRODUCTION

Wireless sensor networks (WSNs) is a field that israpidly increasing in complexity and size due to recent developments in communication technologies, computing and sensing [1]. In most of the possible applications the networks are required to operate autonomously, in environments remote or inaccessible to human reach, and robustly to compensate for node failures or addition of extra nodes. These applications require the sensors to operate optimally because of the resource constraints, energy and bandwidth being the primary. Quality of service can generally be defined as the qualitative and quantitative characteristics of a system that are required to achieve the functionalities expected from the system.

Voice transmission refers to the exchange of audio signals between two nodes. It is one of the multimedia applications that are being focused on in the area of WSN. The factors affecting the multimedia traffic in WSN are:

- *High Bandwidth demand*: The bandwidth required for transmitting multimedia data is very high, which sometimes is hard to satisfy even on wired networks. This makes the bandwidth constraints of WSN seem very large.
- *Multimedia coding techniques*: Raw multimedia data requires a huge amount of bandwidth, of the order of MBps. This amount of bandwidth is not easily available in WSN. So, the raw data has to be processed and compressed into smaller packets, by encoding the data. The raw data is then sent in the encoded form and decoded at the destination.
- *Delay*: Delay is the most important factor of all QoS parameters required for multimedia transmission. For the Telosb motes used, delay is the parameter that has been worked on. Delay is reduced by using scheduling algorithm at the transmitting base station mote, to schedule the packet transmission rate.

II. LITERATURE SURVEY

SadafZahedi, et al. [2], proposed that WSN systems have become applicable in various domains. The scale and resource constraints which are the major problems of sensor networks have resulted in the development of various protocols but their focus was on traditional metrics of *quality of service* of network data transport.

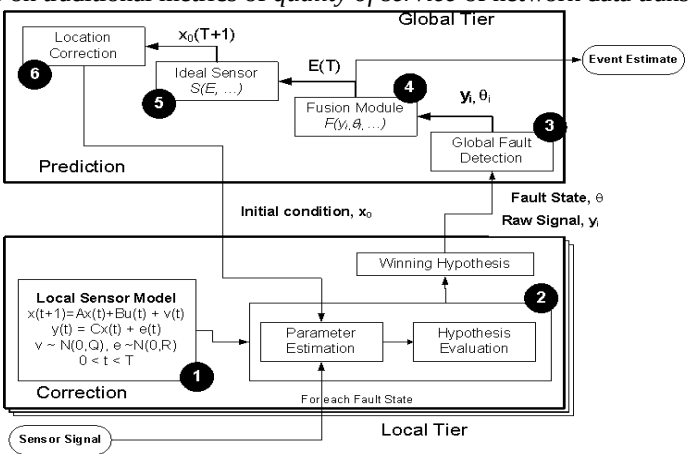

Fig-1: architecture of the two tier fault detection mechanism [2]

In this paper the author explains how the network protocols, data integrity and sensor fusion algorithms are designed to optimize the "Quality of Information "returned by a sensor network. The WSN system should be designed with constraints on cost, bandwidth and energy resources while optimizing performance metrics such as reconstruction fidelity, detection performance, and latency etc. The multidimensional performance metric is called "Quality of Information." Multiple sensors observations are sent to a sink node where all these information is fused and processed at the sink. The QoI not only depends on the properties of sensor fusion algorithms, in addition to it is also dependent on the quality of data received.

The architecture used by the author has several components. The first is the two tired fault detection mechanism. The two tiers in this are the local tier and the global tier. The local tier consists of independent fault detectors embedded at each sensor node that have access to high frequency samples. These sensors respond to the changes in the measurement. The fault detection is done in a twostep process of parameter estimation and hypothesis evaluation. The sensor node sends at a low rate measurement combines with the normal or faulty status to a global tier.

