An Energy Efficient Remote Data Collection and Reprogramming of Wireless sensors Networks

Hemappa B[1], Shylaja B T[2], Dr.Manjaiah D H[3] , Dr. Rabindranath Bera[4]

[1]*Research Scholar, Dept,of Electronics and communication Engineering, Sikkim Manipal Institute Of Technology, Majitar, Rangpo, East Sikkim Sikkim-737132, India.*

[2]*Consultant, Parivartan Software Solutions, Hubli, India.*

[3]*Professor ,Department of Computer Science, Mangalore University, Mangalore, India*

[4]*Professor, Department of Electronics & Communication Engineering, Sikkim Manipal Institute of Technology, Sikkim-737132, India.*

***Abstract*— Energy constrained sensor nodes are preloaded with the program code and data, which are usually unattended after deployment. Due to change in user requirements and environmental changes, preloaded software and firmware deployed in sensor nodes have to be reprogrammed on large number of sensor nodes to resolve the post deployment issues. It's not easy and may be impossible to reach the remote sensor nodes deployed in hostile or private environment. The data collection from the sensor node to sink is equally important to disseminate the required program code and data from sink or host to sensors for reprogramming the new version of the software. The energy efficient dissemination of required program code and data to all the nodes in wireless sensor network is challenging task. Many WSN has proposed regular intervals of sleep-wake up of sensor nodes to enhance the lifetime of the sensor network. We propose an energy efficient gateway based remote reprogramming of the required software/firmware in sleep scheduled wireless sensor network without affecting the functionality of the wireless sensor network. The theoretical analysis and the simulation results reveal that our proposed scheme effectively transmit and reprogramming the software on deployed sensors by consuming less power resource and increases the network life time.**

***Keywords*— *Reprogramming, cluster head, gateway, firmware update, event-driven, sleep-wakeup*.**

I. INTRODUCTION

Wireless sensor network (WSN) consists of a large number of tiny embedded devices with battery-powered sensor nodes that integrate sensing, computing and communication capabilities. WSN is one of the rising technologies for next decade in wireless communications with a wide range of applications, such as environmental monitoring, health care, seismic and structural monitoring, factory, process automation monitoring and battlefield control. Each device is in charge of sensing (measuring), computing and transmitting its information. Individually, each node is autonomous and has short range; collectively they are cooperative and effective over a large area. The general functions of each sensor node are the data collection, analysis and transmission, normally using wireless links and multi-hop routing to a sink or base station(BS). WSNs are usually deployed to monitor static or dynamic events. Static events (such as temperature and humidity) are easy to capture. The dynamic events are typically non-cooperative and they are not easily monitored as they come and go. As a result, they can be observed only if the sensors are constantly monitoring the environment. The energy efficiency is a serious concern if the sensors are always ON and event missing rate will be increasing if the sensors are not constantly monitoring the environment. The improvements in hardware design along with the wide availability of economically viable embedded sensor systems will soon drive us into the ubiquitous silicon era[2].

Usually in software development, essential features will be planned in advance and accordingly software releases happens to meet the time to market requirements. The additional features those were not planned during the initial deployment and post deployment bug fixes will be deployed in subsequent releases. In a normal scenario where sensor nodes are easily human accessible could be reprogrammed by transferring the required software objects into sensor nodes and reprogram them to newer version. In this case, sensor node to be reprogrammed should stop its functionalities until completion of program code and data transfer from host server/sink to sensors nodes. As a WSN is usually deployed in hostile environments, secure reprogramming is and will continue to be a major concern[3] and it is not feasible to manually collect the deployed nodes[4]. One of the primary characteristics of a WSN is its longtime functionality. Because of the large number of sensor nodes that may be deployed, the large and diverse geographical area that may be covered, replacing the battery is usually not an option[8]. One of the main requirements of reprogramming the software in wireless sensor networks is to perform remotely when nodes are in situ, embedded in their sensing environment[15]. Software reprogramming should not impact on the regular functionalities of the sensor network and it has to be in an energy efficient way.

