Test bed for wireless sensor networks using XMesh networking protocol

Conference Paper · August 2013

DOI: 10.1109/ICACCI.2013.6637403

CITATION
1

READS
228

2 authors, including:

Ravi Kishore Kodali
National Institute of Technology, Warangal
115 PUBLICATIONS **2,469** CITATIONS

SEE PROFILE

Test bed for Wireless sensor networks using XMesh networking protocol

Ravi Kishore Kodali and Narasimha Sarma, NVS
Department of Electronics and Communications Engineering
National Institute of Technology, Warangal
Andhra Pradesh, 506004, India

***Abstract*—The advent of wireless networking, sensor and embedded system technology has given rise to Wireless Sensor Networks (WSN). In WSN, tiny sensor motes manifest the concept of ubiquitous computing by collaborating with each other. Extraction of the information about various events in the field from the data collected by these motes is a crucial stage in any end-to-end WSN solution. Furthermore, the extracted information is required to be stored in local or remote databases and should be retrieved as per the end user requirements. In this paper, a laboratory set-up for WSN with *XMesh* networking protocol is discussed and the results of experiments carried using WSN hardware are presented. This work also provides an insight into the *XMesh* protocol and is useful during the setting up of a WSN laboratory.**

Keywords: WSN, Sensor motes, mesh topology, multi-hop routing

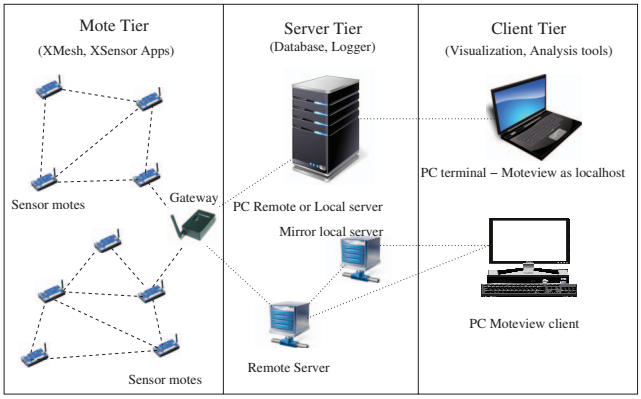

Fig. 1. Software framework of XMesh protocol

I. Introduction

Tracking and monitoring of events in remote and hazardous areas is a challenging task. WSN technology strives to solve this problem by self sustainable and scalable network design approach [1][2]. A WSN consists of many resource constrained sensor nodes, which are deployed randomly in the field. These nodes are battery operated and work without any human intervention after the deployment, battery maintenance becomes a challenging task. This poses energy constraint on the network design. The designer should consider this energy constraint during the development of various network protocols and reduce energy and computational overheads.

One of the main duties of a sensor node is to gather some physical phenomena related data near its vicinity. It is the wireless communication module embedded on the sensor node making it *smart*. By using a wireless transceiver, the node collaborates with various neighbouring sensor nodes to support ubiquitous computing. The nodes can be arranged in a flat or hierarchical topology. In an hierarchical topology, the nodes are grouped into many clusters and cluster heads are elected for each cluster. A cluster head aggregates the information within the cluster and forward it to a higher authority, termed as the base station (BS). The BS monitors the data flow in the network and an authorised user can access data collected by the WSN through its BS.[3]

The *XMesh* networking protocol [3],[4] is analysed in the present work. The mote layer lies on the tiny sensor motes and the software part is written in nesC language [5], a dialect of C. The sensor mote used for the implementation consists of an 8-bit ATmega128 micro-controller operating at 8 MHz frequency. The *XMesh* protocol is used for communication among the nodes, which are randomly deployed in the field. The protocol supports addition and removal of nodes from the network at any point of time, providing a scalable WSN solution. The data analysis is carried out by a user on a computer connected to gateway device of the network. The gateway device, MIB520, which is connected to a computer. The xserve, an interfacing layer, provides software drivers required for representing the WSN data on a computer. The Moteview platform is a client level application providing user friendly GUI. The rest of the paper is arranged as follows: Section II explains the *XMesh* protocol in detail with its different user configurable features. Sections II, III and IV provide deeper insights of the three tiers in the *XMesh* protocol. Section V gives a step-by-step procedure for setting up of a WSN. In section VI, an experimental analysis of WSN hardware kit is given. Section VII concludes the paper with future perspective.

