International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 5; Issue: 6; June-2016: pp 689-693

A Implementation and Analysis of WSN on TinyOS

Vishal Nagar
PSIT Kanpur

Asmita Dixit
PSIT Kanpur

***Abstract*— In the current scenario of research the Wireless Sensor Networks have been a core field of research. In the area of wireless sensor networks through its different scopes, data collection has been a vital scope of research. For the routing of data packets from the origin to the destination there are several wireless sensor networks which most of the times depend on certain collection driven service. Chosen through several protocols collection tree protocol is a significant one although there are different protocols that have been designed for data collection in wireless sensor networks. For this a specific operating system has been devised named TinyOS. Also keeping in mind the constraints of sensor networks in consumption of power view point, the operating system gives a distinguished platform for the development of several implementations in WSNs in collaboration of limitations of the sensor equipments. Due to sometimes placing of sensor networks in isolated locations, the testing of implementation in such scenarios becomes inconvenient. Henceforth to assist the progress of the complete testing of the protocol a network simulation can be performed. The paper basically manages with the approach based on TOSSIM (TinyOS Simulator) approaching simulation of collection tree protocol in wireless sensor networks and the determination of the performance of the protocol in consideration with several aspects. TOSSIM or TinyOS Simulator is a prominent simulator which can be made use of in simulation of an entire TinyOS application. The paper also brings forth an assessment that will depict a brief look about the performance of collection tree protocol and simultaneously will give awareness on how TOSSIM can be worked with as a simulation platform for implementation of collection tree protocol.**

***Index Terms*—CTP (Collection Tree Protocol), Motes, TinyOS Simulator (TOSSIM), Wireless Sensor Networks.**

I. INTRODUCTION

For capturing the information of the real world scenario through communication component motes, wireless sensor networks can be a medium of collection of such information. Decisions are taken by the administrator on the basis of the data being gathered by motes and their specific reactions to situations. Collection of Data has been a significant field of task in a wireless sensor network. Designing of environment collection tree protocol was done for the gathering of Data. Collection Tree Protocol (CTP) is free from address. Collection Tree Protocol is based on the

concept that the data link layer gives effective restricted broadcast address, synchronous acknowledgements for unicast packets, protocol dispatch area and a single hop origin and target field. Assumption is made on the linking quality of the nodes that are nearby. Various origins can transmit data to a distinct sink with the usage of this routing protocol.

For the forthcoming analysis of the data being gathered at the destination it can be stored and recorded. Since CTP examines the quality of link of the adjoining neighboring nodes, it also works with the expectation about the assumpted units of transposal a node will take to transmit a unicast packet to the target.

For a unique platform for operation of CTP for the gathering of data an event driven operating system is designed uniquely for network embedded sensor motes in addition to it a network embedded system C or nesC has also been programmed [8]. The paper depicts a data collection scheme with the use of Collection Tree Protocol and hence on the basis of few performance reasons like packet delivery ratio, an estimate is made in consideration to the fluctuating levels of radio frequency (RF) and frequency of channel. Based on the concept of Micaz motes the protocol has been designed. TOSSIM (TinyOS Simulator) has been used for the complete simulation. TOSSIM a distinct event based simulator assembles a TinyOS application into TOSSIM framework that implies testing, finding errors and determining algorithms in a restricted and repetitive environment. Furthermore a complete network along with its implementations can be simulated with the use of TOSSIM. It most of the time simulates the working of Micaz motes.

II. RELATED WORKS

Within the last two decades the wireless sensor networks research community has developed a wide variety of algorithms for efficient and reliable source to sink collection of data related to sensed phenomenon. These include several power-aware medium access protocols and reliable routing schemes [2]. In any case the Collection Tree Protocol provides a best effort any cast datagram

