An Adaptive Directed Query Dissemination Scheme for Wireless Sensor Networks

Supriyo Chatterjea[*], Simone De Luigi[§], and Paul Havinga[*]
[*]*Department of Computer Science, University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands,*
[§]*Dipartimento di Ingegneria, Università di Ferrara Via Saragat, 1-44100 Ferrara, Italy*
[*]*{supriyo, havinga}@cs.utwente.nl,* [§]*simone.de.luigi@student.unife.it*

Abstract

This paper describes a directed query dissemination scheme, DirQ that routes queries to the appropriate source nodes based on both constant and dynamic-valued attributes such as sensor types and sensor values. Unlike certain other query dissemination schemes, location information is not essential for the operation of DirQ. DirQ uses only locally available information in order to route queries accurately. Nodes running DirQ are able to adapt autonomously to changes in network topology due to certain cross-layer features that allow it to exchange information with the underlying MAC protocol. DirQ allows nodes to autonomously control the rate of sending update messages in order to keep the routing information updated. The rate of sending updates is dependent on both the number of queries injected into the network and the rate of variation of the measured physical parameter. Our results show that DirQ spends between 45% and 55% the cost of flooding.

1. Introduction

Wireless sensor networks (WSNs) are made up of tiny devices equipped with wireless transceivers that communicate with each other through multihop networks. They have a very limited computing capability and are usually battery-operated. The sensor nodes can have a host of sensor types attached to them to measure a range of physical parameters such as temperature, light, humidity, etc.

While there are a whole spectrum of application scenarios for WSNs driving sensor network research [1], we focus on one of the key applications – Environmental monitoring. More specifically, we concentrate on how to query the environmental sensor network in an energy-efficient manner. Thus, our design describes how to accurately locate the necessary sensors to help fulfill the request specified in the query injected by a user into the network. This paper does not deal with the data extraction mechanisms that come into play once the required data sources have been identified and located.

When designing a querying scheme for WSNs, it is important to recognize the various forms of dynamism that exist within the network due to both extrinsic and intrinsic factors. An example of an extrinsic factor would be the varying rate at which queries may be injected into the network while an intrinsic factor would be the rate of variation of the physical parameter being measured or changes in network topology. Rather than designing a static querying scheme which disregards all sources of dynamism, a protocol designer should try to take advantage of these variations by allowing the querying scheme to adapt accordingly. For example, nodes should spend more energy capturing information when user demand is high or when significant changes take place in the physical parameters being measured. Conversely, nodes should try to conserve energy by operating more economically during periods of inactivity.

In this paper we describe an adaptive directed query dissemination scheme (DirQ) which tries to ensure that queries are only delivered to the relevant nodes in the network. By relevant nodes, we refer to nodes that are either able to service a particular query (source nodes) and also forwarding nodes through which a query needs to propagate in order to reach the source nodes. The DirQ scheme works in contrast to the conventional approach of flooding the entire network every time a user poses a query. In order to ensure accurate delivery, all nodes in the network need to store some information that can eventually be used for limiting the dissemination of a query only to the appropriate regions of the network. Instead of updating this

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

information on a periodic basis, DirQ adapts the update rate based on the rate at which queries are injected into the network and the rate of variation of the physical parameter being measured.

While DirQ is predominantly designed for fixed networks, it is able to cope with changes in network topology caused by the addition or death or removal of sensor nodes. This is due to the fact that DirQ incorporates certain cross-layer features which allow it to gather information from the underlying MAC protocol [2]. It also has a scalable architecture as it allows for the addition of new sensor types after deployment of sensors has been completed, i.e. a user is not required to have prior information about all the types of sensors that may be added to the network after the initial deployment.

Our simulations indicate that DirQ is fairly accurate (i.e. it suffers from an average overshoot of only 3.6%), even though the cost of DirQ hovers between 45% and 55% of flooding. They also illustrate how individual nodes are able to make autonomous decisions about the update rate to be used at any point of time based on locally available information.

In the following section we present current research related to the area of query dissemination in WSNs. Section 3 describes an example application where DirQ could be used. It helps to define the requirements of the design and the assumptions we make. We then present the actual operation of DirQ in Section 4 and an analytical evaluation in Section 5. Section 6 describes details of the Adaptive Threshold Control mechanism which allows nodes to change update rates autonomously. Simulation results are presented in Section 7 and finally, the paper is concluded in Section 8.

