International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:8, 2010

Analysis of Data Gathering Schemes for Layered Sensor Networks with Multihop Polling

Bhed Bahadur Bista, *Member, IEEE,* and Danda B. Rawat, *Student Member, IEEE*

Abstract—In this paper, we investigate multihop polling and data gathering schemes in layered sensor networks in order to extend the life time of the networks. A network consists of three layers. The lowest layer contains sensors. The middle layer contains so called super nodes with higher computational power, energy supply and longer transmission range than sensor nodes. The top layer contains a sink node. A node in each layer controls a number of nodes in lower layer by polling mechanism to gather data. We will present four types of data gathering schemes: intermediate nodes do not queue data packet, queue single packet, queue multiple packets and aggregate data, to see which data gathering scheme is more energy efficient for multihop polling in layered sensor networks.

Keywords—layered sensor network, polling, data gathering schemes.

I. Introduction

As sensors are becoming inexpensive, it is believed that sensors will be widely used for environmental monitoring, medical treatment, emergency response, weapon detection, security and for other domains [1]. Sensors will form their own network in which each sensor both serves as a host to generate sensed data as well as a router to transmit and receive data to and from other sensors. However, the main constraint of wireless sensor networks is the low finite battery energy of sensors nodes and their limited computational capabilities. This limits the network lifetime and has significant impact on the quality of the network. The challenge is how to design a sensor network that lasts longer and is robust.

In order to reduce energy consumption in sensors while delivering sensed data from sensors to sink nodes various energy aware routing protocols have been proposed. A good survey of such routing protocols can be found in [2]. The main motivation of such protocols is to find ways for energy-efficient route setup and reliable relaying of data from the sensor nodes to sink nodes so that the lifetime of the network is maximized. However, these protocols are designed for homogeneous sensor networks where all nodes are homogeneous. The disadvantage of homogeneous networks is that if a node is given a special role, such as cluster head, its energy will deplete faster.

To overcome this problem in homogeneous sensor networks, some researchers have focused in heterogeneous sensor networks [3], [4], [5]. In heterogeneous sensor networks, the basic sensing nodes are the same as before. However, the sensing area (i.e. the sensor network) is divided into a number of clusters and in each cluster there is a cluster head which is different from sensor nodes. The main feature of a cluster head is that it has a large computational capability, better power supply and its transmission range is longer than sensors covering all the sensors in its cluster. Therefore communication from sensors to the cluster head is multihop while from the cluster head to sensors is single hop.

The basic task of a cluster head is to form a cluster of sensor nodes around it, gather data from sensor nodes within its cluster area, and forward data to other cluster heads towards the sink node. The basic sensor nodes' task is to forward sensed data to its cluster head. They are not aware of the above layer, i.e. inter-cluster communication. Cluster heads form their own network also. They communicate with each other to forward data in multihop manner to other cluster heads, possibly towards the sink node. For their inter cluster communication they use different channel, i.e. not the same channel they use to communicate to sensor nodes, to avoid collision.

By introducing heterogeneous nodes to form a layered sensor network, the network life time is increased because the life time of sensor nodes is increased due to the reduced number of hops for forwarding data thus consuming less power. Moreover, the sensor network can be easily scaled up if new clusters are introduced to new areas. However, energy spent for channel accessing and maintaining data forwarding protocol remains to be the same as before because it is assumed that a contention-based MAC protocol is used for channel access.

A contention-based MAC protocol is efficient but it is not good for energy-aware sensor networks. Due to the basic operation of the contention-based MAC protocol quite a lot of energy is consumed in idle listening, collisions, overhearing and in collision avoidance. It is reported in [6] that power consumption ratio of a sensor for sending, receiving, idle listening and sleeping is 8.2:7.5:7.3:4.2. A sensor node consumes a significant amount of power for accessing the channel only.