interaction with one of the collection roots in a network [3]. Collection Tree Protocol or CTP is widely regarded as a reference protocol for performing data collection in wireless sensor networks and its specification is provided in TinyOS Enhancement Proposal 123 [5]. Gnawali et al. also report a throughout description and performance evaluation of CTP in realistic settings, demonstrating the ability of the protocol to reliably and efficiently report data to a central collector [5, 6]. Several works has also been done mainly in testing the CTP using Telosb motes as well as simulating it using network simulator. Several simulators have been developed for simulation of networks [1], for example SimOS [4] used binary rewriting techniques to provide enough detail and yet enough speed and flexibility to allow workload-driven evaluation of machine architectures and operating systems for multiprocessors by running whole programs. Ns-2 [4], using an object approach, provided a common toolbox for studying a wide range of network protocols and implementations against various traffic models. Proteus [6] provided broad feedback on the design of parallel programs. TOSSIM, a simulator mainly meant for simulating TinyOS applications simulates an entire application and thus captures the network interactions as well [9].

III. TOSSIM

TOSSIM or TinyOS Simulator, primarily a discrete event based simulator provides platform to depict errors, evaluate and analyse algorithms in a restrained and repeatable scenario. It provides complete scalability, fidelity and bridges the gap between algorithms and applications. Using it at a very fine grain it can capture the nature of a mote. Large amount of simulation of motes can be performed at once. Limited to its scope As TinyOS provides a very event driven implementation, this accomplishment scheme most of the times goes well with the discrete event simulation given by TOSSIM [9]. In Sensor networks evaluation of an individual mote is inadequate because it generally follow an event driven nature. In order to capture a large range of communications, programs must be tested at scale, in complex and rich environments. Since development of sensor network needs large amount of motes leading to a difficult task as a result much of the time is spend in maintenance rather than development. Hence the complete process is bit time consuming and if anyway a mote of a remote location fails, situation handling of failure and redeployment becomes more time consuming for the complete network. In the scenario of evaluation such type of overhead are not taken into account. For all these difficulties simulator is a solution because it can tackle them by giving restricted, repeatable environments, by having tools like debuggers and delaying deployment till the code is better tested and algorithms better understood.

IV. SYSTEM MODEL

Typically in a sensor network, the information that is collected by multiple sensors needs to be transmitted to a remote central processor that is called a sink node or simply sink. If the sink node is far away, the data of the sensed phenomenon recorded by the sensor node may first be transmitted to a relay node, and then multihop routing is used to forward the data to its final destination. There are multiple nodes that want to transmit their collected information to the sink node. The corresponding scenario is illustrated in figure 1. If there are N sensor nodes in the network and without loss of generality if the sink node is denoted as the Nth node then the other N-1 nodes either have their own data to send to node N, or they just act as relay nodes to help others. The topology that has been followed is strictly a tree topology taking a total of ten nodes and one node acting as the sink node. The routing gradient used is called expected transmission value or Expected Transmission Count (ETX) **[**5**]**. One hop ETX is determined by calculating the number of transmissions it takes for a node to send a unicast packet to its neighbor whose acknowledgement is successfully received. ETX (root) = 0. The equation for determining the ETX is given as ETX (node) = ETX (parent) + ETX (link to parent)

The arrangement of nodes has been shown in fig.1.The nodes follow a tree topology.

CTP chooses the route with the lowest ETX value. Link estimation in CTP design is used for determining the communication link quality between the neighbors. The bidirectional link estimate value ETX is computed by using both routing beacons and unicast data packets. The routing packets are sent periodically to calculate the bidirectional link quality between the neighbors. This value fills the link estimator neighbor table. The CTP make use of the data transmissions as well to calculate the outbound link quality which is then combined with the control packets link estimate. In a stable network data packets are used to keep track of any link quality changes and therefore the number of control packets is reduced. After the transmission of *n* number of data packets new outbound quality estimate is performed [3], where *n* is implementation dependent. The outbound quality estimate value is the ratio of number of data packets transmitted to the number of acknowledgements received. The MAC layer gives the acknowledgement information to the forwarding engine. The forwarding engine removes the data packet from its send cache and informs the link estimator engine about the acknowledgement.