2. Related work

Our work focuses on directing injected *one-shot range* queries to the relevant parts of the network instead of carrying out a simple flooding of the entire network. What sets DirQ apart from the majority of the other existing schemes for sensor networks is that queries can be directed based on a *combination* of static and dynamic attributes, e.g. sensor values (dynamic), sensor types (static) and even location (static) if it is available. Thus two identical queries could follow two completely different paths during different times of the day. Moreover, since DirQ can operate based on just sensor value and sensor type information, it is *not* dependent on any underlying localization mechanism. Having location information would of course extend the capabilities of DirQ. The problem of directing queries to the appropriate section of the network has been addressed in numerous query dissemination architectures.

Directed diffusion [3], [4] is an example of a data-centric routing protocol, i.e. routing is performed based on the name of the data rather than the identity of the destination node. The Directed diffusion substrate can route queries to specific locations. However, it requires geographic information embedded in it in order to do this.

Our work is closely related to Semantic Routing Trees (SRT) presented in [5]. SRT is based on a distributed index. SRT however, only considers single attributes where as DirQ can use multiple attributes. Also, SRT is more suited for constant attributes such as location, where as DirQ is capable of working with varying attributes.

The authors in [6] present COUGAR which views the sensor network as a distributed database. However, the main problem with their approach is that a large amount of meta-data needs to be transferred to the query-processor on a periodic basis. The architecture of DirQ ensures that nodes are able to take autonomous decisions based on locally available information and thus adapt to changing network dynamics.

There are also a number of data-centric storage mechanisms designed specifically for wireless sensor networks. DCS [8] is a scheme which provides a hashing function for mapping an event name to a specific location. DIFS [7] and DIM [9] extend the data-centric approach to provide spatially distributed hierarchies of indexes to data. However, DCS, DIFS and DIM all require location information and are actually dependent on GPSR as the underlying routing protocol. Additionally, these complex indexing schemes may be too complex for sensor-class nodes as they require maintaining significant state and large tables. It should be noted that DIM for example, was tested on PDA and PC platforms. The operation of DirQ on the other hand, has been kept relatively simple as nodes only store information from their local, one-hop neighborhood.

3. Details of application

This query dissemination scheme is targeted for applications such as environmental monitoring where a large number of queries can be expected to be injected into the network. For example a network that is laid out to monitor a various physical parameters in a forest may be used by researchers at the university, and the public in general.

We have made a few assumptions about the queries posed to the network and about the network itself.

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

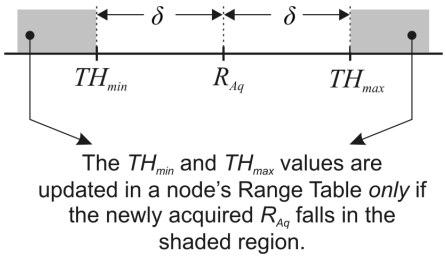

Figure 1. Local update of range table

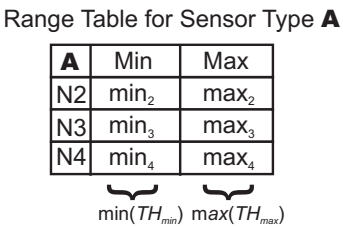

Figure 2. Picking the maximum and minimum threshold values

Firstly, due to the large variety of users, ranging from researchers to students and even the public, the network can expect to be faced with a very large number of queries. Queries can also be multi-dimensional in nature, e.g. while some users might be interested in temperature data, others might be interested in humidity. Users are expected to inject one-shot range queries into the network, e.g. "Acquire all temperature readings that are currently between 22°C and 25°C." We also assume that the server connected to the root of the sensor network (which is in-charge of injecting queries) is capable of predicting the number of queries that will be posed to the network in the next hour based on historical data. The techniques used to make this prediction can be similar to the ones used in web servers to predict web page accesses [10].