Network reprogramming renewal the running code through wireless channel while traditional way through parallel or serial cable; and network reprogramming can update the whole network with only one node, on the contrary, traditional way can only update one node at a time. So, network reprogramming is more comprehensive and effective[5]. Reprogramming requires the program code to be delivered in its entirety. However, wireless communication is unreliable due to possible signal collisions, interferences and packet contentions. Furthermore, network topologies may change due to node failures, node mobility[1], which makes reliable protocol designs more challenging.

The critical service required to enable over-the-air reprogramming is a reliable code dissemination protocol (i.e., reprogramming protocol). Deluge [6] is a highly optimized reprogramming protocol for TinyOS. It uses a NACK-based protocol for reliability and employs segmentation (into pages) and pipelining for spatial multiplexing. However, in sparse and lossy networks, the performance of Deluge degrades due to the need for a large amount of data retransmissions [7].

A. Our Contribution

Network reprogramming have many sub tasks that are open for addressing them and we consider the problem as a whole and then divide it into different sub issues. We then propose to resolve sub issues such as code disseminate, encoding /decoding and validation of received image in the receiver sensor node.

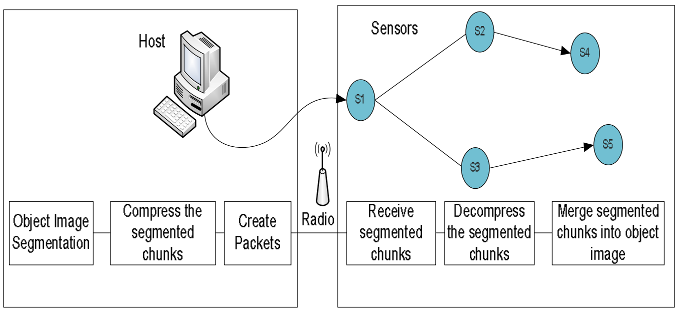

Fig.1 Data dissemination in network.

We have divided the network reprogramming task into multiple sub tasks and addressed them in an efficient manner.

- Study and analyze existing reprogramming algorithms [1, 2,5,19,20] for different parameters including power efficiency and reliability.
- We proposed a new algorithm for code dissemination which would be most suitable for remote reprogramming of wireless sensor network when nodes are at a distance of more than single hop from the base station.
- Designed the sub system as shown in Fig. 1 to divide the binary image to be reprogrammed into multiple segments of equal sizes. Each segment should be compressed using the compression algorithms to reduce the transmission cost. At the receiver node, care is taken to validate the complete data transfer and initiating the request to retransmit if the segmented image is not transmitted completely. After receiving, the complete segmented image is decompressed and merged into single image then stored into the flash memory of the sensor node which could be used for flashing and rebooting the node.
- Sensor nodes periodically transmit their data to the corresponding Cluster Head(CH) nodes, they naturally deplete the energy at higher rates to balance the energy across the network we propose an energy efficient routing algorithm which will be rotating the CH role after certain intervals.

II. Related Work

Many data dissemination protocols for sensor networks have been proposed in the literature to address the data communication problem in these networks. Deluge [6] and CORD[18] are an energy-efficient reliable bulk data dissemination protocol for large scale sensor networks. To reduce power consumption as much as possible, the transceiver should be in a sleep mode as much as possible. This can save a lot of power, but there is a caveat in that: constantly turning the transceiver on and off can consume energy. A distinctive feature of CORD is that in addition to adopting a two-phase approach, it aggressively uses sleep scheduling in order to further reduce energy consumption for large object dissemination.