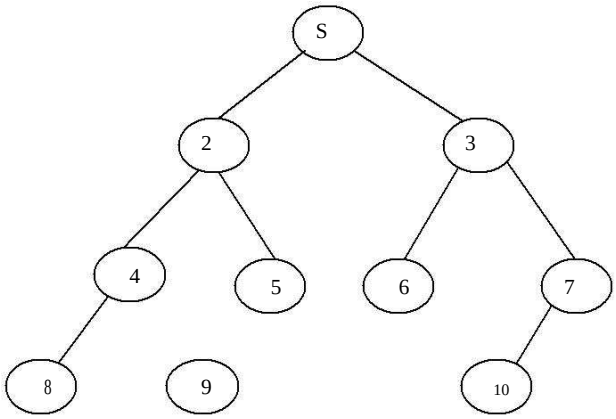

Fig.. 1. Arrangement of nodes

V. CTP IMPLEMENTATION

CTP uses a set of beacon messages for construction of the tree topology and data messages to report to sink. In

particular, application-level modules can call a generic collection service which is in turn implemented through CTP [5, 6]. The standard implementation of CTP consists of three main logical software components, the Routing Engine (RE), the Forwarding Engine (FE), and the Link Estimator (LE).

A. Routing Engine

The Routing Engine, is a detail that runs on every node analyses well the sending and receiving beacons also creates and updates routing table [4]. There is a list of neighbors held in this table so that node can choose its parent in the routing tree. In the table the information gathered from the beacons is filled. Further information is held by the routing table in account with the identifier of neighboring nodes for example indication of the quality of a node as a potential parent. In the case of CTP, this metric is the ETX (Expected Transmissions) which is communicated by a node to its neighbors through beacons exchange. A node having an ETX equal to n is expected to deliver a data packet to the sink with a total of n transmissions. The ETX of a node is defined as the "ETX of its parent plus the ETX of its link to its parent" [5]. More precisely, a node first computes, for each of its neighbors, the link quality of the current node-neighbor link. This metric, which is referred to as the 1-hop ETX, or ETX1hop, is computed by the LE. For each of its neighbors the node then sums up the 1-hop ETX with the ETX the corresponding neighbors had declared in their routing beacons. The result of this sum is the metric which has been called the multi-hop ETX, or ETXmhop [6, 7].

B. Forwarding Engine

The forwarding data for which the forwarding engine takes care for may be originated from either the application layer of the similar node or from the neighboring nodes. Suppression of duplicate packets as well as detection and repairing of routing loops is one of the responsibilities of forwarding engine. Also these two responsibilities are one of the main characteristics of TinyOS[6]. Whereas that meant for the estimation of 1-hop link quality is dealt by Link Estimator in CTP.

C. Link Estimator

The Link Estimator takes care of determining the inbound and outbound quality of 1-hop communication link [5, 10]. As mentioned before, reference to the metric that expresses the quality of such links as the 1-hop ETX has been drawn. The LE computes the 1-hop ETX by collecting statistics over the number of beacons received and the number of successfully transmitted data packets. From these statistics, the LE computes the inbound metric as the expected number of transmission attempts required by the neighbor to successfully deliver a beacon. To gather the necessary statistics and compute the 1-hop ETX, the LE adds a 2 byte header and a variable length footer to outgoing routing beacons.

VI. PERFORMANCE METRICS

To evaluate a particular protocol we always need a set of performance metrics. The performance metrics should be such that they give an outline of the protocol. For evaluating Collection Tree Protocol the following performance metrics has been taken.

A. Packet Delivery Ratio

Packet delivery ratio is the ratio of the total number of packets delivered to the total number of packets sent by the senders. Packet delivery ratio also takes into account the number of data values the network can send to the sink. If the packet delivery ratio equals to 1 then it can be said that all the packets has been delivered successfully by the network. In the worst case, none of the collected data values reaches the sink. This may happen if the sink is disconnected from the network and causes the packet delivery ratio to be zero. Packet delivery ratio being a very important performance measurement metrics, it has been calculated under varying channel frequency and variable radio frequency power as well.

B. Throughput

Throughput refers to the total number of packets delivered to the sink node. This is done by checking the counter value at the sink. If a packet arrives at the sink node, then the counter increments by 1 and thus the value is printed in the screen along with the node id. The counter value is depicted in hexadecimal format. With each node id one counter value is associated. Figure 3 gives a rough idea about the packet count for four nodes.