4. Operation of DirQ

The operation of DirQ has a number of phases. We first provide the reader with an overview of the phases and then describe them in greater detail in the following sections. Once the nodes have been placed in the network, a spanning tree is set up. Subsequently, the root node broadcasts an estimate message, E_{Hr}, of the number of queries that can be expected to be injected into the network over the next hour. This operation is repeated by the root once every hour. Next, a receiving node combines the estimate received with the local conditions of the physical parameter being measured (e.g. temperature, humidity, etc.) to decide upon a suitable threshold value. A node decides whether to transmit an update message when the threshold value is exceeded. An update message is made up of a tuple that consists of minimum and maximum sensor readings and is used by nodes to maintain range information of sensor readings. This range information is subsequently used to direct a query only to the relevant nodes instead of flooding the entire network, thus resulting in substantial energy savings.

4.1. Maintaining the range table

Upon receiving an E_{Hr} message from the root, a node calculates the threshold value, δ. (Section 6 describes how δ is calculated.) When a node acquires a sensor reading, R_{Aq}, it sets a minimum threshold, TH_{min} and a maximum threshold, TH_{max} where,

$$TH_{\min} = R_{Aq} - \delta \quad (1)$$

$$TH_{max} = R_{Aq} + \delta \quad (2)$$

TH_{min} and TH_{max} are put together to form a tuple which is referred to as a *Range Message*. The Range Message is then inserted into the *Range Table* of the node. If a newly acquired sensor reading falls outside the range of TH_{min} and TH_{max}, it is considered as the new R_{Aq} and new TH_{min} and TH_{max} values are computed as shown above and inserted into the Range Table. However, if the newly acquired reading falls within TH_{min} and TH_{max}, the existing range values in the Range Table are left unchanged. Thus only major temperature changes are reflected in the node's Range Table. This process that takes place within every node is illustrated in Fig. 1.

Apart from storing a node's own TH_{min} and TH_{max} values, the Range Table of a node also contains all the minimum and maximum threshold values of *all* its *immediate* child nodes (i.e. child nodes that are just one hop away). Thus the Range Table of a node that has *n* child nodes one hop away contains n+1 tuples of TH_{min} and TH_{max} (i.e. one tuple for a node's own entry and one tuple each for all n child nodes).

Every time the Range Table of a node is modified (i.e. one of the TH_{min} or TH_{max} values is changed), the node parses through the table and picks out the minimum TH_{min} and the maximum TH_{max} among all the entries, i.e. min(TH_{min}) and max(TH_{max}) respectively. This is shown in Fig. 2. These values are then compared with the previously transmitted min(TH_{min}) and max(TH_{max}) values, i.e. prev_min(TH_{min}) and prev_max(TH_{max}). If min(TH_{min}) or max(TH_{max}) differs

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

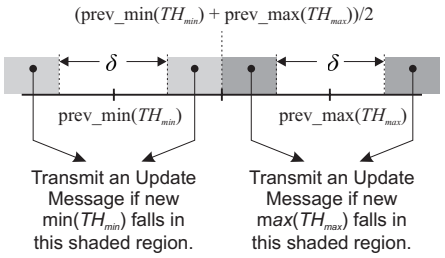

Figure 3. Transmission of an update message

from prev_min(TH_{min}) or prev_max(TH_{max}) respectively by an amount greater than δ, then the node transmits a new *Update Message* that contains the new min(TH_{min}) and max(TH_{max}) values. Thus, only significant changes in sensor readings trigger an Update Message that traverses upwards towards the root of the network which in turn ensures that the range information stored in nodes en route to the root remain updated. This helps to direct range queries more accurately to the relevant parts of the network. This process of transmitting Update Messages is illustrated in Fig. 3.

Every node can contain one or more Range Tables. Each Range Table contains range information for a single sensor type. As an example, Fig. 4 shows that Node N1 maintains three separate Range Tables for sensor types A, B and C. This is despite the fact that N1 only has sensor type C. The presence of a particular range table within a node means that the corresponding sensor type exists either within the node itself, or within one or more of the children deeper within the tree structure. This design of the Range Tables allows the network to be made up of heterogeneous nodes (i.e. different nodes can possess a different combinations of sensors). This is a great benefit over previous architectures such as TinyDB which only supports homogeneous networks.