The remote and automatic reprogramming is an indispensable part in the wireless sensor networks [10]. Type-safe modular OS "LORIEN" has designed in [17] and updating software on running system is as easy as its modular architecture. In [9], a MinTax high-level language for energy-aware updates in WSN has been proposed. Remotely updating of software of larger size on deployed sensor nodes is still an open problem. Especially, the energy saving problem is an important challenge because of a limited energy characteristic. Installing new functionalities or modifying existing on-node ones have traditionally taken the form of monolithic binary images, such as those of TinyOS [13]. TinyOS is not an OS in the traditional sense; it is a programming framework for embedded systems and set of components that enable building an application-specific OS into each application. TinyOS is the most common operating system for many WSN. Because TinyOS is a monolithic operating system and full image binaries are needed for each architecture, one may be presented with an energy problem when updating a WSN consisting of such nodes. Updates will be distributed over the network for each architecture, even if the update pertains to the current node's architecture or not. Another commonly used operating system is Contiki and it has alleviated the high cost of dynamically updating functionalities in a WSN to some extent. But they lacked heterogeneity support, as again, binary modules needed to be distributed for each target instruction set. Modular updates in ELF file format such as those for Contiki offer an independence of node hardware at the expense of a larger file to be sent[9]. Updating the large files on remote sensor using wireless communication may lead to resending the files multiple times due to interference in the communication or if sensors are scheduled to sleep while transfer of file in progress. To handle this kind of situation either we have to take the complete control of that sensor node or we have to implement an efficient algorithm to update the software without affecting the scheduled functionalities. We have proposed the software update scheme by splitting the complete image into multiple segments and send it to the target sensor in sequence and this scheme will reduce the retransmission overhead of entire file.

Size reduction is important for software updates; it has a direct impact on the transmission power used and helps in extending the sensor network lifetime[11]. There are two type software updates such as by providing the modular software, so that just the necessary modules need to be transmitted and incremental updates, so that just the changes from the previous version need to be transmitted. The major limitation and incremental updates algorithm is that the scripting language developed is cpu-specific[11], so we have used the modular approach to transfer required module into a sensor from the lighttpd server via gateway.

III. System Model

In our work, it is assumed that all sensors are randomly and densely deployed over the monitoring region such that no isolated node exists in the network. The communication range of the sensors is fixed and the monitoring region is fully sensing covered. The sensing range of a node is variable, which may be larger or smaller than its communication range. Each node has a unique ID for its identification and knows its location information via GPS or through some positioning methods. In our work, we are using new energy efficient algorithm to select the cluster head (CH) as explained in the next section and we have developed and tested the novel software update scheme for sleep-wake up scheduled wireless sensor network as shown in the Fig 2. WSN is considered as a technology to connect information world with physical world together and there is an ongoing trend of integrating multiple wireless network interfaces into user devices, such as laptops and cell phones. Such mobile devices, which are becoming increasingly popular for downloading , uploading, viewing multimedia content and controlling the WSNs[x1, x2], are often located in the coverage area of multiple wireless networks, such as wide-area 3G access networks and local-area WiFi networks.

Proposed system model has three major components such as *gateway*, lighttpd server and *wireless sensor network* which has large number of nodes for collecting the data. In the proposed scheme user can update the software remotely from its system or through smart phone by sending the http command request to the gateway device. Intern gateway will read command file from the http server and parse the client(user) requests which includes one or more nodes to be reprogrammed and number of segments of a software to be reprogrammed on the sensor devices[20]. Example of client request files and command syntax has explained in section IV.

Clustering Model: Clustering data dissemination is proposed for wireless sensor network to decrease the energy consumption and extend the life time of the network [12], [19]. Usually, the nodes within the cluster send the data to CH and CH performs the data pre-processing such as aggregation or compression before transmitting the data to sink. As CH is involved in more data processing and communicating it deplete the energy more quickly. We have proposed the energy efficient CH selection algorithm and clusters are formed among all sensor nodes and in cluster the nodes are further divided into small groups. In each group of nodes , node having maximum residual energy is selected. Among these selected nodes from each group within that cluster , the node having highest residual energy, which was not elected as a cluster head in the previous rounds would be elected as a cluster head.