Apart from packet delivery ratio and throughput, to evaluate the performance, the packet delivery ratio has been checked under variable radio frequency (RF) power and variable channel frequency. This has been done to estimate the reason for packet loss.

VII. SIMULATION

The TinyOS stack uses three network sampling rates at different phases of packet reception and transmission: 40Kbps for data, 20Kbps for receiving a start symbol, and 10Kbps for sending a start symbol [7, 8]. In TOSSIM, adjustments to radio bit-rates are made by changing the period between radio clock events. The combination of bit sampling and bit-rate changes nearly captures the entire stack. There is one exception: the pair of spin loops to synchronize a sender signal, the one place where TinyOS breaks its event-driven methodology.

Under simulation, the event-driven concept is maintained by ignoring the first spin loop (for the zero) and handling the second (for the one) with additional state. Whenever a mote transmits the synchronization bit, it checks if any of the motes that can hear it are in the synchronization listening state. If it finds such a mote, it enquires a radio event for the receiver which represents the occurrence of the input capture. This implementation results in an almost perfect simulation of the TinyOS networking stack at a bit level. TOSSIM accurately simulates the hidden node problem and can simulate errors at all phases of packet reception. When two nodes have interfering transmissions, a third listening node will just see the union of the two

sender's bits, leading to both signals being corrupted. Additionally, delay arises when motes repeatedly enter CSMA wait because they continue to hear a signal on the channel.

In this simulation ten nodes has been taken, with one node being the sink node where all the packets are being collected from the source nodes. Nodes nearer to the sink acts as relay nodes.

The Easy Collection module present in /opt/tinyos-2.1.0/apps/CTP1/EasycollectionC.nc has been run at the forwarding nodes as well as at the sink node and other nodes. The simulation output shown in figure 2 reflects the programs being successfully installed and compiled in the motes. After the successful compilation, the python file is run which provides the printed output of the nodes booting up and connecting with the sink node that is node1.

After the initial connection set up, the Easy collection module is again run by considering the sink node. Thus the simulation results in fig.3 show the node id and the counter value which reflects the number of packets received at the sink.

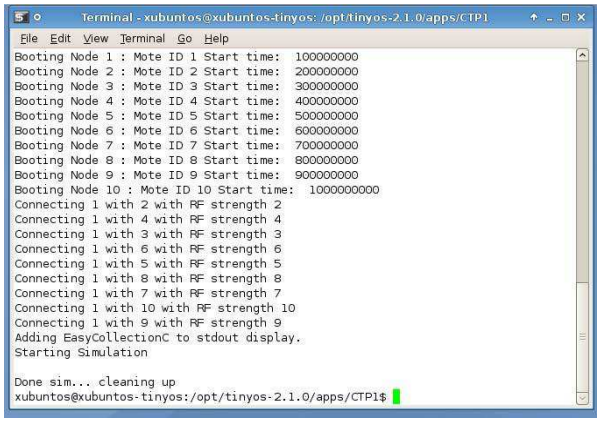

Fig. 2. Simulation output showing the connection initiation.

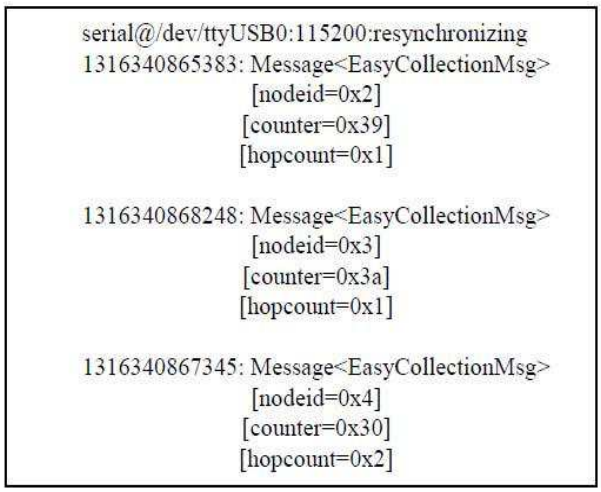

Fig. 3. Simulation output at sink node for up to four nodes.