The second component is fault aware fusion module that estimates the occurrence of events of interest. This estimate is fed to an ideal sensor model to predict what the response of each sensor should be to this event. Local tier is fed with the value at each sensor. Thus these two components becomes a faulty detector wherein a faulty sensor is isolated.

The third key component of the architecture is the quality aware data dissemination mechanism that uses a random rerouting algorithm and augments it with the priority bit of packets and controlling their rates according to the decision from the local tier and the relative importance of the node to the QoI.

Tuanli Wang, et al. [3], proposed that sensor networks are like distributed systems with processors, memory and short range wireless communication. So QoS plays a major role in wireless sensor networks. Work within wireless sensor networks Quality of service is an isolated one and research in this area remains largely open. In this paper the author describes the WSN QoS requirements within an application and then analyzes issues for QoS monitoring. WSN basically differ from the traditional networks because these devices have limited capabilities, high resource constraints and are deployed in high densities with in a dynamic and harsh environment. In this paper the author describes the WSN QoS parameters within certain functional layers within a WSN application.

In the reservation based approach the resources are reserved by a particular node and in reservation less approach no reservation is required. QoS is achieved by some strategies like admission control, traffic class, queuing mechanisms etc. QoS challenges in WSN mainly come from the scarce bandwidth and the complexity of user mobility during the last wireless hop. So, the QoS architecture is deployed in WSN with wireless MAC protocols. QoS in wired networks is different from ad-hoc networks because of the bandwidth constraint and dynamic network topology. So in WSN, QoS needs to be applied in a dynamic environment with a very limited bandwidth.

WSN when used in critical applications like target trafficking in battle fields require reliable and timely delivery of the sensory data. The challenges like resource constraints, unbalanced mixture traffic, data redundancy, network dynamics, and energy balance are some of the common characteristics shared by the sensor applications. So, QoS support for the WSN must take at least few of the previously mentioned challenges into account when an application is specified.

The QoS requirements for the application layer are specified by the users of the network. The QoS requirements for WSN applications layer are system lifetime, response time, data novelty, detection probability, data reliability and data resolution. System life time is the total lifetime of the sensor node, response time is the time taken to get the response of a user query. Data novelty is the time taken for the data of an event to arrive at the sink from the time the event is detected. Data reliability and data resolution defines on the data quality. Detection probability is the probability that a real world situation can be detected and reported to the user. The QoS requirements for the transport layer are defined as reliability, bandwidth, latency and cost.

All of the QoS requirements of the transport layer use the collective concept that is only the unique packets are counted as received at the destination. The QoS requirements for the network layer are path latency, routing maintenance, congestion probability, routing robustness and energy efficiency. The QoS requirements for the connectivity maintenance layer are network diameter; average path cost, connectivity robustness, and connectivity maintenance whereas the QoS requirements for coverage maintenance layer as proposed by the author in this paper are coverage percentage, coverage reliability, coverage robustness and coverage maintenance. Communication range, throughput, transmission reliability and energy efficiency are the QoS requirements for the MAC layer. The physical layer QoS encompasses wireless unit capabilities, processor capabilities and sensing capabilities.

HakanOztarak, et al. [4], in their paper stated that in order to study the extracted information from the environment multimedia data is playing a crucial role. Due to the unique properties of multimedia delivery it has become very costly and is facing the novel challenges of the wireless sensors that is resource constrained, requires a high bandwidth. Processing limitations prohibit the use of local processing that is processing at the individual nodes. In this automatically extracted moving objects are treated as events and joint processing of this collected data at the sink is done using the fuzzy membership to identify the actual data to be sent. In multimedia sensor networks the data that has to be communicated to the sink becomes a major research challenge because these nodes are resource constrained prohibiting lossless transmission. So, to decrease this processing and storage cost a framework is developed for both events triggered observations and streaming multimedia data which is generated over longer time periods. In this framework data is processed at the sensor nodes and at the sink node it is compiled to produce fuzzy object and event definitions. High data reduction rate low complexity and expressiveness of produced analysis are the major objectives to be achieved in multimedia processing at sensor nodes.