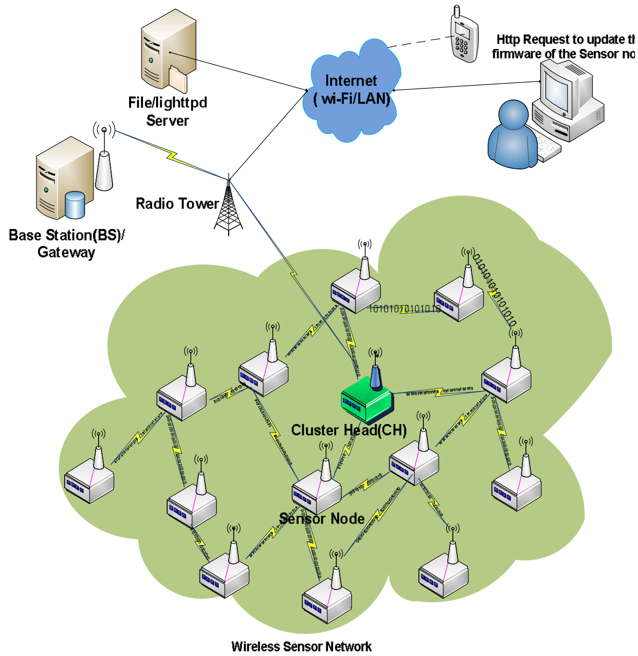

Fig.2 Proposed system layout.

Gateway: A gateway is a device that interconnects two networks and whose presence is usually visible to network users. Gateway can be another sensor which is having more processing and communication power. In our proposed scheme gateway is part of the sink as shown in the Fig 2. A gateway connects devices to help users to store, share, and protect information. It may be required to manage one or more differences between the networks it connects. Change of addressing domain, when the networks have addressing domains managed by separate groups, a gateway may be used to handle address transformation. Control of charging, a gateway may be used to handle user authorization and usage accounting. Change of protocol, when the networks use different protocols a gateway could be used to carry out protocol conversion.

http server: We used the Lighttpd web server to get the images to be reprogrammed on sensor network and cross compiled images are uploaded on the server. The Lighttpd web server is a great way to off load static content like pictures or binary downloads from an over loaded Apache server. Its high speed io-infrastructure allows them to scale several times better with the same hardware than with alternative web-servers. Its event-driven architecture is optimized for a large number of parallel connections which is important for resource constrained embedded systems like WSN applications. The daemon will load a few modules to compress outgoing data and set the expire header to reduce bandwidth of client cached traffic. The security module "evasive" will be used to limit access per ip address to avoid distributed denial-of service attack by multi-connection hackers.

WSN: In proposed WSN, initially deployed in the field, a sensor node turns off and selects a time t_0 to wake up, where t_0 is a random number uniformly distributed within network. At *local time* t_0, it turns on and remains awake for a duration of T_a. Then, it turns off and remains asleep for a duration of ($T - T_a$). This *active-sleep cycle* repeats periodically with a period of *T. As* shown in Fig. 3, during a listen interval, the sensor nodes turn their sensor boards on and take measurements. During a sleep interval, sensors are switched off to save energy.

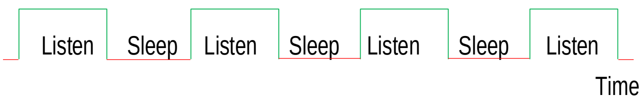

Fig. 3 Sleep-wake up intervals

IV. PROPOSED SYSTEM

Tiny sensors are equipped with small battery and it's a major constraint of wireless sensor nodes which limits their lifetime and affects utilization of the wireless sensor networks. To extend battery lifetime and network utilization, constant changing of the batteries when they run out of energy may not be practical, since these nodes in most cases are many (tens to thousands of sensor nodes) and not easily accessible. Thus recharging the weaken batteries at all time may not be feasible. The energy cost for transmission and reception of the packets is high and the energy cost is depending on the distance between two nodes in transmission [16]. In order to achieve high energy efficiency and increase the network scalability, sensor nodes can be organized into clusters. When cluster heads cooperate with each other to forward their data to the base station, the cluster heads closer to the base station are burdened with heavy relay traffic and tend to die early[12], leaving areas of the network uncovered and causing network partition. In LEACH , each node has a certain probability of becoming a cluster head per round, and the task of being a cluster head is rotated between nodes. We have proposed novel clustering scheme where cluster head role will be shifting to another node in the same cluster and each of the nodes in cluster must have been a cluster head (CH) once and below is the proposed cluster head selection algorithm.