The motivation of this paper is two folds. First we consider a three layer sensor network as shown in Figure 1. In the bottom layer (sensor layer), there are sensor nodes which sense environment, in the middle layer (super node layer), there are so called super nodes which have large amount of power supply and long transmission range. These nodes divide the sensor nodes into clusters and control them, i.e. when to send data by polling mechanism. In the top layer there is a sink node where all data is collected. The sink node controls second layer super nodes by polling them. It asks the super nodes when to collect data from the lower layer sensor nodes. Since the nodes are controlled by polling instead of using a contention-based MAC protocol, the power consumed while

B. B. Bista is with the Faculty of Software and Information Science, Iwate Prefectural University, Japan 020-0193, e-mail: bbb@soft.iwate-pu.ac.jp.
D. B. Rawat is with ECE Department, Old Dominion University, Norfolk, VA 23529, USA, e-mail: drawa001@odu.edu

accessing channel can be saved.

Second we investigate data gathering schemes for polling mechanism mentioned above. Depending upon how the data is gathered, polling schedules need to be changed. We consider three types of data gathering schemes and the effect they have in energy consumption in sensor nodes. In the first scheme sensor nodes do not queue data packet. They forward the packet they received in the next polling. In the second scheme, sensor nodes can queue data packets they have received until they are polled to forward them. In the third scheme, sensor nodes are capable of data aggregation such as averaging the data, finding maximum or minimum of data and so on. The intermediate nodes receive packets from their children nodes and they aggregate it with their own data. When they are polled they forward the aggregated data.

The paper is organized as follow. In Section II, we discuss related works. In Section III, we present layer construction and its operation. In Section IV, we explain data gathering schemes followed by performance analysis in Section V. Finally we conclude the paper in Section VI.

II. Related Works

There are various works on forming hierarchical sensor networks [7], [8], [9], in which the main focus is to form clusters of a sensor network. However, these works assume that the sensor nodes are homogeneous. In our case we consider layered sensor network in which each layer has different nodes, i.e. with different capabilities and power supply.

There are works which have considered heterogeneous sensor nodes [3], [4], [5], but they assume that sensor nodes use a contention-based MAC protocol for channel access. Ye et al. [10] have proposed an improved MAC protocol for sensor network called SMAC protocol in which sensor nodes enter sleep mode periodically to save energy. Though SMAC is more energy efficient than a normal contention-based MAC protocol, it still consumes significant amount of energy in idle listening to access channel as reported in [11].

Zhang at el. [11] proposed polling mechanism in heterogeneous sensor network where a cluster head which has significant amount of computation power and power supply, polls sensors in its cluster to collect data. They have assumed that each node has one packet to send and have shown that finding optimal solution for one packet multihop polling scheme is NP-hard. They have given an suboptimal heuristic online algorithm. Their work is similar to ours but there are two significant differences.

The first main difference is that in [11] all the sensor nodes may not be discovered by cluster head while forming clusters. So the second round of discovery is required. The first round of discovery is initiated by the cluster head and the second round of discovery is initiated by sensor nodes which are not discovered by the cluster head. In our case all the sensor nodes are discovered by the cluster head in the first around. Unlike in [11], there is no need for second around of discovery protocol installed in sensor nodes in our case.

The second difference is the way the connectivity and compatibility for sending data in the cluster by sensor nodes is calculated. In [11], a cluster head polls sensor nodes one at a time to send a packet and other nodes to measure the signal strength. After all the nodes' signal strength is measured by all other nodes then the cluster head asks each node to send the measured data to it one by one. The cluster head then calculates Signal-to-Noise Ratio (SNR) and Signal-to-Interference-and-Noise Ratio (SINR) to find connectivity and compatibility of sensor nodes using SNR and SINR.

We take different approach. In our case each sensor runs neighbor discovery protocol to find neighbors where two way communication is possible and interference nodes where only one way communication is possible due to the different transmission ranges of sensor nodes. The cluster head then collects neighbor nodes set and interference nodes set of each node while discovering sensor nodes (i.e. while forming clusters). From these sets the cluster head can easily find connectivity and compatibility of sensor nodes. The detail is described in the next section. Finally, in this paper we investigate which data gathering schemes will save more energy in multihop polling in sensor networks.