4.2. Adapting to network dynamics

The Range Tables of DirQ are able to adapt to changes within the network topology due to dead nodes or the addition of new nodes. This is because of DirQ's cross-layer interaction with LMAC [2]. LMAC is a TDMA-based MAC protocol with a completely distributed and self-organizing scheduling algorithm. When LMAC detects that a neighboring node has died, it sends a notification to DirQ which then checks to see how the removal of the neighboring node has affected its Range Table. Any changes in the range information are then propagated up the tree. Similarly, any changes

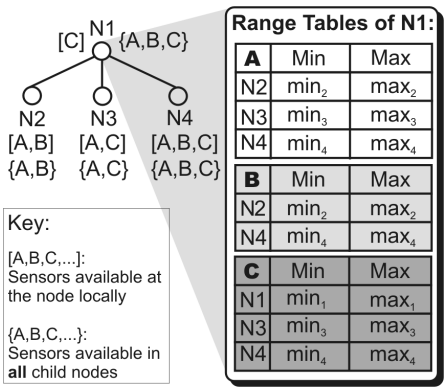

Figure 4. Support for multiple sensor types

in sensor types such as the addition or removal of sensors also propagates up the tree.

5. Analytical analysis

In this section we carry out an analytical comparison of the performance of flooding and directed query dissemination. Since the basic mechanism of DirQ is based on a tree, we perform our analysis using a *k*-nary tree with depth *d*. We also mention another reason for using a k-nary tree for the analytical comparison in Section 5. Additionally, the cost of transmitting a message is assumed to be one unit while the cost of receiving a message is also assumed to be one unit.

5.1. Cost of flooding

Before describing how we analyze the cost of flooding, we first explain how a flooding operation is carried out in order to justify the analysis later on.

When performing a flooding operation, a node transmits a message to its neighbors using a broadcast operation. We assume this behavior is followed by all nodes in the network no matter where the nodes may be located and is carried out regardless of the number of neighbors a node has. Thus even if a node does not have any other neighbor apart from the node it has received a message from, it still carries out a broadcast operation.

Therefore, regardless of the topology of the network, the transmission cost for flooding one query in a network of N nodes is N because each node sends only one MAC broadcast per query. Conversely, during a flooding operation, each node receives the query from all its neighbors. Thus the reception cost for a particular query is determined by two times the number of links in the network. The total cost of

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

flooding in a network of N nodes can then be computed as follows:

$$CF_{Total} = C_{Tx} + C_{Rx} = \text{N} + (2 \text{ x Total no. of links}) \quad (3)$$

where CF_{Total} is the total cost of flooding, C_{Tx} is the cost of transmission, and C_{Rx} is the cost of reception. Note that we use N for the cost of transmission since it is a broadcast operation and not a unicast. We only consider edges for unicast operations.

Since we are focusing on a specific case of using a k-nary tree (due to the nature of DirQ), we calculate the cost to flood one query to a *k*-nary tree with depth *d* as follows (we refer the reader to [13] for the detailed derivation):

$$CF_{Total} = \frac{3k^{d+1} - 2k - 1}{k - 1} \quad (4)$$

5.2. Cost of directed query dissemination scheme

In a flooding operation, energy is consumed disseminating a query to the entire network. In DirQ however, there are two sources of energy consumption – (i) directing a query to the relevant nodes in the network and (ii) the update mechanism to keep the minimum and maximum ranges updated. As there are two sources of energy consumption in the directed query dissemination scheme as opposed to only a single source in the flooding mechanism, the reader may be inclined to think that DirQ consumes more energy than the flooding mechanism. However, we ensure that this does not happen by adding an adaptive feature to the second source in DirQ, i.e. the update mechanism known as the *Adaptive Threshold Control* (ATC) mechanism. ATC ensures that the cost of DirQ always remains below the cost of flooding.

The cost of function of DirQ can be defined as follows:

$$CT_D = CQ_D + CU_D \quad (5)$$

where CT_D is the total cost of DirQ for a single query, CQ_D is the cost of disseminating one query to the relevant nodes and CU_D is the cost of sending an update message. It is important to realize that the cost of disseminating a query to the relevant nodes, i.e. CQ_D, is really dependent on where the nodes are located within the communication tree. For example, if the nodes relevant to the query are located close to the root, the dissemination cost will be much lower than another query whose relevant nodes might be located much deeper within the tree. This is because the deeper the relevant nodes are within the tree, the greater will be number of nodes involved in forwarding the query. The spread of the relevant nodes within the tree is also another factor that affects the cost of disseminating a particular query, i.e. the greater the spread of the relevant nodes, the greater the dissemination cost. From these examples, it is apparent that the maximum cost of disseminating a query (i.e. CQ_{Dmax}) will occur only when *all* the leaf nodes of a tree are involved in servicing a query. This is another reason why we use a *k*-nary tree (as opposed to using an incomplete tree) to compute the maximum cost of query dissemination as this ensures that the number of leaf nodes is maximum.