A. Cluster head Selection

For the node to become cluster head in a cluster the following assumptions are made

- All the nodes have the same initial energy
- There are N nodes in the sensor field.
- The number of clusters is C

Based on the above assumptions, the average number of sensor nodes in each cluster is M where

$$M = \frac{N}{C} \quad \ldots\ldots\ldots\ldots\ldots (1)$$

After M rounds, each of the nodes must have been a cluster head (CH) once. We assigned each node a unique identifier i, M_i for all 0,1,2,3,4,...N-1, variable i is used to test whether it is the turn of a node to become a CH. Originally, all nodes are the same, i.e. there is no CHs in each cluster.

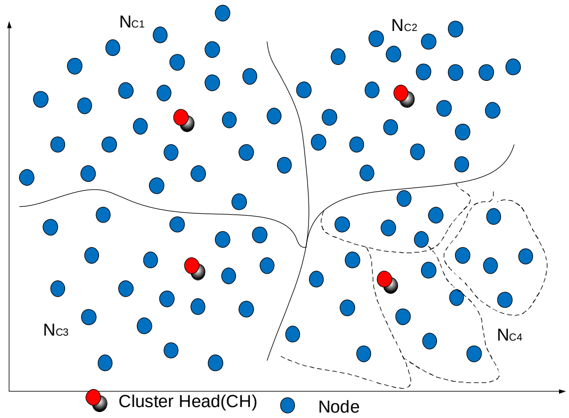

Fig. 4. Clustering of Sensor nodes

Initially a random node is selected as the CH for that round and announces its new position to all member nodes in the cluster. A predetermined value is set (threshold value) for the new CH to transmit for that round. When the threshold value has reached, the process of selection of new CH begins in each cluster regions as shown in the algorithm 1.0. We have used the Divide and Conquer rule to elect the CH in each cluster regions.

Apply Divide and Conquer Rule

Let the number of nodes in a cluster is N_c.
As shown in the Fig 4, we divided each cluster into 'd' number of divisions, i.e. for all k=1,2,3...d, where k is the unique division identifier. And the number of nodes in each division is N_d, where,

$$N_d = \frac{N_c}{d}$$

If Avg(k) is the average residual energy in a particular division k , then we find out the maximum average residual energy among all the divisions, i.e. E_{max}, where,

E_{max} = max(Avg(k_a), Avg(k_{a+1})) where a=1 to N_d

Thus we get the E_{max} and the division index k.

And now in the k^{th} division the node whose residual energy ≥ E_{max} will be the cluster head. This approach helps to identify the CH with minimal computation.

Algorithm 1.0: Find the CH in a region of Cluster

Input : Struct Array of structure sensor, structure SensorGroup , X ← length sensor , K length of SensorGrup

```
for m = 0 to X-1 do
  l ← Nd ← Nc/d
  k ← 0
  for i ← 0 to X-1 step by Nd do
     for j ← i to l-1 do
        SensorGroup[k].average ← SensorGroup[k].average
         + sensor[j].ResidulEnergy
     end for
     if SensorGroup[k].ServedCH == true
     then
               SensorGroup[k].average ← 0
      else
              SensorGroup[k].average ←
            SensorGroup[k].average/Nd
      end if
      l ← l+Nd
      MaxAvg ← SensorGroup[0].average
       groupNum ← 0
       for i ← 0 to k do
          if MaxAvg < SensorGroup[i+1].average
          then
                MaxAvg ← SensorGroup[i+1].average
              groupNum ← i+1
          end if
        end for

      if SensorGroup[groupNum].servedAsCH == true
      then
        Find the next Max residual Energy group
      else
         find the sensor node which is having Max residual
          energy
         Emax ← sensor[0].ResidulEnergy
        for i ← groupNum*Nd to (groupNum*Nd)+Nd do
              if Emax < sensor[i].ResidulEnergy &&
               sensor[i].wasCH
            then
                  Emax ← sensor[i].ResidulEnergy
                  NodeNum ← i
            end if
          end for
          NodeNum is selected as CH for the round
       end if
  end for
end for
```

Thus by applying the divide and conquer method, complexity of calculation is reduced. This approach helps to identify the CH with minimal computation. The new CHs collect sensed data from member nodes, aggregate them, and transmit the compressed data to the next cluster head or base station.