VIII. PERFORMANCE ANALYSIS

The analysis has been carried out based on the experimental results to the metrics introduced in section VI, i.e., the packet delivery ratio, and throughput. The packet delivery ratio being an important performance incentive, it has been evaluated by varying the radio frequency power and the channel frequency.

A. Packet Delivery Ratio

To analyze the ability of CTP to send packets from source to the sink either directly or through forwarding nodes, about 10 nodes has been taken with one node acting as the sink node and the others are acting as source node or the forwarding nodes. The graphical analysis takes into account the packet delivery ratio in y- axis and the nodeid in x-axis. The first counter value at the sending node and the last counter value at the receiving node are taken and the difference gives the packets received. Thus a ratio between the packets sent and packets received is calculated which gives the packet delivery ratio and is plotted in the y-axis. After plotting the values obtained from simulation, the graphical representation has been shown in fig. 4.

From fig. 4, it is seen that the packet delivery ratio significantly decreases as we go down the tree, that means for the nodes which are nearer to the sink, the number of packets received are more as compared to the nodes which are further away from the sink.

The packet delivery ratio is much lesser for the leaf nodes. Observation is made by evaluating the packet delivery ratio by changing the radio power at the nodes, and channel frequency as well. The main reason behind changing the radio power and channel frequency is to get an idea about the reason for the packet loss.. Fig. 5 shows the graphical representation of the packet delivery ratio with respect to changing radio power levels for 7 nodes.

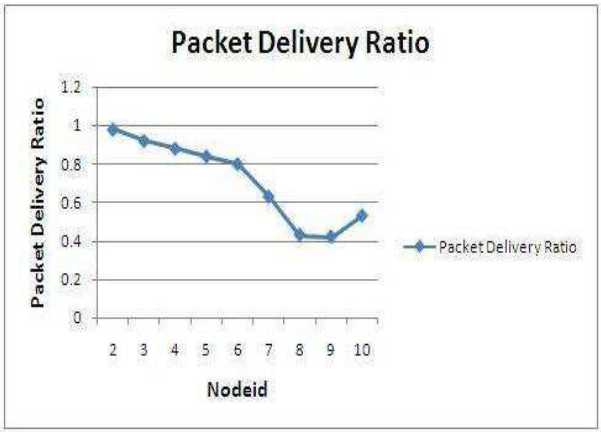

Fig. 4. Packet Delivery ratio

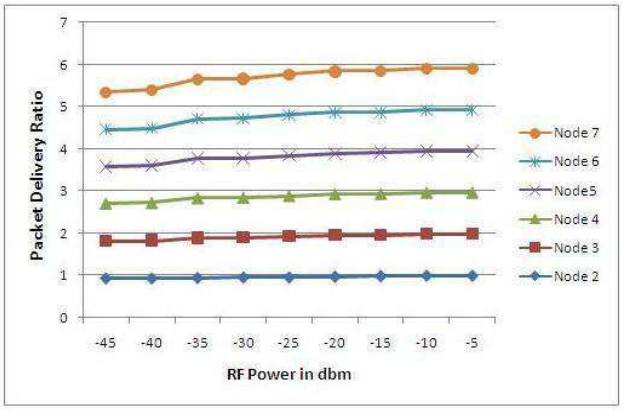

Fig. 5. Packet delivery ratio on changing the RF Power

Thus by changing the channel frequency and radio frequency (RF) power, it is seen that the packet delivery ratio significantly increases as the RF power increases. The changes in radio power and channel frequency have been made in the configuration file for running TinyOS applications in TOSSIM.

B. Throughput

Throughput calculates the number of packets received at the sink node. It is evaluated by checking the counter value at the sink node. The simulation output shows the counter value. The counter is incremented automatically on reception of a packet at the sink node. The output shows the counter value associated with each node id. The counter value is generated in hexadecimal. The hexadecimal value is then converted to decimal and the resulting decimal value is taken as the count for the actual number of packets received at the sink node. Fig. 6 shows the graphical representation of the number of packets received over a period of 5 seconds.

