International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:6, No:6, 2012

Data Gathering Protocols for Wireless Sensor Networks

Dhinu Johnson and Gurdip Singh

Abstract—Sensor network applications are often data centric and involve collecting data from a set of sensor nodes to be delivered to various consumers. Typically, nodes in a sensor network are resource-constrained, and hence the algorithms operating in these networks must be efficient. There may be several algorithms available implementing the same service, and efficient considerations may require a sensor application to choose the best suited algorithm. In this paper, we present a systematic evaluation of a set of algorithms implementing the data gathering service. We propose a modular infrastructure for implementing such algorithms in TOSSIM with separate configurable modules for various tasks such as interest propagation, data propagation, aggregation, and path maintenance. By appropriately configuring these modules, we propose a number of data gathering algorithms, each of which incorporates a different set of heuristics for optimizing performance. We have performed comprehensive experiments to evaluate the effectiveness of these heuristics, and we present results from our experimentation efforts.

Keywords—Data Centric Protocols, Shortest Paths, Sensor networks, Message passing systems.

I. Introduction

Advances in wireless communication and electronics have led to the development of low-cost, low-power, multi-functional sensor nodes which are small sized and can communicate over short distances. Such sensing devices can enable one to create wireless sensor networks (WSN) for monitoring and tracking in different contexts [1], [2]. Many sensor applications are data centric in nature where the main task is that of collecting data from a set of sensors and delivering them to interested consumers for possible actions. Since most of the sensors are immobile and are required to operate over long time intervals without any intervention, there are severe constraints on energy consumption in sensor nodes. Hence, techniques for efficient data propagation with minimal resource requirements are needed. In traditional networks, address-centric approaches are used to route data between two end-points. These approaches excludes the possibility of several types of optimizations such as those related to combining messages based on their contents, and are therefore unsuitable for wireless sensor networks. Rather, data-centric approaches wherein data attributes are used to identify packets and to perform routing are more appropriate for sensor networks [3], [4].

In this paper, we study data-centric algorithms for data collection in sensor networks. The main tasks in a data gathering algorithm (DGA) include interest propagation, data propagation, path maintenance and data correlation. We have implemented a modular infrastructure in TOSSIM, a simulator for TinyOS based systems, to implement each of these tasks [5]. Our infrastructure is configurable in nature so that each task can be configured with heuristics for optimizations. Based on this infrastructure, we propose four algorithms for data collection. The first algorithm, DGA1, is a basic data collection algorithm with no optimization heuristics. The second algorithm, DGA2, incorporates aggregation of messages during the interest and data propagation phases. Algorithm DGA3 includes a shortest path heuristic to reduce tree cost and a greedy heuristic to incrementally add consumers to existing paths. The final algorithm, DGA4, uses a cost division technique that allows shared paths to be given preference over other paths. We also show that existing algorithms can also be cast as variants of DGA1.

We present a comprehensive experimental study of these algorithms using TOSSIM to study the impact of various optimization heuristics on the number of messages and the quality of the data collection paths constructed (tree cost). We evaluated several scenarios by varying factors such as the number of consumers and producers, network size and the aggregation limit. For example, in-network aggregation has traditionally been used during data propagation to reduce the number of messages. Our study shows that aggregation of control messages during the tree construction phase not only reduces the number of messages, but also results in the construction of lower cost trees. The paths on which aggregation is done are likely to be paths shared by different consumers. We find that aggregation results in reduced traffic and lower latency on these paths, which in turn, increases the probability of these paths being selected in the tree. We also find that although the shortest path and cost division heuristics in DGA3 and DGA4 respectively result in lowering the tree cost, they are more effective in networks with higher density of consumers. Results from our experimental studies can be used to develop rules to select the best set of heuristics for an given application and a target topology.

II. Problem Definition

We model a sensor network as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges connecting nodes which can communicate directly. Let S be a subset of V denoting the data sources and D be a set of sink nodes. We will refer to nodes not belonging to S and D as $relays$. Let $v.produce$ denote the set of data items produced by v and

Gurdip Singh is with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, 66506 USA e-mail: gurdip@ksu.edu. Dhinu Johnson was with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506. She now works for Qualcomm Corporation. Email: dhinu@ksu.edu.