B. Proposed Data dissemination Protocol

The protocols developed for disseminating a software update in a wireless sensor network are primarily based on data dissemination protocols, for example Directed Diffusion and RMST. The key differences between protocols for data collection and for software updating are: (1) the data flow for software updating is from the gateway to the nodes, and (2) the transport must be reliable. The normal pattern for dissemination protocols is a three-step: advertisement of available software, selection of a source, and reliable download to the target, which may then become a source in turn. This may be reversed by use of a 'subscription' approach, but for WSN's this results in significantly increased overhead at the source/server end of the connection.

The sensor devices maintenance is a very important issue in sensor network deployed in the human inaccessible area and is directly related with the quality of service offered to the clients. A continuous inspection and monitoring of the senor functionalities could detect a faulty sensor and inform that an action has to be taken to avoid the possible consequences.

Once user decided to update the software on list of sensor, following steps have been done.

1. Software to be reprogrammed is compiled with cross compiler and copied to the document root directory in the http server.
2. Each software images to be reprogrammed on sensor devices are split into number of small chunks or segments as shown in Fig. 1, and compressed using lossless lzw compression algorithm [14]. Compressed file name details along with path in http server are added to a new file with extension ".bin2" and syntax of the file as shown below.

```
Content of MeddlwareManger.bin2

#START-BIN
#SEGMENTSIZE:10K
#DESTINATIONNODES:ID1, ID2, ID8
#BIN:10
mmangerseg1.lzw
#BIN:10
mmangerseg2.lzw
#BIN:10
 mmangerseg3.lzw
    .
    .
    .
BIN:10
 mmangersegN-2.lzw
#BIN:10
 mmangersegN-1.lzw
#END-BIN
```

L is the size of the software or file and λ be the segment size then

$$R_{seg} = \frac{L}{\lambda}$$

Where R_{seg} is number of segments.

Example: MeddlwareManger.so file has to update on the sensor nodes of ID1, ID2, ID8 and its divided into multiple segments with size of 10K (λ) and formed the new file called MeddlwareManger.bin2
Where ID1, ID2 and ID8 are sensor IDs and mmangerseg.lzw are compressed segments of MeddlwareManger.so.

To download segmented image chucks, first the destination sensor node has to obtain the MeddlwareManger.bin2 file and then obtains content of the segmented image chunks details. Then it requests the chunked images via CH or sink. Client/user will send a http command request to gateway device using the send command. Command usage is as below.
Example :$ send "gateway IP address" "http url"
$send 172.18.10.30
http://172.18.10.35:80/MeddlwareManger.bin2

Gateway will receive the url and connect to the server specified in the url and get the meddlwaremanger.bin2 file from the http server. It will parse the .bin2 file and collect info such as sensor nodes to be reprogrammed and list of files to be transferred and its size as defined in the command file. Gateway will broadcast the message to specified sensor to update their software and will send the total no of segments and size of each segment via cluster head. Later it will send the compressed file to cluster head and cluster head will send the files to corresponding nodes.

HTTP progressive download simply enables destination sensor to start unzip the received chunks/segments when other segments transfer is in progress and start forming the actual file prior to completing the full download of all the segments. This way, communication and computation of resource occur in parallel and are used efficiently in sensor devices.

Following are steps to disseminate the software on sensor nodes from the gateway.