III. Layer Construction and Operation

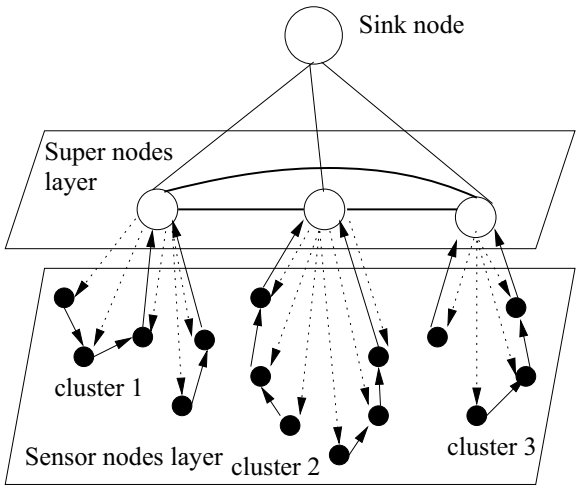

Fig. 1. Layered sensor network

In this section, we describe layer construction and operation of a sensor network.

A. Layer Construction

The network consists of three layers as shown in Figure 1. The lowest layer is a sensor layer consisting of sensors for sensing the environment. These sensors are cheap, have low computational power, small energy supply mainly battery and have short transmission range. They can be deployed randomly, sometimes even dropped from air, in targeted area.

The middle layer consists of super nodes. These nodes have high computational power and have large energy supply, i.e. they do not have energy constraint. They have long transmission range covering a large number of sensors. They are deployed manually so that they can cover all sensor nodes

which are already deployed in sensing area. If the sensing area and transmission range of super nodes are known, the number of required super nodes can be easily calculated and can be deployed within a reasonable amount of time. They have two radio channels, one is used for communicating with lower layer sensors and the other is used for communicating with other super nodes or upper layer nodes. A super node's main task is to form cluster of sensor nodes, poll them to collect data and forward the collected data to other super nodes or to upper layer nodes.

In our case, the top layer consist of a sink node. The sink node instructs super nodes to collect data from sensor nodes and forward the data to it.

In this paper, we only consider how super nodes form clusters of sensor nodes and poll them to send data. We do not consider inter-super nodes communication and how the sink node instructs super nodes to collect data.

B. Clustering of Sensor Nodes

There are various sensor network partitioning researches for flat sensor networks [12], [7]. However, cluster partitioning in layered sensor networks is different. In layered sensor networks, upper layer nodes partition the lower layer network. In our case, first the lower layer sensor network is partitioned into clusters by super nodes in the middle layer. The number of clusters are the same as the number of super nodes. Since it is layered network and the upper layer nodes control the lower layer nodes, it is important for super nodes to know which sensor nodes belong to it and each sensor node to know which one is its super node.

The basic cluster partitioning steps are as follows.

1) Super nodes broadcast message to sensor nodes in turn. Lets say the lowest ID super node starts first.
2) When all super nodes finish broadcasting message, sensor nodes select a super node as the most preferred super node, second most preferred super node and so on depending upon the received signal strength.
3) Super nodes in turn again, lets say starting with the lowest ID super node, asks sensor nodes to send reports to it if they have chosen it as the most preferred super node.
4) When a super node finishes receiving reports from sensor nodes which have chosen it as the most preferred super node, it knows which sensor nodes are in its cluster and the sensor nodes also know which their super node is.

C. Cluster Operation

In this paper, we consider that transmission of each sensor node is asymmetrical which is more realistic than symmetrical because in real world power supply of some nodes depletes faster than others even though they might start with the same power supply.

In order to control sensor nodes, a super node needs to know which sensor nodes can communicate with each other and which sensor nodes are interfered by which other sensor nodes. For example, in Figure 2, *n1* and *n4* are within each others transmission range and can communicate with each other, whereas *n3* can send data to *n2* but *n2* cannot send data to *n3* because *n3* is not within the transmission rage of *n2*. In such case we say *n2* is interfered by *n3*.