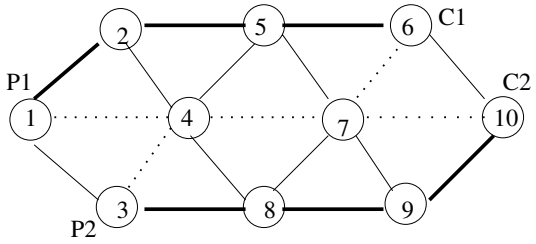

Fig. 1. Examples of data gathering subgraphs

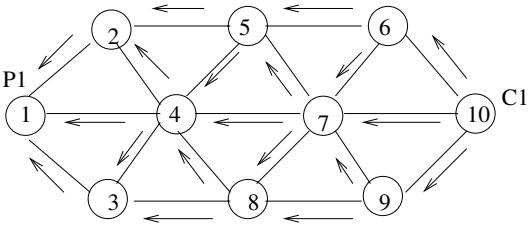

Fig. 2. Illustration of the Interest propagation phase

$v.consume$ denote the set of items v is interested in. We define the cost of a subgraph of V as the total number of edges in the subgraph. The goal of a data gathering algorithm (DGA) is to find a minimal cost subgraph of G such that for each $v \in D$, the subgraph consists of a tree, $tree_v$, consisting of v and at least one source node for each data item in $v.consume$. Fig. 1 shows a network with nodes 6 and 10 as sinks and nodes 1 and 3 are sources. Node 6 is interested in data produced by 1 and node 10 is interested in data produced by 3. The bold edges shows a possible subgraph for delivering data to 6 and 10. Although this subgraph consists of a shortest path tree for each sink node, its overall cost (which is 6) is not minimal. The dotted lines in Fig. 1 shows the minimal cost subgraph with cost 5. The problem of constructing a minimal cost subgraph is reducible to the Steiner Tree problem [6] which is NP-hard. Hence, heuristics are needed to obtain suboptimal solutions to the data aggregation problem.

III. Infrastructure for Data Propagation Algorithms

To design and evaluate DGAs, we have implemented a modular infrastructure in TOSSIM, a simulation environment for TinyOS-based sensor network systems [5]. The execution of a data collection algorithm can be decomposed into several phases (which are inspired by the phases described in the diffusion algorithms in [4]). The first phase involves the sink nodes sending *interest* messages to indicate their interest in specific data items and to locate the corresponding source nodes. When a source node receives an interest message, it starts transmitting *data* messages. These are sent along the paths formed during the propagation of the interest messages. A sink node, on receiving a data message, may choose to *reinforce* one or more of these paths by transmitting a *reinforcement* message. This execution pattern allows us to decompose an algorithm into interest propagation, data propagation and reinforcement phases (although we must note that these phases are not entirely sequential). We have implemented a configurable module for each phase which can be configured with various policies to optimize performance. In the following sections, we first describe the basic phases of a DGA. Subsequently, we will discuss how variations of the algorithms can be obtained by configuring each module.

A. Interest Propagation phase

The algorithm is initiated by a sink node by sending interest messages to potential source nodes. Since a sink may not know the location of source nodes, it must propagate interest messages throughout the network. The arrows in Fig. 2 shows a possible propagation of interest messages initiated by node 10. As an interest message diffuses through the network, it accumulates path cost (hop count) in a cost field. An *interest table* is maintained at each node which has at most one entry per data type for each sink. Each entry in the table is a tuple

$\langle$ sink, data_name, cost, neighbor, timestamp, reinforced $\rangle$

The fields $sink$ and $data\_name$ uniquely identify each entry, and the $neighbor$ denotes the node from which the least cost interest message has been received so far (other fields will be explained later). Consider the case in Fig. 2 in which the first interest message received by node 4 is from node 5. In this case, a new entry is created in the interest table with sink = 10, data_name = X, and neighbor = 5, where X is the data item requested by 10. The interest message is then forwarded to the neighbors. Subsequent interest messages received by 4 with <source = 10, data = X> from other neighbors do not result in the creation of a new entry. Thus, in the default configuration of the interest propagation module, an interest message is forwarded only once for each <sink, data_item> pair. The interest table is periodically refreshed to remove expired entries (using the $timestamp$ field in the table).

B. Data Propagation phase

This section describes the operation of the data propagation phase which starts when a source node receives an interest message. If the interest message is the first request for a <sink, data item> pair or the interest message has a lower cost than the one received earlier, then the source node responds with a data message. For example, in 2, if node 1 receives the first interest message from node 2 (with cost 3), then it will reply with a data message. If node 1 subsequently receives an interest message from node 3 (with cost 3), then the message is ignored as it does not have a lower cost. However, when the interest message from node 4 with cost 2 is received, the source will reply with a data message. Thereafter, the source will send data messages periodically, tagged with increasing sequence numbers, to each neighbor from which it has received a reinforcement message. As data messages propagates through the network, they also accumulate the hop count.