1. Gateway broadcasts Request_info to cluster head, which contains the unique ID of the gateway , software update request, list of sensor ID to be reprogrammed with latest software, filename and its size and wait for the response. Upon receiving that packet, CH will forward the same message to requested list of sensors. On receiving, targeted sensor nodes will responds with Reply_info packet, which contains the residual energy, duration of sleep and listening state of the node. Upon receiving Reply_info packet, CH will add the duration of CH (threshold value) and remaining time of the CH node will continue as CH.
2. Gateway will decide based on the values received from the CH to send the segments to sensor nodes.
3. If the approximate time taken to communicate a segment to the targeted nodes is less than the remaining time of the CH node to continue as CH, then the gateway will send the segments to CH, otherwise it will wait for new CH turn. On receiving the segments, CH will transmit the segments to the targeted nodes using shortest path algorithm.
4. If the node is in listening state, segment will be transmitting to the target node and node responds with Ack_info packet, which contains the total number of segments received. If the node is in sleep state, segments will be transmitting to nearest node and same will be transmitting once target node is in listening state.
5. If the CH turn is getting completed and next CH selection process is starting(use CH selection process), then CH will send the Ack_info packet to gateway, which contains number of segments transmitted and total remaining time as CH.
6. Upon receiving the Ack_info, gateway will repeat the step 3 to 5.
7. Sensor node's computing component will start decompressing the previously received segments when transmission is in progress and create the file name(i.e MeddlwareManger.so) as in Request_info packet received from the gateway and start appending these decompressed segments into to a new file.

This way sensor network utilizing the resource in parallel and also if there is a packet loss during the segment transmission due to interference, retransmission will not cost more as it has to retransmit only the missing segment instead complete file. This approach consumes less energy and will help in extend the lifetime of the network as well as its very good approach to update the software on sensor remotely.

V. SIMULATION RESULTS

There are N sensor nodes in the WSN and they are uniformly distributed in the network coverage area. Each node has a binary event detector with sensing range r. It is assumed that the network covers a 100 m × 100 m area. Simulated the proposed CH selection algorithm throughput using Castalia simulation tool and software update on deployed sensor using

C based algorithms. We have measured the power resource usage of each nodes and found that our proposed system is consuming approximately 18% less power resource compare to LEACH algorithm.

VI. CONCLUSIONS

In this paper we have proposed novel remote software update solutions for WSNs and energy balanced cluster head selection algorithm. Designed software update scheme can be used to handle both sleep-wakeup scheduled and non sleep-wakeup scheduled wireless sensor network. Our approach has proven energy efficiency as retransmission cost during packet loss is comparatively very low. This work can be extended to get the diagnostic information of the deployed sensor nodes to maintain the complete sensor networks remotely.

REFERENCES

[1] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming Wireless Sensor Networks: Challenges and Approaches," IEEE Network Magazine, vol. 20(3), pp. 48–55, 2006

[2] Zeinalipour-Yazti, S. Neema, V. Kalogeraki, D. Gunopulos, W. Najjar, "Data Acquisition in Sensor Networks with Large Memories", Proceedings of the 21st International Conference on Data Engineering (ICDE '05).

[3] He, D., Chen, C. , Chan, S. , Bu, J. "SDRP: A Secure and Distributed Reprogramming Protocol for Wireless Sensor Networks", IEEE Transactions on Industrial Electronics, Dec 2011, doi : 10.1109/TIE.2011.2178214

[4] Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, "Efficient Sensor Network Reprogramming through Compression of Executable Modules", 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,2008.SECON'08.doi:10.1109/SAHCN.2008.51

[5] Huang Ruo-Hong ,Zhang Xiao-Yi ; Wang Xiao-Mei ; Huang Yu-Mei , "Two Energy-efficient and Timesaving Improvement Mechanisms for Network Reprogramming in Wireless Sensor Network", 2nd international Conference on Education Technology and Computer (ICETC), V5-224 - V5-228, 2010. Doi: 10.1109/ICETC.2010.5529779

[6] J.W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04), 2004.