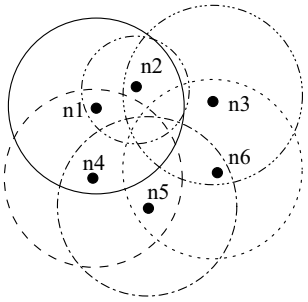

Fig. 2. Neighbors and interferences of nodes

We define *Neighbor Set* (*NS*) and *Interference Set* (*IS*) of a node *n* as *NS(n)* and *IS(n)* respectively. If sensor nodes can communicate with each other then they belong to each others *NS*s. If a node interferes to another node then the interfering node belongs to interfered node's *IS*. For example, for nodes in Figure 2, we have *NS* and *IS* of each node as shown in Table 2. Note that a node cannot be both in *NS* and *IS*.

TABLE I
NEIGHBOR AND INTERFERENCE SETS FOR FIGURE 2

NS(n1) = {n2, n4},	*IS(n1)* = { }
NS(n2) = {n1},	*IS(n2)* = {n3}
NS(n3) = {n6},	*IS(n3)* = { }
NS(n4) = {n1, n5},	*IS(n4)* = { }
NS(n5) = {n4, n6},	*IS(n5)* = { }
NS(n6) = {n3, n5},	*IS(n6)* = { }

We now explain how a super node constructs its cluster and collects *NS*s and *IS*s of sensor nodes in its cluster.

Sometimes after sensor nodes receive broadcast message from super nodes and make their preferred super node list as explained above, each sensor node runs a neighbor discovery protocol. For this they will use a contention-based MAC protocol for channel access. This protocol is executed at the beginning and sometimes in future when some nodes may lose neighbors due to their power depletion.

We describe neighbor discovery protocol briefly as follows. Each node broadcast a "hello" packet containing its ID and the most preferred super node's ID and waits for replies. If it receives a reply with its ID attached in *neighbor field* then the sender of the reply is its neighbor and it puts it in its *NS*. This means it can send and receive packets to and from it. If the sender of the reply was in its *IS*, it will remove it from *IS*.

When a node receives a "hello" packet from a node, it puts its ID in *neighbor field* in the reply packet and replies to the sender. It puts the sender's ID in its *IS* if it is not in *NS*. If it is in *NS* it does nothing. In this way a node cannot be in both *NS* and *IS*. Furthermore, each node stores the most preferred super node IDs of its neighbors.

After sometimes, all nodes will know which their neighbors and interference nodes are. Each node will then check its the most preferred super node ID and its neighbor's the most

preferred super node IDs. If none of its neighbors has the same most preferred super node as its own then it will change its the most preferred super node to one of its neighbor's the most preferred super node from its super node list. This will avoid the second discovery of sensor nodes as in [11] because each node will have the same most preferred super node as one of its neighbors.

Now we explain how a super node collects *NS*s and *IS*s of sensor nodes. We will explain for a single super node, say *H1*. During neighbor discovery, *H1* receives packets from sensor nodes which are near to *H1*. So it knows which sensor nodes can communicate directly to it. *H1* sends a message to these sensor nodes saying that, "send me your *NS* and *IS* if you have chosen me as the most preferred super node". After it receives the list, say from node *n1*, it stores it in a table as shown below.

NS(n1) = {..}
IS(n1) = {..}

The super node will broadcast *ACK* to these nodes from which it has received *NS* and *IS*. The super node will select one of the nodes from the table say *n1*, and asks it to broadcast a message to its neighbors asking them to send their *NS*s and *IS*s to *n1* if they

1) have heard this broadcast from *n1*
2) have chosen *H1* as their the most preferred super node and
3) have not been *ACK*ed by *H1*.