The computation of the worst case of DirQ occurs when both CQ_D and CU_D are maximum. CU_{Dmax} occurs when *all* the nodes in the network transmit an Update Message.

Furthermore, it should be noted that when calculating the maximum query dissemination cost, the leaf nodes do not transmit the query. This implies that only nodes within *d*-1 hops from the root will be involved in query transmission. Also, since all queries are being unicast, we assume that the cost of transmission is equal to the cost of reception. Therefore, the maximum cost of query dissemination is (we refer the reader to [13] for the detailed derivation),

$$CQ_{Dmax} = \frac{k^{d+1} + k^d - k - 1}{k - 1} \quad (6)$$

For calculating CU_{Dmax} we assume that all the nodes in the network send one update message. We introduce a variable f which indicates the frequency at which updates are received at the root. So for example, if one update is received at the root every 10 queries, then f = 0.1. Thus the maximum cost of sending an update can be calculated as follows:

$$CU_{Dmax} = \frac{2(k^{d+1} - k)}{k - 1} \quad (7)$$

Then the maximum cost of DirQ is,

$$CT_{Dmax} = CQ_{max} + f \text{ x } CU_{Dmax}$$

Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06)
0-7695-2637-3/06 $20.00 © 2006 IEEE

$$= \frac{k^{d+1} + k^d - k - 1 + 2f(k^{d+1} - k)}{k - 1} \quad (8)$$

5.3. Keeping the cost of DirQ below that of flooding

When using DirQ, it is essential to ensure that its cost is always kept below that of flooding. Thus we need to consider the *worst case* of DirQ and adapt it to make sure that it does not exceed the cost of flooding. This is performed by ensuring that $CU_{Dmax} < CF_{Total}$. Thus,

$$f_{Max} < \frac{2k^{d+1} - k^d - k}{2(k^{d+1} - k)} \quad (9)$$

where f_{Max} is the maximum number of updates that a node can transmit per query to guarantee that DirQ does not surpass the cost of flooding. So as an example, if $k = 2$ and $d = 4$, then $f_{Max}<0.76$, i.e there can be at the most 0.76 updates per query or 1 update every 1.03 queries in order for the directed query dissemination scheme to be more energy efficient than flooding. Note however, that this is for the *worst* case. This implies that for example that you could have say 2 updates per query (i.e. $f_{Max}=2$) when the query is disseminated only to a few nodes that are close to the root or all nodes in the network are not involved in sending update messages.

6. Adaptive threshold control

DirQ uses an Adaptive Threshold Control (ATC) mechanism which changes the value of δ dynamically. The chosen value of δ is dependent on the number of queries that are expected to be injected into the network over the next hour and also on the rate of change of the measured data. Due to space constraints we refer the reader to [13] which contains a detailed description of the ATC mechanism. The ATC mechanism ensures that the cost of DirQ ranges between 45% and 55% the cost of flooding.

7. Simulation results

In this section, we evaluate the performance of DirQ using simulations and compare its performance in relation to flooding. Our simulations intend to highlight the strengths of DirQ in terms of energy savings and also in terms of the accuracy of our directed dissemination scheme. The primary objectives of the simulations are to illustrate: (i) how DirQ maintains a cost below that of flooding and yet attains a high level of accuracy for the current network and environmental conditions, (ii) how DirQ allows individual nodes to make their own decisions (about what threshold level to use) autonomously based on current network and environmental conditions without the external influence of the user of the sensor network.

We initially examine the effects of accuracy and efficiency of query dissemination when using fixed thresholds (i.e. fixed values of δ). We then illustrate how the ATC mechanism curbs the total cost such that it is below flooding and yet maintains an acceptable level of accuracy. We also show how nodes adjust their threshold levels autonomously using the ATC mechanism according to the current environmental conditions.