[7] Wei Dong, Chun Chen, Xue Liu, Jiajun Bu, and Yi Gao, "A Lightweight and Density-Aware Reprogramming Protocol for Wireless Sensor Networks", IEEE Transactions on Mobile Computing,Vol 10, No. 10, October 2011.

[8] Yan Zhang, Laurence T. Yang and Jiming Chen,, RFID and Sensor Networks: Architectures, Protocols, Security and Integrations, CRC Press. (c) 2010.

[9] Mihai Galos, David Navarro, Fabien Mieyeville, Ian O'Connor, "Energy-aware Software Updates in Heterogeneous Wireless Sensor Networks" ,IEEE 9th International conference on New Circuits and Systems Conference (NEWCAS), 2011, DOI:10.1109/NEWCAS.2011.5981238

[10] Seung-Ku Kim, Jae-Ho Lee, Kyeong Hur, Doo-Seop Eom, "Tiny Function-Linking for Energy-Efficient Reprogramming in Wireless Sensor Networks", Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp-208 – 213, 2009. UBICOMM'09. oi:10.1109/UBICOMM.2009.22

[11] S. Brown, C. J. Sreenan: Updating Software inWireless Sensor Networks: A Survey. Technical Report UCC-CS-2006-13-07, University College Cork, Ireland, July 2006.

[12] Chengfa Li, Mao Ye, Guihai Chen, Jie Wu, "An Energy-Efficient Unequal Clustering Mechanism for Wireless Sensor Networks", IEEE International Conference on Mobile Adhoc and Sensor Systems Conference,2005. oi:0.1109/MAHSS.2005.1542849

[13] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for wireless sensor networks," in Ambient Intelligence. Springer-Verlag, 2005

[14] Zhou Yan-li, Fan Xiao-ping, Liu Shao-qiang, Xiong Zhe-yuan, "Improved LZW Algorithm of Lossless Data Compression for WSN", 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), pp-523 – 527, 2010, doi: 10.1109/ICCSIT.2010.5563620

[15] Abolhassan Shamsaie, Jafar Habibi "Planning Updates in Multi-Application Wireless Sensor Networks",IEEE Symposium on Computers and Communications (ISCC), pp-802 – 808, 2011, doi:10.1109/ISCC.2011.5983940

[16] Wei Wang, Tie Qiu*, Lei Wang, Feng Xia, Guowei Wu, "A Balanced Energy Consumption Solution for Wireless Sensor Networks with Failure Clusters", IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011, pp:415 - 419 , doi: 10.1109/CSAE.2011.5952880

[17] Barry Porter, Utz Roedig . Geoff Coulson, "Type-Safe Updating for Modular WSN Software" 7th IEEE International Conference on Distributed Computing in Sensor Systems IEEE DCOSS'11 (2011).

[18] L. Huang and S. Setia, "CORD: Energy-efficient Reliable Bulk Data Dissemination in Sensor Networks" , Proceedings of IEEE INFOCOM, 2008.

[19] M. Guerroumi, N. Badache , S. Moussaoui, Z. Doukha, "Clustering Data Dissemination in Wireless Sensors Network" , International Conference on Communications, Computing and Control Applications (CCCA), pp 1 - 6 , 2011

[20] Hemappa B, Shylaja B.T, Dr. Manjaiah D.H, Dr. Rabindranatha Bera,"A Novel Energy Efficient Software Update for Sleep-Wakeup Scheduled Wireless Sensor Network", National Conference on Emerging Trends in Computer Science and Information Technology, 16-17 December 2011, India

[21] D Kaspar, K Evensen, P Engelstad, A F. Hansen, " Using HTTP Pipelining to Improve Progressive Download over Multiple Heterogeneous Interfaces ", IEEE International Conference on Communications, ICC 2010, Cape Town, South Africa, 23-27 May 2010.

[22] Chunguo Jing, Dongmei Shu and Deying Gu , "Design of Streetlight Monitoring and Control System Based on Wireless Sensor Networks ", IEEE Conference on Industrial Electronics and Applications, pp 57-62, 23-25 May 2007, doi: 10.1109/ICIEA.2007.4318369.