All the sensor nodes that replied to *n1* will be children of *n1*. *n1* will forward these sensor nodes' *NS*s and *IS*s to the super node which will add them to the table. The super node will ACK them. Then it will choose another sensor node from the table which it hasn't asked yet to do the same procedure. In this way all the sensor nodes which have chosen *H1* as the most preferred super node will be discovered. These sensor nodes will form a cluster of which *H1* is the cluster head. Once a super node finishes discovering sensor nodes, another super node will perform the same procedure.

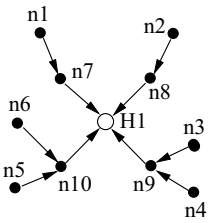

Fig. 3. Polling Example

IV. Data Gathering Schemes

Polling is scheduled according to data gathering schemes. We investigate four type of schemes.

A. No Packet Queuing

In this scheme when a node receives a packet in a polling, it will send the packet in the subsequent polling. In other words, intermediate nodes are not allowed to queue data packet. In other words, when a node is polled to send its packet at time slot T_i, the intermediate node which receives the packet should be polled to forward the packet at time slot T_{i+1} and subsequent intermediate node at T_{i+2} and so on until the packet is received by the super node. The super node, from the *NS* and *IS* of each node, calculates which node can be polled to send their packets at time slot T_i ($i = 0$ initially) without interfering with each other. For example, for Figure 3, a polling schedule for this scheme can be as shown in Table II.

TABLE II
Polling schedule for Figure 3 for No Packet Queuing

Time Slot	*sending node→receiving node*
T_1	*n8→H1, n1→n7*
T_2	*n7→H1, n2→n8*
T_3	*n8→H1, n3→n9*
T_4	*n9→H1, n6→n10*
T_5	*n10→H1, n4→n9*
T_6	*n9→H1, n5→n10*
T_7	*n10→H1*
T_8	*n9→H1*
T_9	*n10→H1*
T_{10}	*n7→H1*

B. Single Packet Queuing

In this data gathering scheme one packet can be queued in an intermediate node. When a node is polled to send its packet at time slot T_i, the intermediate node which receives the packet can queue it until the super node pools it to forward the packet at time slot T_j where $j > i$. Similar to above scheduling, the super node calculates which nodes can send their packets without interfering with each other by looking at *NS* and *IS* of each node. Since intermediate nodes can queue a packet, more nodes can be polled at the same time initially. For example, for Figure 3, a polling schedule for this scheme can be as shown in Table III.

TABLE III
Polling schedule for Figure 3 for Single Packet Queuing

Time Slot	*sending node→receiving node*
T_1	*n1→n7, n8→H1, n3→n9, n5→n10*
T_2	*n7→H1, n2→n8*
T_3	*n8→H1*
T_4	*n9→H1*
T_5	*n10→H1, n4→n9*
T_6	*n9→H1, n6→n10*
T_7	*n10→H1*
T_8	*n9→H1*
T_9	*n10→H1*
T_{10}	*n7→H1*

C. Multiple Packet Queuing

In this scheme an intermediate mode can queue more than one received packets before it can be polled to send the queued packets. Note that if a node queues n packets it needs n time slots to send them. So it will be polled n times. In this scheme also maximum nodes will be polled initially. However, when packets arrive to intermediate nodes which are near to the super nodes only one can be polled at one time slot in order

to avoid collisions. More nodes which are at the leaf of the delivery tree can be polled simultaneously. For example, a polling schedule for this scheme for Figure 3 can be calculated as shown in Table IV.

TABLE IV
POLLING SCHEDULE FOR FIGURE 3 FOR MULTIPLE PACKETS QUEUING

Time Slot	*sending node→receiving node*
T_1	*n1→n7*, *n8→H1*, *n3→n9*, *n5→n10*
T_2	*n7→H1*, *n2→n8*, *n4→n9*, *n6→n10*
T_3	*n8→H1*
T_4	*n7→H1*
T_5, T_6, T_7	*n9→H1*
T_8, T_9, T_{10}	*n10→H1*

D. Data Aggregation

In this scheme, we assume that each node has data aggregation functionality [13], [14]. A node receives data from its children and then aggregates with its own data such as taking average, maximum, minimum etc. When it is polled to send data, it sends the aggregated data. In this scheme since intermediate nodes perform aggregation, polling starts from leaf (edge) nodes. For example, a polling schedule for this scheme for Figure 3 can be as shown in Table V.