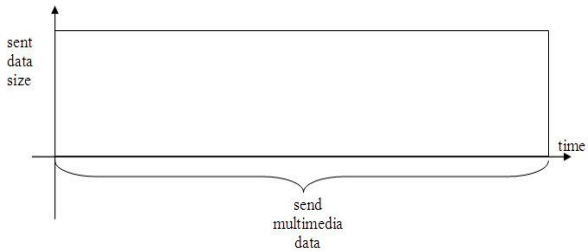

Fig-2: Communication overhead of the traditional wireless multimedia sensor networks [4]

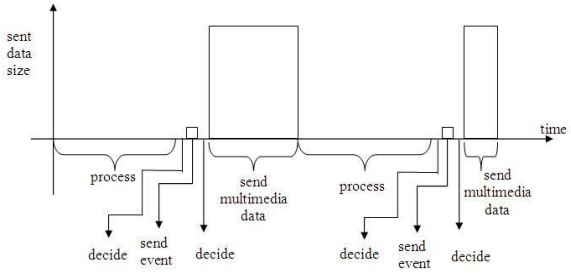

Fig-3: Communication overhead of the proposed architecture [4]

For detecting the moving objects at sensors the author has used Hierarchical parzen Window Based Moving object Detection method which has two stages in it. The received data is jointly fuzzy processed using Micro SEBM. Specific events that occur together in a particular interval of time is called a video event which is defined in Micro SEBM model with VE={ID, Time, MBR,F}.Once the event information is received from the relevant sensor nodes. The sink node stores the events in a multimedia data base. Since the data received are a sequence of rectangles it can be processed easily. Only regional and trajectory queries are investigated for the WSNapplications and Micro-SEBM.

Ranjitiyer, et al. [5], proposed an amalgamation of QoS feedback and sensor networks. In this the base station is allowed to communicate to the sensors using a broadcast channel and Gur game is used as a mathematical paradigm to dynamically adjust to the optimum number of sensors. Sensor networks are robust because they continue to provide information in spite of the failure of the individual sensors. In this paper sensor network quality is explored because sensor deaths and sensor replenishments make it difficult to control the number of sensors sending information at a particular time. The result will be a robust sensor network that allows base station to activate the number of sensors at a particular time which controls thereby the resolution of QoS it receives from the sensors. In this paper the author describes QoS as sensor network resolution. The bases stations or sinks need some information from different sensors but sensors which are in close proximity to each other allow many of the sensors to be powered down.

In this paper algorithms to address two goals are proposed. The algorithm tells that there are a collection of m sensors from S_1 to S_m and one base station and the total time is divided into intervals each of one second duration. Each sensor S_i is at some distance d_i from the base station. Here, 'i' is assumed that the packet from the sensor node to the base station is delayed and from the base station to the sensor node is immediately received. Each sensor node is related with a finite state automation M. The sensor will power up when it reaches a positive numbered state as in a Gur game. There is a tradeoff between higher quality encoding or low quality audio with redundant packets.The mining scenario consists of five tasks:a sensing task,audio task,localization task,network management task,and link layer task.For managing individual link TDMA communication link layer is responsible.The goal of voice scheduling is to minimize the number of unique slots to maximize n. Scheduling is done in two ways one is to fix or reserve the time slots and the other is reserveless allocation.The first step in audio streaming schedule is to form a spanning tree across the network with root as located at the gateway.Each node is indicated with an arrow which indicates the direction of flow and the number of slots of latency associated with the next hop.Both voice quality evalutaion and energy analysis is done to determine the effetiveness of the voice transmission using firefly nodes.

III. NETWORK MODEL

The network model consists of three nodes (Telosb motes), each having the same capabilities, i.e., we consider a homogeneous sensor network. Two of the nodes are connected to two computers, as base-stations at a distance from each other. The other node works as an intermediate node for the two base stations. The network topology is created by working on neighbor detection.