II. *XMesh* protocol

The *XMesh* is an ad-hoc, multi-hop networking protocol for Wireless sensor networks (WSNs) [6]. It consists of many sensor nodes and a base station (BS). *XMesh* is self-healable and self-sustainable protocol, supporting addition and revocation of nodes at any instant of time [7]. The *XMesh* implements a mesh topology [8], in which motes can route the data and establish end-to-end connection in an efficient

978-1-4673-6217-7/13/$31.00

manner. To decide various routing paths, cost metric (CM) [9] used and the link quality, which is calculated by a receiving mote as given equation (1).

$$CM_{link_quality} = \frac{n_R}{n_E}, \quad (1)$$

where n_R represents the number of messages received and n_E represents the number of expected messages. The number of expected messages is calculated by keeping track of the serial number of the message from its neighbouring mote. By updating the link quality parameter periodically, route from parent to sink node is selected eliminating various bad links. The multi-hop routing capability [10] not only guarantees end-to-end connectivity but also reduces energy cost incurred due to message transmission. The *RouteSelect* interface is used to select the efficient route, while the *RouteControl* interface is used to monitor route quality and alter the route state update interval.

The Xmesh protocol can be used to configure a node to operate in any of the three modes of operation: high power(HP), low power (LP) and extended low power (ELP). In HP and LP modes, all multi-hop mesh networking capabilities are enabled, while in ELP mode motes can not route the data and is connected to the BS using a single hop. The communication latency is LP mode is more, as the on board sensors are kept in *SLEEP* mode for most of the time. Also, in low power mode the status indication LED's are turned off. The operation mode can be selected at the time of programming a node by passing an appropriate parameter. As an example, to force low power routing on iris motes the following command is used:
make iris route, lp

The Xmesh protocol is implemented over the sensor motes, the gateway and the BS. Application programs written in *nesC* for different types of sensor motes and the BS makes use of interfaces supported by the *XMesh* protocol. The application code varies from a sensor mote type to another one. XMDA100, XMTS30 and XMTS500 have different types of sensors. In this work, various experiments are carried out using the MDA100 sensor board and a brief analysis of the nesC code of the same is presented. In the application program for the MDA100, during the initialization phase, the timer rate is set according to the defined sample rate. Initialization is done under an atomic section. Various time critical operations are written under atomic section, which can not be interrupted and avoids instability caused due to the overwriting of the global variables. Further, when the timer is fired temperature, light and other sensed data are measured. Asynchronous events at ADC data ready are implemented and respective *getData()* function is called when an event occurs. While transmitting the message, along with the data, node id, packet id and parent id are also transmitted. The *RouteControl* interface provides the parent id. The *Send* interface selects the route and transmits the message. The broadcast messages are handled by the *xcommand* interface and for different commands like *SETRATE, SLEEP,WAKEUP*, appropriate actions are taken. The application program for base station forwards the packet for any *XMesh* application and injects xcommand packets into the network. The following command is used to program the gateway as part of its application code on iris platform to use a frequency of 2.41 GHz:
make iris base freq,2.410

The messages exchanged using *XMesh* protocol contain three headers attached to the sensed data values. In order to provide a sensor specific information, the *XSensorHeader* is also added. Further, a *MultihopHeader* containing source and origin addresses is attached for multi-hop messages. Since the *XMesh* protocol uses TinyOS, a *TOSMsgHeader* is added to provide TinyOS packet type and TOS-NODE-ID. The total header size is of 16- bytes. Also, the messages are appended with a CRC value of 2- bytes for error detection. The data payload size and consequently the message size changes from mote to mote. For instance, the data payload in case of MDA100 sensor mote is 16- bytes and the total data message size is of 34- bytes. Figure 2 provides the complete message structure in *XMesh*.