The simulation is performed using OMNeT++ which is a discrete event simulator [11]. The results are based on a network topology of 50 nodes which includes one root where k=8 and d=10. DirQ was implemented on top of the LMAC protocol [2]. A synthetic dataset with 4 sensor types has been generated where sensor values of nodes located close to one another are spatially related. The generated sensor data is also related in the temporal dimension. Each sensor acquires a reading every time unit for a period of 20,000 time units. We refer to each time unit as an epoch [12]. Random queries which covered 20%, 40% and 60% of the nodes were generated every 20 epochs.

7.1. Using fixed threshold values

We initially perform simulations to investigate the effects when fixed threshold values were used. Threshold values are fixed at δ = 3%, 5% and 9%. For every value of δ we also examined how the percentage of nodes involved in responding to a query would affect the results.

Note that the percentage of nodes involved in a query is not directly dependent on the selectivity of the query itself. As an example, even if the selectivity of a particular query is very high (i.e. only a small number of nodes are involved) the percentage of nodes involved in answering the query is highly dependent on the location of the relevant nodes within the communication tree. If the relevant nodes are located very close to the root, the selectivity would be proportional to the number of nodes involved in the query. However, if the nodes are located deep within the network, propagating the query to the relevant nodes would be a lot more expensive due to the large number of intermediate forwarding nodes involved.

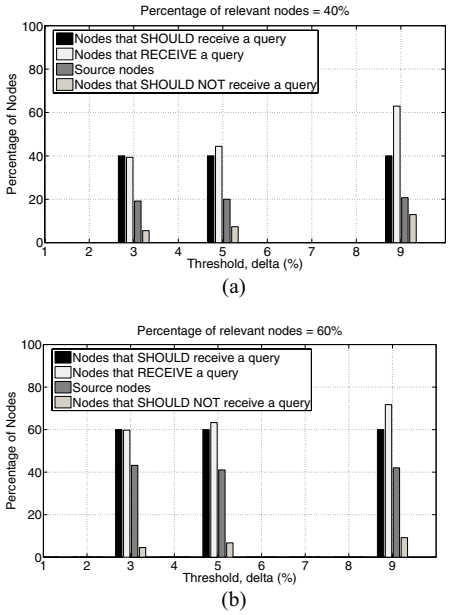

Figure 5. Effect of delta on accuracy

Thus our definition of "percentage of nodes involved in responding to a query" involves not only the relevant nodes but also the intermediate forwarding nodes.

Our first result studies how the accuracy of the directed query dissemination scheme is affected using various levels of δ. Let us first state how we define the term accuracy. When a query is injected, ideally it should be directed only to the relevant nodes in the network. However, this does not happen in reality due to the threshold levels used in the network. Since the transmission of updates is dependent on the value of δ and the rate of change of the measured parameters, the range values maintained by the nodes is not always accurate. Thus in certain instances, queries could be routed to nodes which are not relevant to a particular query. Naturally, routing queries to the non-relevant nodes, also consumes energy. We measure accuracy by computing the proportion of nodes that *are* being reached in response to a query to nodes that *should* be reached. Nodes that "should" be reached refer to both source nodes and intermediate forwarding nodes.

The results in Fig. 5 indicate that as the threshold increases (i.e. value of δ) the difference between the percentage of nodes that receive a query and the percentage of nodes that should receive the query increases. This is because as δ increases, the range information becomes more inaccurate. This effect is less pronounced as the percentage of relevant nodes increases. This is because when more nodes are involved in servicing a particular query, the probability

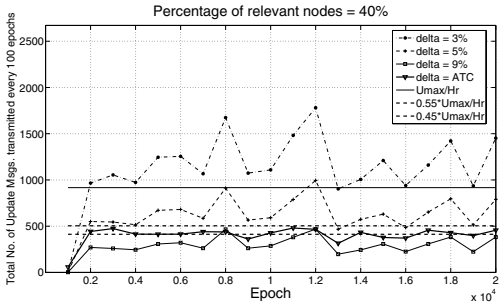

Figure 6. Effect of delta on accuracy

of routing queries to wrong nodes diminishes greatly. The value of δ clearly plays a more significant role when the percentage of relevant nodes is small.