Neighborhood of a node is defined as the set of nodes that are within the vicinity of the node, i.e., a node (S_j)is said to be the neighbor of a node (S_i), or S_j is in the neighborhood of S_i, if S_j falls within the sensing range of the node S_i, which is pre-defined for any specific sensor node. This algorithm for detecting neighbors is based on the range of the sensor nodes, and it is used, in Octopus, to find the neighbors of a node.

IV. STARTUP WORK

The start up work was to install TinyOS (XUbuntu) on the system and start working with the Telosb motes. The working with Telosb motes is as follows:

To begin with working on the Telosb motes on TinyOS, we need to build these motes on the system. The Telosb motes, once connected to the system through the USB ports, register themselves with the system automatically.

1. To get a list of motes connected to the system, the ***motelist*** command is used as follows:

$*motelist*

The above command gives us the reference of the mote (unique id), along with the device (the device/port to which the mote is connected) and its description

2. To build the motes, we use the ***make*** command as follows: **$***make*telosb

The ***make*** command makes the mote ready to install any application on it.

3. The applications, which can be run on the motes in TinyOS, have specific conventions as follows:

Interfaces file of the form '*App.nc*', containing the non-standard interfaces that the application requires.

A header file of the form '*App.h*', containing any non-standard structures or variables required for the application.

Public modules file of the form '*AppC.nc*', containing the modules with functionalities of the application.

Private modules file of the form '*AppP.nc*', containing private modules that would only be required by the current application and by no other application.

One 'Makefile' file, which contains parameters for building the application.

4. To test the mote that have been built on to the system, there are sample applications that come packed with TinyOS. 'Blink', one of such applications, has been used to test the motes. The 'Blink' application resides at the location "/opt/tinyos-2.x/apps/Blink". The 'Blink' application is a basic application that makes the 3 LED's (LED0, LED1, LED2), with different frequencies. This application is installed into the Telosb mote as follows:

$ *make* telosb*install*,1

On successful installation of the application, the mote is assigned id '1', and the LED's on the mote start blinking with different frequencies (LED0 with frequency 0.25Hz, LED1 with frequency 0.5Hz, and the LED2 with frequency 1Hz).

5. After testing the mote, we developed a small sample application for the mote to mote communication 'BlinkToRadio'. The application consists of a configuration file '*BlinkToRadioC.nc*', a header file '*BlinkToRadio.h'*, an application file '*BlinkToRadioAppC.nc*', and the '*Makefile*'. After developing this application, this was tested on the motes for radio communication, and was successful. This application sends packets from one telosb mote, to another telosb mote (both of the motes had the '*BlinkToRadio*' application installed using the *make* and *install* commands as mentioned before and on of them was connected to the computer, while the other was running on batteries).