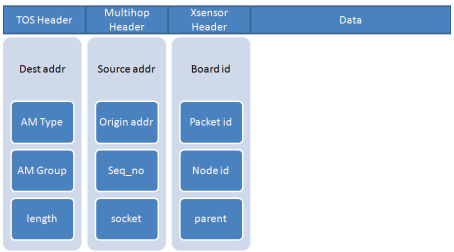

Fig. 2. *XMesh* message structure

III. Mote Tier

Sensor motes, gateway and *XMesh* application program installed on motes and gateway comes under mote tier of *XMesh* protocol. Following subsections discuss different components in Mote tier.

A. Sensor board (MDA100)

In this work, MDA100 sensor board is used to capture environmental data from the surroundings. The MDA100 sensor board contains precision thermistor, light sensor and general prototyping area. All ADC channels of the mote (ADC0-7) can be used for this prototyping area. Other sensor devices also can be hard-wired using this area.

Table I provides various details of the sensor board.

B. Mote (IRIS)

This work uses IRIS motes [11], designed and manufactured by MEMSIC. The IRIS mote has Atmel's AT1281 microcontroller with Atmel's AT86RF230 transceiver based on IEEE 802.15.4 protocol and Zigbee compliant radio [12]. Using the transceiver, the mote can be tuned to any of the 16 frequency

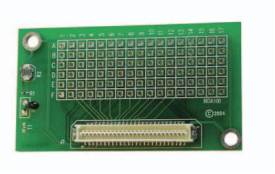

Fig. 3. Sensor board

Fig. 4. IRIS mote

TABLE I
CHARACTERISTICS OF SENSOR BOARD MDA100

Sensor	Description
Thermistor	Accuracy 0.2^oC Base resitance $10K\Omega$ at 25^oC
Light sensor	ON resistance (white light) $10K\Omega$ OFF Resistance $520K\Omega$
Prototyping area	17×6 matrix Connection to USART, ADC, Interrupt and PWM ports of mote

channels, with each separated by 5 MHz ISM band of 2.405 GHz to 2.48 GHz. The communication range of a mote can be set by varying its transmission power in the range of 3 dBm to -17.2 dBm. IRIS mote provides 4 KB RAM and 128 KB flash memory, which is sufficient for most of the WSN applications. Apart from the flash memory available in the micro-controller, the IRIS mote also contains an external 512 KB flash memory to support *On The Air Programming* (OTAP). The mote can be programmed using its air interface.

C. Programming board (MIB520)

A programming board, MIB 520, is used to program IRIS sensor motes using USB ports of a computer. This board consists of an on-board ATMega16 controller. In order to

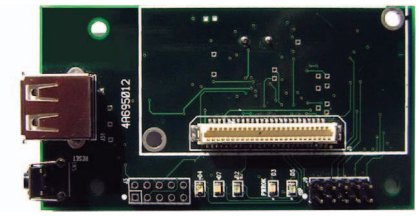

Fig. 5. Programming board (MIB 520)

program various IRIS motes, TinyOS needs to be installed on the computer, being used for this purpose. An image of the TinyOS and the application code is downloaded into the mote through ATMega16 controller. To enable programming using the USB ports, FTDI USB virtual COM port drivers need to be installed in the computer to which the programming board is connected.

IV. SERVER TIER

A reliable interaction between wireless sensor motes and various clients is guaranteed with the assistance of the server tier of the framework. The Xserve is used and this manages database at a remote or local host. Also it handles the presentation of data originating from the WSN to the client software. The serial forwarder service of the XServe allows applications to send and receive raw data directly from the mesh network. Also the Xserve process parses the data during run time to make it presentable form to a client. The command to start the XServe in terminal, to view the data from a WSN connected through the MIB520 gateway using COM port 2, is given as follows:
xserve.exe device=com2

V. CLIENT TIER

The *MoteView*, a GUI platform sourced by *MEMSIC*, provides user friendly data visualization environment. It connects all the three layers of the *XMesh* protocol and provides an end-to-end solution for WSN applications. To begin with a WSN can be connected using *connect to WSN* command. The gateway, database location (remote/local), sensor board type and *Live or History mode* can be selected thereafter. After successful connection, the topology tab in the Moteview GUI shows all the motes and the gateway graphically, with their position and connection status. The current status of the motes can be analysed by using charts, histogram, scatter plot and data tabs, provided in the *Moteview* interface. Various other features like exporting the data to a spreadsheet, programming the mote, calibration of sensors are also available.