A *data table* is maintained at each relay whose format is similar to the interest table except that it also contains a sequence number. When a relay receives a data message for item d from source s, it checks whether an entry for that data

item exists in the data table. If it is the first such message, then a new entry is created in the table and the message is forwarded to all neighbors with matching entries in the interest table. If an entry already exists but with a smaller sequence number, then the sequence number is updated and the message is forwarded to only those neighbors whose entries have already been reinforced in its interest table. Otherwise, the data message is discarded. Thus, the first data message is sent along all potential paths with matching interests, whereas the subsequent data messages are sent only along selected (reinforced) paths.

C. Path Reinforcement

The path reinforcement phase starts when a sink receives a data message in response to its interest messages. In this case, the sink selects one of its neighbors from which it has received a data message to send a reinforcement message. In the default case, the node selected is the first neighbor to deliver the data message. At each relay node, on reception of a reinforcement message, its parameters are compared with entries in the interest table. If a matching entry is found, then the entry is updated to indicate that it is *reinforced* and its expiry time is extended. The reinforcement message is then forwarded to the node from which the least cost data was received from the source. On receiving a reinforcement message, a source node updates its interest table in a similar fashion. Thus, the reinforcement messages traverse the preferred path from the sink to the source, reinforcing the entries on the way. Subsequently, the source periodically sends out *data* to neighbors that delivered *reinforcements*. As long as there are reinforced entries in the interest tables on relay nodes along a data path, the flow of data continues. If a sink does not receive data it requested after a certain span of time, it may recommence the *interest cycle* by propagating interest messages.

We have described the basic functioning of the DGA phases. There are additional mechanisms and data structures such as those related to multiple data items, timers, refresh cycle and interest cycle needed in the various phases. Overall, our algorithm is organized in terms of three components: Sink Node Algorithm, Relay Node Algorithm and Source Node Algorithm. The Sink Node Algorithm describes the behavior of the sink nodes. A sink node maintains two timers, one for $interest$ cycles and the another for $refresh$ cycles. When the sink starts up, it starts its interest timer and enters the interest propagation phase when the timer times out. As discussed above, this phase involves the sink node broadcasting interest phases to locate potential sources. When the first data message in response to the interest message is received, a sink node starts its reinforcement timer. Again, as described above. the reinforcement message (and phase) is started when this timer times out. A relay node maintains an interest table to cache interest messages, and a data table to cache data mesages. When a data or an interest message is received, the appropriate table is consulted (a new entry is made is one does not exist) to generate a response as discussed in the description of the phases above. Every source node maintains an interest table similar to a relay node. When a source node receives an interest message, it creates a new entry if one does not exist. Otherwise, it updates the entry if the new message provides a lower cost. Source nodes use timers to periodically send data messages to reinforced data requests. This timer is started when the node receives the first interest message. When this timer expires, data is sent for requests which have been reinforced. One such aspect is the *sink correlation-logic*. Although a sink may be interested in more than one data item, it may require all of them (And-correlation), any one of them (Or-correlation) or some combinations of the data items. We have added the mechanisms necessary to implement different types of sink correlation.

IV. Variants of Data Gathering Algorithms

In this section, we propose a set of data collection algorithms. Our basic data collection algorithm, DGA1, corresponds to the default configuration for each of the modules discussed in the previous section. Thus, DGA1 does not include any optimizations. The variations discussed in the following involve incorporation of heuristics in different phases to reduce messages and tree cost.

A. Algorithm DGA2

This algorithm studies the impact of aggregation on the performance of data collection protocols. Interest and data aggregation is possible during the interest and data propagation phases respectively. Aggregation can be done at the sources, sinks or relay nodes. For example, several sink nodes may be in the interest propagation phase at the same time. As a result, a relay node may receive several interest messages to be forwarded, which can be aggregated into a single message. We refer to this a *relay aggregation*. The amount of relay aggregation, however, is limited by the message size allowed by the underlying protocol stack. *Sink-aggregation* can be done at the sink nodes. If a sink node is interested in several data items, then the interest messages for each of the data items can be aggregated into a single message and propagated together (rather than independent propagation of the interest messages). Similarly, *data aggregation* can be done at the relays or at the source nodes.