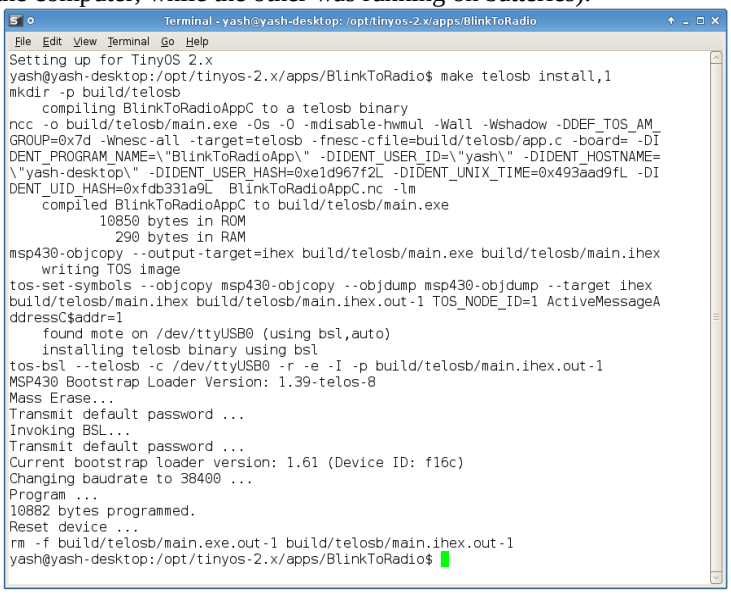

Fig-4: Installation of BlinkToRadio application on the telosb mote

6. To check the contents of the packets, an inbuilt java application 'listen', is used, which sniffs the packets arriving through the radio medium, and displays the contents of the packets in raw format. The packets are of the form

- Destination address (2 bytes)
- Link source address (2 bytes)
- Message length (1 byte)
- Group Id (1 byte)
- Active message handler type (1 byte)
- Source mote ID (2 bytes)
- Counter value (2 bytes)

7. The '*serial forwarder*' application has been used to check the traffic arriving at the mote connected to the computer. It displays the number of packets read by and written by the mote connected to the computer, and the number of clients that are connected to the current mote, as follows:

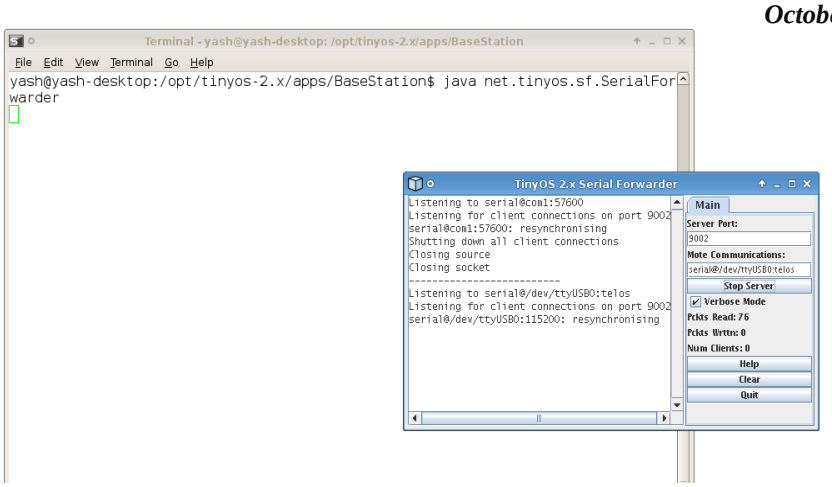

Fig-5: Snapshot of Serial Forwarder

V. IMPLEMENTATION

After the basic working of TinyOS and motes had been understood and experimented on, the next part of the work was the major contribution to the project. This part of the project is done on the Octopus, a middleware that runs above the TinyOS platform. The Octopus provides a platform with a visual representation of the sensor network, with the interface to change certain parameters, like sleep-duty cycle, active-duty cycle, sampling period, etc., which have an impact on the connectivity of the motes in the wireless sensor networks.

1. Creating the base stations:

Two of the telosb motes are made base stations, one for transmitting and other for receiving. The base station application is installed on both of these motes by using the '*make*' and '*install*' commands, and the result is as follows:

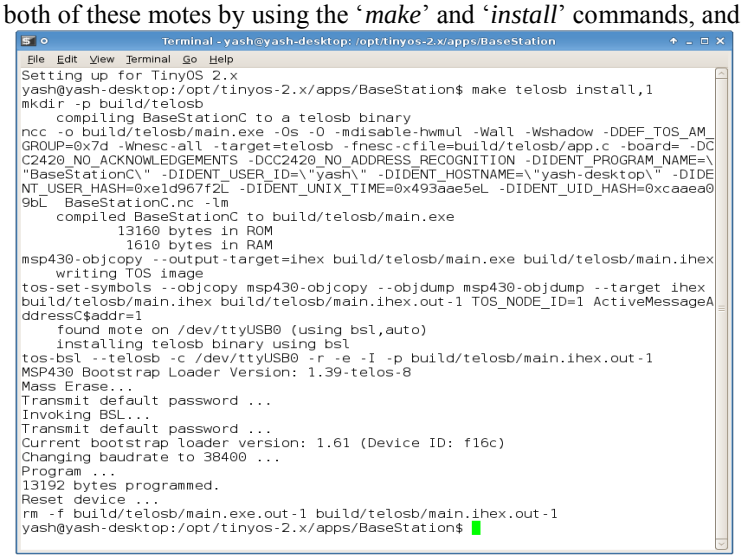

Fig-6: Installation of Base Station on the telosb mote

2. Installing Octopus:

Octopus application has been installed on TinyOS platform, with all of its features. To migrate it into other systems, it is sufficient to copy the dump of Octopus from the current system to the others.

3. Installing Octopus on to the telosb motes:

After installing Octopus on TinyOS, for it to visually represent the network, it has to be installed on the motes as well. The Octopus application contains the following directories:

- Java – contains the java files that form the visual component of Octopus
- Motes – contains the configuration files corresponding to different families of motes that are compatible with Octopus
- Doc – contains the documentation of the application
- Src – contains the source code of Octopus

In order to install Octopus on the telosb motes, the motes have to be connected to the computer one by one, and built using the '*make*' command. After the make command configures the motes to be able to install Octopus on them, the same is installed using the '*make*' and '*install*' commands as follows:

Browse into the motes directory of Octopus and type the following command:

$ *make*telosb

This command, when executed successfully, creates a directory '*build*' in the Octopus directory, with a subdirectory '*telosb*' in it. This directory contains a '*main.exe*' file if the build was successful, and if it is not present, we have to rebuild the mote by using the above stated command.

If the build is successful, we have to choose one mote out of the three to be the root mote for the network. This mote is then connected to the computer and Octopus is installed into it as follows:

In the motes directory in Octopus, type the following command:

$ *make*telosb*reinstall*,0

If the command executes successfully, the mote is equipped with Octopus software, and is assigned the id 0 (id 0 for a mote makes it the root mote for Octopus). Then, we install Octopus on the other motes, giving each of them their ids, which can be any integer.Next, we have to configure Octopus to display the network topology, which is achieved as follows:

Browse into the java subdirectory of Octopus and execute the following commands in order:

$ make
$ export MOTECOM=serial@/dev/ttyUSB0:telos
$ javanet.tinyos.sf.SerialForwarder
$ javaOctopusGui

The first command (*make*), compiles and builds all the java files in the Java subdirectory of Octopus, for building the visual interface of Octopus. The next command (*export*) sets the "MOTECOM" variable to the mote connected to the computer. Then, we switch on the serial forwarder application to check on the traffic through the sensor nodes, and the last command (java OctopusGui), runs the GUI of Octopus, which shows the topology of the WSNand the connectivity across different motes in the network. The GUI of Octopus presented as follows