TABLE V
POLLING SCHEDULE FOR FIGURE 3 FOR AGGREGATION

Time Slot	*sending node→receiving node*
T_1	*n1→n7*, *n2→n8*, *n3→n9*, *n5→n10*
T_2	*n8→H1*, *n4→n9*, *n6→n10*
T_3	*n7→H1*
T_4	*n9→H1*
T_5	*n10→H1*

E. Data Delivery Path

When a sensor node receives *NS*s and *IS*s from its neighbors, it forwards them to its parent node which will forward to its parent and eventually to the super node. As they do so they add their IDs also. So the super node knows by which path *NS* and *IS* of a node has arrived to it. This path can be used for data delivery path but it may not be an appropriate path depending upon what our interest/constrain are. The data delivery path can be reconstructed from *NS*s and *IS*s of sensor nodes also. In this paper, our aim is not how to construct a data delivery path so we do not explain it in detail here.

F. Polling

Now a super node has enough information to control sensor nodes. The super node calculates which node to poll to send data and which node to receive the data by looking at the *NS* and *IS* of each node.

Since energy consumption ratio for sending, receiving, idle listening and sleeping is 8.2:7.5:7.3:4.2 for sensor nodes [6], what is important to consider in multihop polling is how to reduce the number of sending, receiving, idle listening and sleeping in the cluster. A super node can simultaneously poll nodes n_i and n_x to send their data to n_j and n_y respectively if the following condition is fulfilled.
$n_i \notin NS(n_x)$ and $n_i \notin IS(n_x)$
$n_i \notin NS(n_y)$ and $n_i \notin IS(n_y)$
$n_x \notin NS(n_i)$ and $n_x \notin IS(n_i)$
$n_x \notin NS(n_j)$ and $n_x \notin IS(n_j)$

As shown in previous subsections, polling schedules depend upon on data gathering schemes and will be calculated separately. When the super node finishes calculating the polling schedule it broadcast synchronization signal to all nodes. After receiving synchronization signal all sensor nodes synchronize with the super node. After broadcasting synchronization signal the super node broadcasts waking time for each node according to the polling schedule. After receiving the waking time message each node sets its wake up timer and goes to sleep. The super node records the waking time of each node too. We briefly describe the polling steps of a super node to collect sensed data as shown below.

Polling Steps to collect data

1) Sensor nodes wake up at the previously set wake up time and wait for message from their super node (cluster head).
2) The super node broadcasts synchronization signal. After receiving the synchronization signal from the super node, waken up sensor nodes synchronize with the super node.
3) The super node then polls the sensor nodes according to the calculated polling schedule to send their data. At the beginning of the polling message it attaches which nodes should send and which nodes should receive in the polling.
4) Nodes scheduled for the next polling must have waken up by this time.
5) The super node broadcasts synchronization signal. All woken up sensor nodes will synchronize with the super node.
6) The super node broadcasts next wake up time to those nodes which have already sent data and are not expected to receive data from other nodes for forwarding.
7) The sensor nodes which receive the next wake up message will set the wake up timer and go to sleep.
8) The super node then polls to the next set up sensor nodes according to the polling schedule.

The above steps are repeated until all nodes are polled to send data in one *data collection cycle* which will be explained in next section.

V. PERFORMANCE ANALYSIS

In term of energy efficiency, Zhang at el. [11] have shown that multihop polling in sensor networks performs better than contention-based MAC protocol. So we do not compare them here again. We perform an analysis for a single cluster in which we assume that each sensor node has one data packet to send. We define a *data collection cycle* as one round of data collection from all sensor nodes, i.e. first data of all sensor

nodes are sent to the super node. Since energy consumed by a sensor node depends upon whether it is sending data, receiving data, idle listening or sleeping [6], we count the total number of sending, receiving, idle listening and sleeping in one *data collection cycle* in the cluster. For example, in a single polling, if 5 nodes send, 5 nodes receive, 3 nodes idle listen and 2 nodes are sleeping, then we count the number of sending as 5, receiving as 5, idle listening as 3 and sleeping as 2. A node is sleeping before it sends, receives or it will no more send or receive data within the *data collection cycle*. A node is idle listening if it expects either sending or receiving after its first sending or receiving of data in the *data collection cycle*, i.e. the state of the node between sending and/or receiving.