VI. LABORATORY SETUP

In this section, a step by step procedure for setting up of a WSN is presented. For this purpose, 4 IRIS motes with 4 MDA100 sensor boards and the fifth mote with a gateway device MIB520, are used. The client layer software is installed on a Windows 7 platform.

A. Programming the sensor mote

The sensor motes can be programmed using *MoteConfig* software which can be accessed from MoteView→Tools→Program mote. One mote is programmed as the gateway, while other motes are loaded *XMesh* sensor application for MDA100 sensor board. A mote to be programmed is attached to the MIB520, whic is connected to one of the USB ports of the computer. In Moteconfig→Settings→Interface board, MIB520 gateway with lower COM port number is then selected. The corresponding Moteconfig GUI is illustarted in Figure 6.

B. Connecting to WSN

After programming each of the motes, two 1.5 V, AA size batteries are inserted in each mote, and the motes are deployed randomly. The WSN is connected using *Connect to WSN* option in the Moteview. The gateway with a higher COM port number and MDA100 sensor board are selected from the *connect to WSN*options as shown in Figure 7.

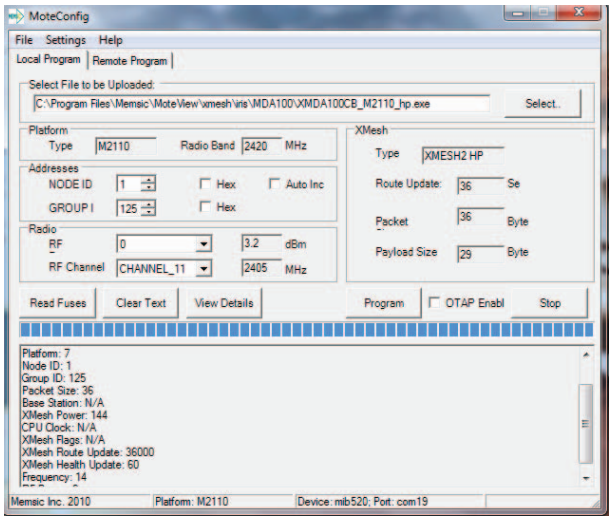

Fig. 6. Programming sensor mote using Moteconfig

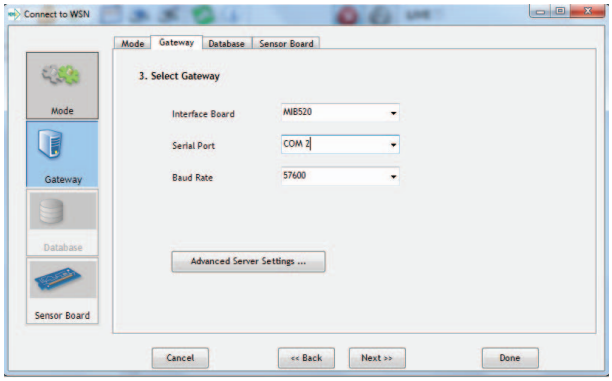

Fig. 7. Connecting to wireless sensor network

C. Monitoring WSN in Moteview

After configuring the WSN connection options, the Xserve starts and presents the WSN state in the Moteview GUI. All the connected nodes and their physical parameters sensed by them are monitored in the Moteview. The topology of the network is as shown in Figure 8. On the left side a list of all the current nodes can be seen and on right side the current topology and the present state of the selected node can be noticed.

VII. Analysis of WSN data in Moteview

The *Moteview* with *Xserve* allows a user to analyse the data collected by the sensor motes. In this section, an experimental analysis using WSN hardware is presented.