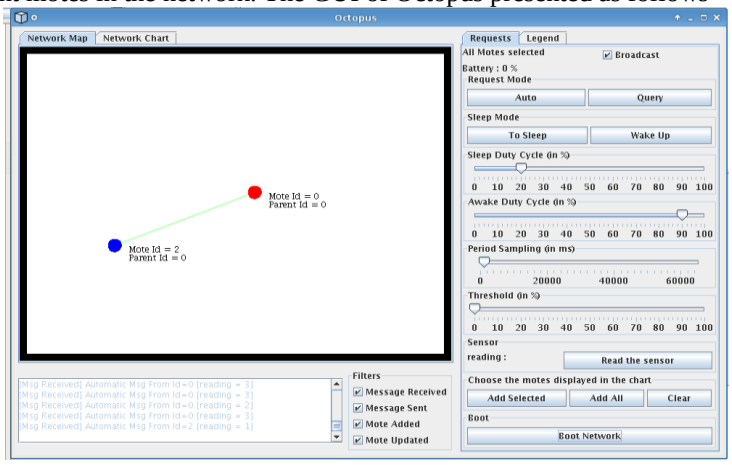

Fig-7: GUI of Octopus

The network of sensor motes has been developed and is ready for data transfer. Since voice transmission requires buffers at both ends of the system, i.e., at the initial transmitter and the final receiver, we built the network with one mote at each of the two computers and used one intermediate mote. This network has enough buffers at both ends, in the form of storage devices in computers, and one intermediate mote for passing on the packets from one mote (on one computer) to the other (on the other computer). This network can be expanded to make the distance between the sensor motes large and introduce many intermediate motes.

VI. RESULT

Voice Transmission requires a lot of hardware capabilities, like processing power, and memory, which are not sufficiently available on Telosb motes. So the main part of communicating from one computer to another computer through wireless sensors has been achieved. This can be used to achieve voice transmission, with little difficulty, by acquiring sensor motes with higher capabilities, like the "Firefly" motes, and adjusting the packets to carry encoded voice, instead of raw data.

Thus, the connectivity between computers through wireless sensors has been successfully achieved and the network is ready for utilization to transmit voice.

Some of the snapshots of the terminals on executing the commands and installing the applications are given below:

Fig-8: Telosb mote installed with BlinkToRadio application

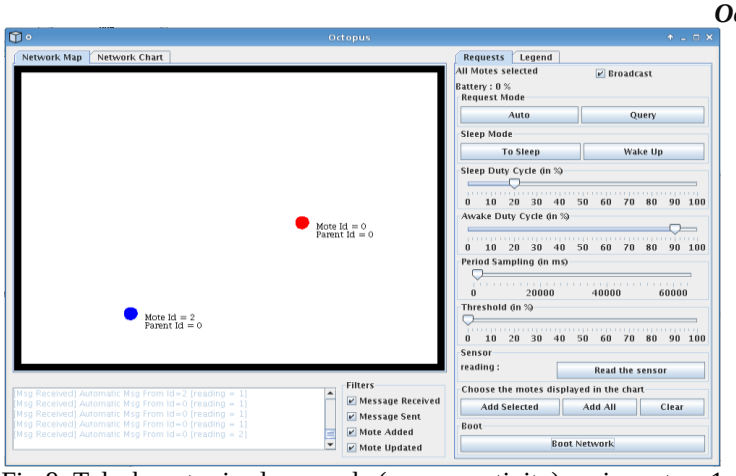

Fig-9: Telosb motes in sleep mode (no connectivity), as in system 1

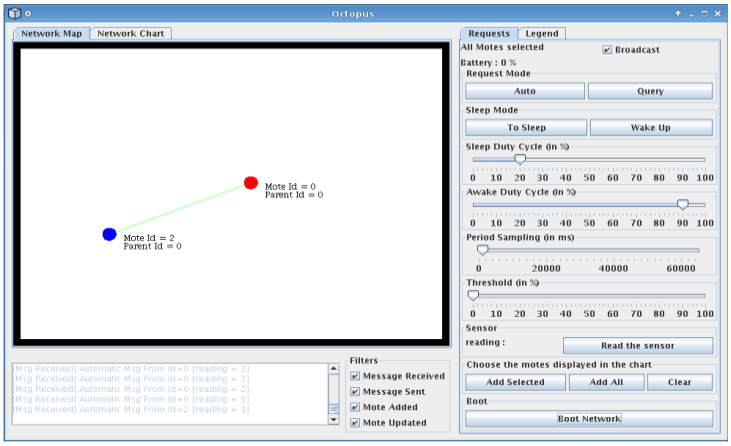

Fig-10: Telosb motes in active mode (full connectivity), as in system 1

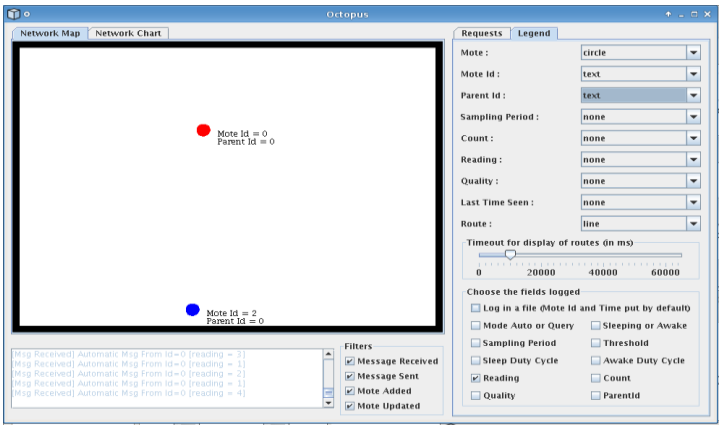

Fig-11: Telosb motes in sleep mode (no connectivity), as in system 2

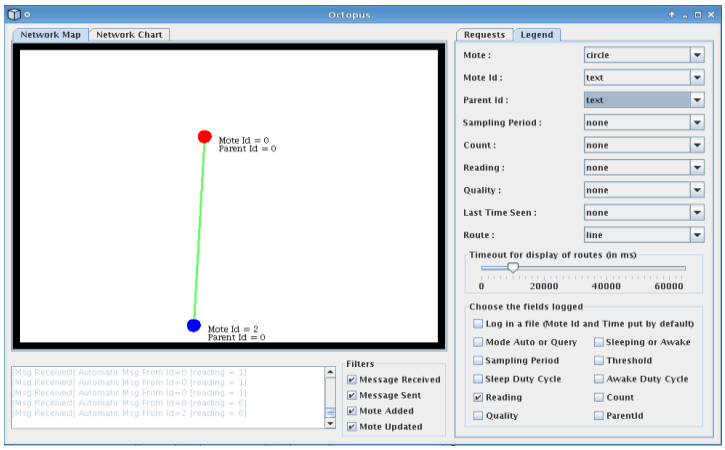

Fig-12: Telosb motes in active mode (full connectivity), as in system 2

VII. CONCLUSION

This paper deals with the implementation of voice transmission using firefly nodes, which has successfully achieved. The experiments conducted suggests that the experimental setup can be used efficiently for the purpose of voice transmission. This research also gives a model for voice transmission that has high throughput and reduced overheads

REFERENCES

[1] James Kay and Jeff Frolik, "Quality of Service Analysis and Control for Wireless Sensor Networks", *IEEE,* 2004.

[2] SadafZahedi (UCLA), Edith Ngai (Imperial College), ErolGelenbe (Imperial College) DinkarMylaraswamy and Mani Srivastava, "Information Quality Aware Sensor Network Services".

[3] Tuanli Wang, Xianghui Liu, Jianping Yin, "Requirements of Quality of Service in Wireless Sensor Networks", *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, IEEE,* 2006.

[4] HakanOztarak, Adnan Yazici, DemetAksoy and Roy George, "Multimedia Processing in Wireless Sensor Networks", *IEEE,* 2008.

[5] Ranjitiyer and Leonard Kleinrock, "QoS Control for Sensor Networks", *IEEE,* 2003.

[6] Rahul Mangharam, Anthony Rowe, Raj Rajkumar, Ryohei Suzuki, "Voice over Sensor Networks", *Proceedings of the 27th IEEE International Real-Time Systems Symposium, IEEE,* 2006.

[7] NavrajChohan and Wei Zhang, "Voice Streaming over a Sensor Network"

[8] Affan A. Syed, Muhammad Omar Khan, John Heidemann, Jack Wills, and Wei Ye, "A Sensornet-inspired Underwater Acoustic Modem for Wake-up and Data", *ACM WUWnet,* September 2008.