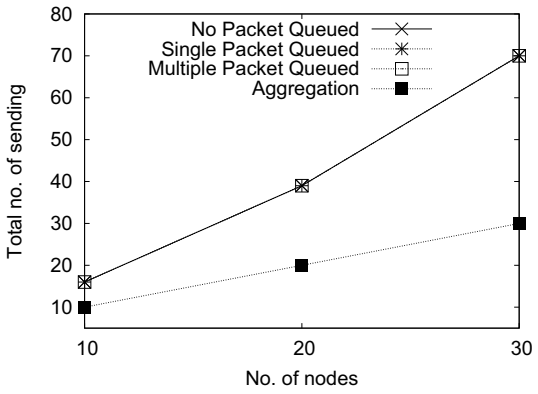

Fig. 4. No. of sending in one *data collection cycle*

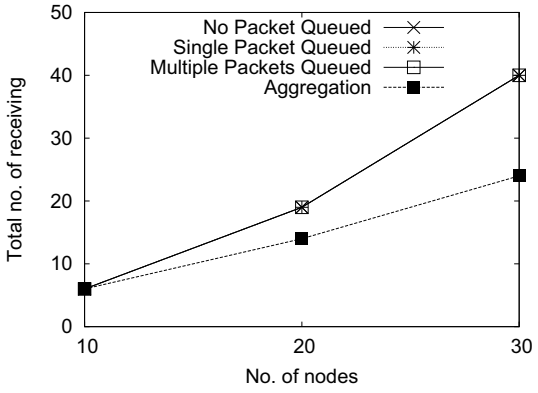

Fig. 5. No. of receiving in one *data collection cycle*

We compare the number of sending, receiving, idle listening and sleeping of data gathering schemes describe in Section IV for multihop polling in sensor network to see which scheme saves more energy in the network.

Figures 4 and 5 show that all schemes except the aggregation scheme have the same number of sending and receiving. This is because whether packet are queued or not the number of packets to be sent are the same for those schemes. For aggregation scheme, there are less number of packets to be sent and thus less number of sending and receiving. Furthermore,

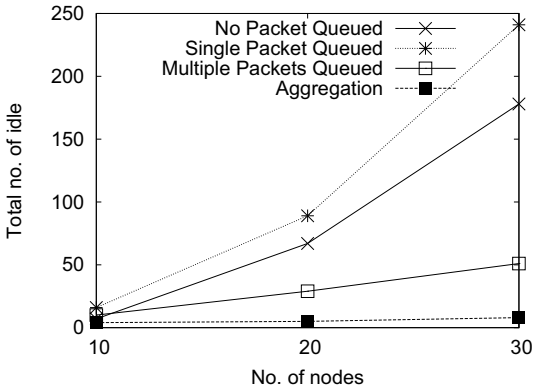

Fig. 6. No. of idle listening in one *data collection cycle*

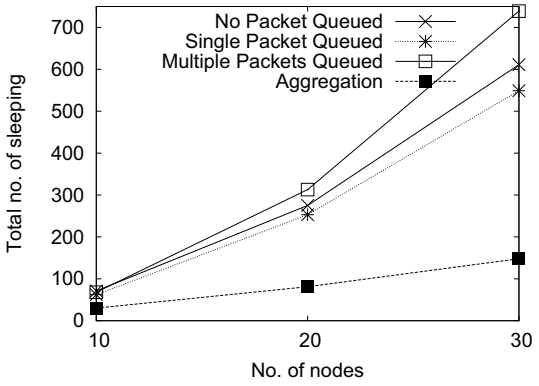

Fig. 7. No. of sleeping in one *data collection cycle*

the number of receiving is less than the number of sending for all schemes because we are counting the number of receiving performed by sensor nodes only.