Thus from fig. 6 it is evident that the throughput is maximum for the nodes which are located near the sink. The throughput decreases as the distance of the nodes from the sink increases.

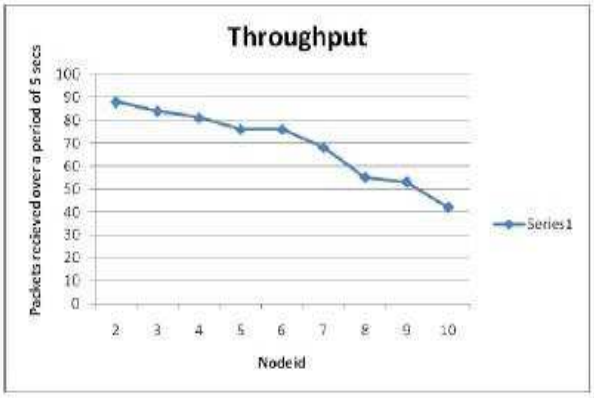

Fig. 6. Throughput

IX. CONCLUSION

Hence a detailing is provided in the paper regarding the application of Collection Tree Protocol in wireless sensor networks using a simulator TOSSIM (TinyOS Simulator). The basic focus area of paper is done on the application part in contrary to other researchers who present an accessible reference for those who feel an interest in CTP and attempt its re-implementation in some other platforms. The effectiveness of the application has been evaluated through a considerable simulation study, which also validates persuasive performance of CTP in context of packet delivery ratio and throughput. As we go down the level in the tree the throughput and packet delivery ratio degrades, the application and the results of simulation also depict this. Distance is also contributing its role as the nodes which are a bit farther from the destination have less amount of packet delivery ratio. Although the packet delivery ratio can be gradually increased by altering the radio power and channel frequency. Hence the ratio can be increased if there is an increment being done in the radio frequency power in combination with the channel frequency. Also a loss of packets is suffered because of low radio frequency power and channel frequency. Since at the current scenario TOSSIM works only for Micaz series of motes but later works can be implemented for simulating the behavior of a sensor network for Crossbow Telosb series of motes.

REFERENCES

[1] Athanassios Boulis et al. Castalia: “A Simulator for Wireless Sensor Networks.” Available at URL :http://castalia.npc.nicta.com.au/.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. “Wireless Sensor Networks: A Survey.” Computer Networks, 38(4):393–422, March 2012.

[3] Ioannis and Kinalis, Athanasios and Nikoletseas, Sotiris Chatzigiannakis, "Sink Mobility Protocols for Data Collection in Wireless Sensor Networks", Proceedings of the *4th ACM International workshop* on Mobility management and wireless access 2014.

[4] J. E. Egea-López, A. Vales-Alonso, P. S. Martínez-Sala, J. Pavón-Mariïo, and García-Haro. “Simulation Tools for Wireless Sensor Networks,” In International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2015), Philadelphia, PA, USA, July 2015.

[5] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis “Collection Tree Protocol,” In Proceedings of the *7th ACM Conference* on Embedded Networked Sensor Systems (SenSys 2012), Berkeley, CA, USA, November 2012.

[6] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, and Philip Levis. “CTP: Robustand Efficient Collection through Control and Data Plane Integration”. Technical report,The Stanford Information Networks Available at URLhttp://sing.stanford.edu/pubs/sing-08-02.pdf.

[7] Pascal von Rickenbach, Nicolas Burri, and Roger Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2013), Cambridge, MA, USA, April 2013.

[8] Philip Levis and David Gay “TinyOS Programming”, Cambridge University Press, 2012.

[9] Philip Levis, Nelson Lee, Matt Welsh, and David Culler "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications." In Proceedings of the *First ACM Conference on Embedded Networked Sensor Systems* (SenSys 2013).

[10] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, and A.Fapojuwo, “A centralized energy-efficient routing protocol for wireless sensor networks”, *IEEE Radio Communications* Magazine, 2015, pp.S8-S13.