A. Multi-hop routing

The *XMesh* protocol supports multi-hop routing to conserve energy of the nodes and provides a guaranteed connectivity among sensor motes and the gateway. To test the multi-hop routing feature of the *XMesh* protocol, initially all the nodes are deployed nearer to the gateway. Then the node 4 is moved away from the gateway ($> 30m$ in indoor conditions). Figure

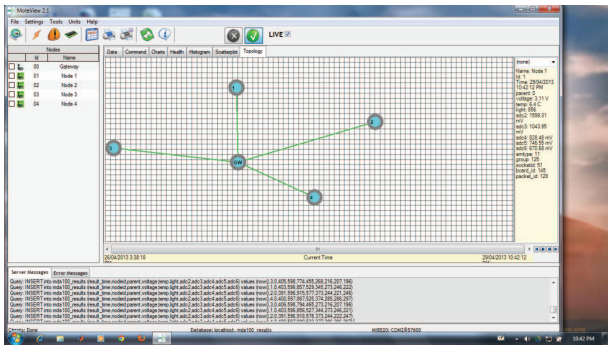

Fig. 8.

9 shows the corresponding multi-hop topology. The node 3 is the parent node for the node 4. When two nodes are kept even

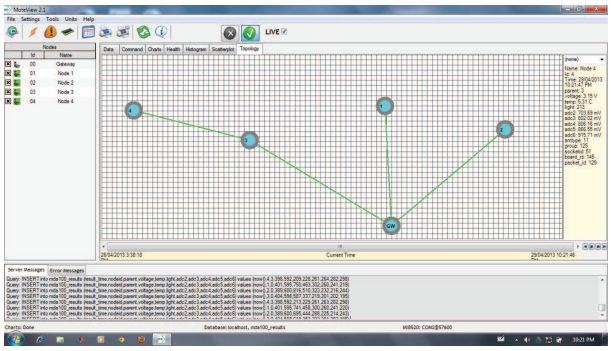

Fig. 9. Multi-hop routing in WSN (two hops)

at longer distances from the gateway ($node4 : 30m, node2 : 50m$), a child node takes total of three hops to get connected to the gateway. This experiment proves the reliability and robustness of the *XMesh* protocol. However, the time required to update the topology is on an average 1 minute, which causes loss of data packet during this time.

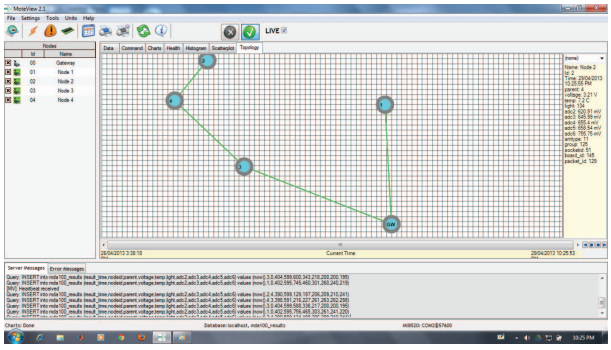

Fig. 10. Mulit-hop routing in WSN (3 hops)

B. Node failure detection

To test a node failure detection feature of the *XMesh* protocol, the node 3 is switched off. The node failure is detected and showed in the topology tab of the Moteview,

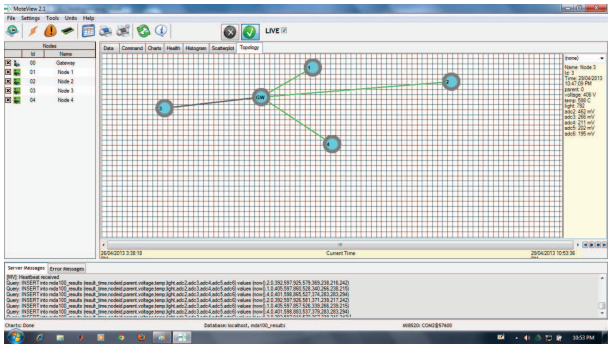

Fig. 11. Faulty node detection in Moteview

as depicted in Figure 11. The detection process takes on an average 5 minutes.