Figure 6 shows the number of idle listening in the cluster. The single packet queuing scheme performs worst because initially many nodes can be polled simultaneously but later on only a few nodes can be polled because either a packet is already queued in intermediate nodes or only a few nodes can be polled due to the interference. The aggregation scheme perform best though multiple packet queuing scheme also perform better because intermediate nodes which do not interfere to each other can receive packet at any time. As shown in Figure 7, the number of sleeping does not vary much in other schemes except the aggregation scheme.

From the analysis we see that the aggregation scheme is the most energy efficient data gathering scheme for multihop polling in sensor networks. The next best is the multiple packet queuing scheme. The single packet queuing or no packet queuing schemes are not that different in energy efficiency.

It is obvious that the number of polling required for no packet, single packet or multiple packet queuing schemes will be the same whereas it will be less in the aggregation scheme as there will be less packets to forward in this scheme.

VI. Conclusions

In this paper, we study the layered sensor networks where upper layer nodes control lower layer nodes using polling scheme. Each layer will contain nodes with different computational capabilities, power supply and transmission ranges. We mainly consider the lowest layer which contains sensors and the middle layer which contains so called super nodes which form clusters of sensors. Super nodes poll sensor nodes in their clusters to collect data. Such polling saves energy of sensor nodes. We further analyze data gathering schemes: no packet queuing, single packet queuing, multiple packet queuing and data aggregation, in intermediate nodes to see which scheme performs better in saving energy in the network. The drawback of the multihop polling in layered sensor networks mentioned above is that nodes near to a super node forward packets from other nodes specially in non-aggregation data gathering scheme. Though the network life time is extended in multihop polling, nodes near to a super node deplete their energy faster. Our future work is to calculate the way to place more super nodes and alternate them as cluster heads so that sensor nodes near to the cluster head become edge nodes when the cluster head is changed.

References

[1] M. Kuorilehto, M. Hännikäinen, and T. D. Hämäläinen, "A survey of application distribution in wireless sensor networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 2005, no. 5, pp. 774–788, 2005.

[2] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, May 2005.

[3] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *Proc. IEEE INFOCOM*. IEEE, March 2005, pp. 878–890.

[4] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, "A minimum cost heterogeneous sensor network with a lifetime constraint," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 4–15, 2005.

[5] S. Rhee, D. Seetharam, and S. Liu, "Techniques for minimizing power consumption in low data-rate wireless sensor networks," in *Proc. IEEE Wireless Comm. and Networking Conf.* IEEE, 2004.

[6] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-awarewireless microsensornetworks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, March 2002.

[7] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in *in Proceedings of IEEE INFOCOM*, 2004, pp. 629–640.

[8] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. IEEE INFOCOM*. IEEE, March 2003, pp. 1713–1723.

[9] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "Ttdd: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 161–175, 2005.

[10] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the IEEE Infocom*, USC/Information Sciences Institute. New York, NY, USA: IEEE, June 2002, pp. 1567–1576. [Online]. Available: http://www.isi.edu/ johnh/PAPERS/Ye02a.html

[11] Z. Zhang, M. Ma, and Y. Yang, "Energy-efficient multihop polling in clusters of two-layered heterogeneous sensor networks," *IEEE Transactions on Computers*, vol. 57, no. 2, pp. 231–245, February 2008.

[12] A. D. Amis, R. Prakash, T. H. Vuong, D. T. Huynh, T. H. P, V. Dung, and T. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *in Proceedings of IEEE INFOCOM*, 2000, pp. 32–41.

[13] O. Chipara and G.-C. R. Chenyang Lu, "Real-time query scheduling for wireless sensor networks," in *28th IEEE International Real-Time Systems Symposium*. IEEE Computer Society, December 2007, pp. 389–399.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.