Our results in [13] also indicate that the energy saved due to the transmission of a lesser number of updates is far greater than the cost incurred due to the transmission of incorrect queries. Thus increasing δ does have a net positive effect.

7.2. Using the adaptive threshold control

We now describe the effects of the Adaptive Threshold Control (ATC) scheme. ATC enables individual nodes to autonomously adjust the value of δ. The main drawback of using a fixed threshold is that there is a possibility that the cost of the directed dissemination scheme may exceed the cost of flooding. Fig. 6 shows the total number of Update Messages that are transmitted by all the nodes in the network every 100 epochs over a period of 20,000 epochs. It can be seen that the ATC is successfully able to adapt the transmission rate such that its cost lies around the region where the cost is roughly around 45-55% the cost of flooding. The performance remains constant for varying percentages of relevant nodes.

Fig. 5 has shown that the danger of increasing δ results in the query dissemination scheme becoming less accurate. The main idea of having the ATC is to ensure that while the number of updates transmitted is limited, the accuracy (i.e. overshoot) should not decrease significantly. Fig. 7 shows that the average overshoot when using the ATC is only around 3.6% when we consider the 20% of relevant nodes scenario.

8. Conclusion and future work

We presented a Directed Query Dissemination Scheme which tries to ensure that queries injected into the network are only sent to the relevant nodes instead of flooding. DirQ routes queries based on sensor

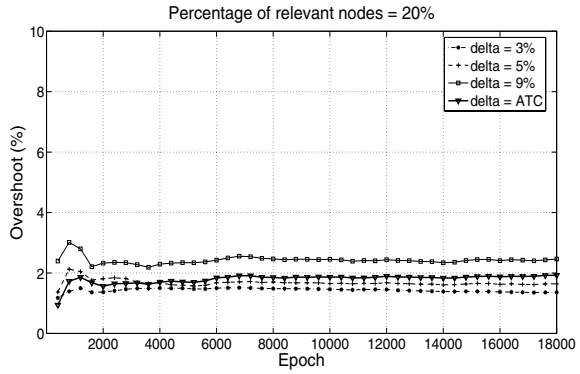

Figure 7. Overshoot using different δ and the ATC

values and sensor types. It ensures that information in routing tables is accurate by sending update messages. The rate at which update messages are transmitted is dependent on the level of usage of the network and the also the rate of variation of the physical parameter being measured by the sensors. Thus every node in the network is able to control its threshold level autonomously.

A drawback of DirQ is that we assume that nodes are able to sample sensors continuously to check if the thresholds have been exceeded. This consumes a lot of energy. We are currently developing a statistical prediction technique that can be used by DirQ to ensure that sensor sampling costs are minimized.

9. References

[1] I. F. Akyildiz et al., Wireless sensor networks: a survey, *Computer Networks*, Vol. 38, pp. 393-422, March 2002.

[2] L. van Hoesel and P. Havinga, A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In Proceedings of *1st International Workshop on Networked Sensing Systems*, 2004.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proceedings of the *ACM/IEEE International Conference on Mobile Computing and Networking*, Boston, MA, USA, Aug. 2000.ACM, pp. 56-67.

[4] D. Estrin, R. Govindan, J. Heidermann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *MobiCOM*, 1999.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In SIGMOD '03: Proceedings of the *2003 ACM SIGMOD international conference on Management of data*,pages 491–502. ACM Press, 2003.

[6] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD Record*, September 2002.

[7] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A Distributed Index for Features in Sensor Networks. In Proceedings of *1st IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, May 2003.

[8] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin,R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In Proceedings of the *First ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[9] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in Proceedings of the *First International Conference on Embedded Networked Sensor Systems*. ACM Press, 2003, pp. 63–75.

[10] E. Cohen, B. Krishnamurthy and J. Rexford, Efficient algorithms for predicting requests to web servers. In Proceedings of the *IEEE INFOCOM '99 Conference*, 1999.

[11] Omnet++ discrete event simulator, http://www.omnetpp.org.

[12] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks, In Proceedings of *VLDB*, 2004.

[13] S. Chatterjea, S. de Luigi and P. Havinga. DirQ: A Directed Query Dissemination Scheme for Wireless Sensor Networks, In Proceedings of the *IASTED International Conference on Wireless Sensor Networks,* Banff, Alberta, Canada, July 2006.