To analyse the robustness of *XMesh* protocol, 11 sensor motes and a gateway (GW) have been deployed in indoor conditions. Initial set-up of nodes to achieve full connectivity across the complete area (approximate radius of the area- 30 m) takes time and require more number of nodes to connect the farthest node. But after the deployment, even after some nodes have failed , the network still survived and topology has been rearranged without any human intervention. Figure 12 shows the network topology during the initial phase. The parent node 7 is turned off to study the protocol's robustness against node failures. The child nodes connected to node 7 started searching for new parent node on the basis of link quality. No node other than 7 is disconnected and the network continued to sense the data from the surroundings. The updated network topology is depicted in Figure 13. Similarly, another parent node 2 is also turned off. This time other nodes in the network have been affected. Figure 14 shows the updated topology.

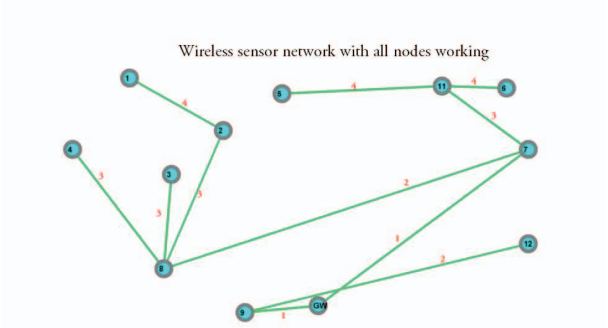

Fig. 12. WSN multi-hop topology with all the nodes working

C. Data fusion in WSN

In a WSN based on the *XMesh* protocol, a parent node connecting two or more child nodes has to forward lot of extra packets to the gateway. This leads to energy drain of the parent node. To avoid such extra communication overhead, a dynamic data aggregation technique is proposed in this work.

XMesh protocol is a dynamic routing protocol and hence parent nodes are altered frequently based on the link quality

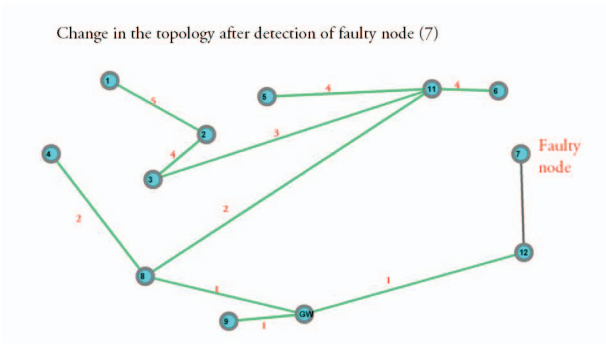

Fig. 13. WSN multi-hop topology update with a faulty node (7)

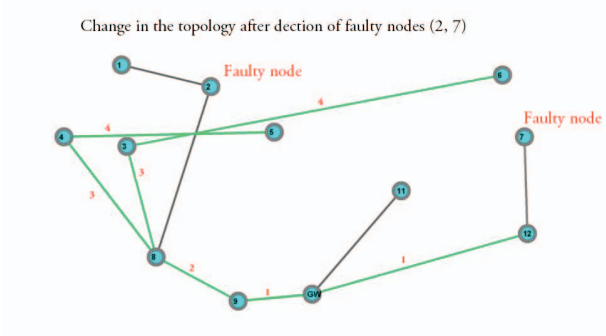

Fig. 14. WSN multi-hop topology update with two faulty nodes (2,7)

parameter. Based on the current topology, the gateway sends a command to the parent node with more children nodes to carry out data aggregation. The *xcommand* interface provided by *XMesh* protocol is used for this purpose. After receiving the *xcommand* from the gateway, the parent node intercepts the data it is forwarding from its children. The *intercept()* interface in the *XMesh* protocol is used instead of the *Receive()* interface.

D. Statistical analysis in Moteview

Various statistical properties of the data collected by sensor mote can be extracted by plotting its histogram. In Moteview, by selecting the nodes and parameter (light, temperature etc.) histogram can be plotted as shown in Figure 15.

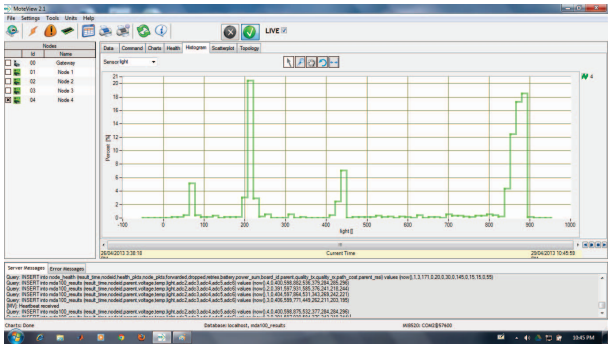

Fig. 15. Histogram of light intensity perceived by node 4

E. Temporal analysis in Moteview

A temporal analysis of the data can also be carried out using Moteview, by selecting *Charts* tab in it. The data for the past hour/day/week/month can be analysed. Also, the Moteview allows to view graphs of more than one sensed parameters and nodes together in a single window. Figure 16 shows a chart of light intensity during an hour span.

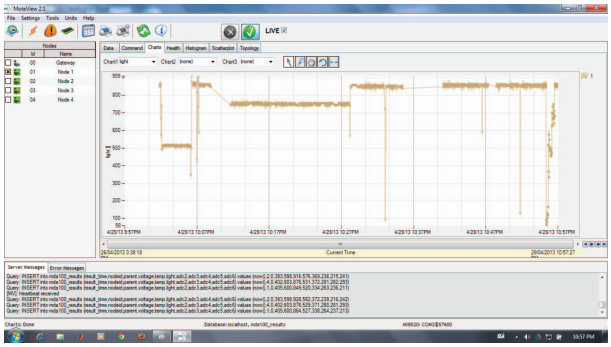

Fig. 16. Temporal analysis of light intensity perceived by node 4

VIII. Conclusion

In this work, *XMesh* networking protocol is presented and its characteristics are experimentally verified using WSN hardware. The *XMesh* has proven to be a reliable and robust protocol for WSN applications, where network scalability and connectivity are important issues. This paper can be considered as a stepping stone during the development of WSN applications using the *XMesh* protocol. A similar set-up can also be used for the development of new networking protocol based on TinyOS. The demonstration of Moteview GUI shows its capability to analyse and present the data in a user friendly manner. Various changes in sensor data is immediately depicted using the Moteview graphing tools. But it is found that topology updates seen in the Moteview are not getting depicted in real time. To conserve the energy of parent nodes, a data fusion approach is proposed using the available interfaces in *XMesh* protocol.

References

[1] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications*. Wiley-interscience, 2007.

[2] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, *Wireless Sensor Networks: Signal Processing and Communications*. Wiley, 2007.

[3] M. Turon and J. Suh, "Mote-view 1.0 users manual," Part 7430-0008-02-A, Crossbow Technology, Inc, Tech. Rep., 2005.

[4] X. U. Manual and D. Revision, "Crossbow technologies," *Milpitas, CA, Apr*, 2007.

[5] D. Gay, P. Levis, D. Culler, and E. Brewer, "nesc 1.3 language reference manual," 2009.

[6] T. W. Davis, X. Liang, M. Navarro, D. Bhatnagar, and Y. Liang, "An experimental study of wsn power efficiency: Micaz networks with xmesh," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[7] A. Teo, G. Singh, and J. McEachen, "Evaluation of the xmesh routing protocol in wireless sensor networks," in *Circuits and Systems, 2006. MWSCAS'06. 49th IEEE International Midwest Symposium on*, vol. 2. IEEE, 2006, pp. 113–117.

[8] S. Waharte, R. Boutaba, Y. Iraqi, and B. Ishibashi, "Routing protocols in wireless mesh networks: challenges and design considerations," *Multimedia Tools and Applications*, vol. 29, no. 3, pp. 285–303, 2006.

[9] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[10] M. Conti and S. Giordano, "Multihop ad hoc networking: The reality," *Communications Magazine, IEEE*, vol. 45, no. 4, pp. 88–95, 2007.

[11] C. Mica, "Micaz, and iris motes."

[12] Z. Alliance, "Zigbee specification," *ZigBee Document 053474r13*, pp. 344–346, 2006.