Algorithm DGA2 is a modified version of DGA1 with support for both data and interest aggregation. Clearly, aggregation is expected to reduce the number of messages transmitted. However, it may also lower the tree cost. For example, consider two source nodes s_1 and s_2 interested in data items being produced by nodes in V_1 and V_2 respectively, where V_1 and V_2 may overlap. In traditional aggregation techniques, the trees to deliver data items to s_1 and s_2 are first formed, and then in-network aggregation is performed on edges common to both trees. DGA3, however, performs aggregation of the interest messages during the tree formation stage. This allows us to exploit the overlap between V_1 and V_2 to construct trees with more shared paths, thereby lowering the overall tree cost. In particular, our experimentation results show that with multiple consumers, aggregation reduces traffic along shared paths resulting in faster interest and data propagation along

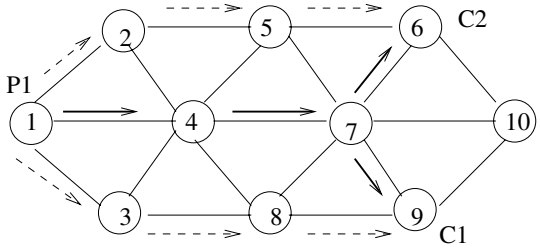

Fig. 3. Illustration of the Cost division heuristic

these paths, which in turn increases the probability of such paths being reinforced and selected to be part of the tree.

B. Algorithm DGA3

Algorithm DGA3 incorporates the following two heuristics in addition to interest and data aggregation:

• *Shortest path heuristic*: This heuristic introduces rules for more aggressive propagation of interest, data and reinforcement messages to lower the tree cost. Hence, it lowers the tree cost at the expenses of number of messages. As an example of one such rule, in Fig. 2 where node 10 initiates the interest propagation phase, node 4 may receive 10's interest message via multiple neighbors. Consider the case where 4 receives the interest message from 7 after receiving the one from 5. In both DGA1 and DGA2, the message from 7 is considered a duplicate as a matching entry in the table is already present, and is discarded. However, since this interest message has a lower cost (shorter distance to 10), updating the interest table entry will lead to a shorter path. In DGA3, node 4 updates the interest table to reflect 7 as the new neighbor for sink 10, and then forwards this message. Thus, an interest message for the same <sink,data_item> pair may be forwarded more than once in DGA3. However, each subsequent propagation contains a lower hop count.

• *Greedy Heuristics*: This heuristic is used to incrementally add consumers to already established paths between existing producers and consumers. For example, consider the case in Fig. 2 where node 9 starts an interest propagation phase for the same data item as 10. If 4 receives 9's interest message via 8, and 4 has already entered the data propagation phase with respect to 10, then a matching entry for the data required by 9 will be found in node 4's data table. In this case, in DGA3, we do not propagate the interest message further and 4 simply responds with the data message itself.

C. Algorithm DGA4

DGA4 employs a *cost-division heuristic* which reduces the tree cost by giving preference to shared paths in the presence of multiple consumers and producers. We will explain this heuristic via an example. Consider the scenario in Fig. 3 where both consumers C1 and C2 need data produced by P1. The dashed arrows show a possible DGA graph with shortest paths for each consumer. However, the solid arrows present an alternative subgraph which has the same lengths for the shortest paths between sources and sinks but has an lower overall cost due to a shared path. We have incorporated a heuristic which allows nodes to divide the cost of a shared path when computing the distances. In Fig. 3, when node 7 sends the first data message to C1 and C2 (which is a feedback on the quality of the path), it divides the cost of the path accumulated from the sink node to 7 by the number of interested consumers. In this case, the cost is 2, which is divided among 2 consumers. Hence, a cost of 1 is propagated to each consumer. Thus, C1 will compute the cost of sending via 7 as 2, which is lower than the cost of the path via 8. A complete description of DGA4 with additional details such as how to handle cost-division when new consumers are added to existing paths are given in the full paper.

V. Discussion and related work

Many existing algorithms can be cast as variants in our infrastructure, and new algorithms can be developed by incorporating new optimization heuristics. For example, we are currently studying heuristics which optimize the tree cost when the rates at which data is produced and consumed by various nodes is taken into account. Diffusion algorithms for data dissemination which perform in-network data aggregation have been studied in [3], [4], [7], [8], For example, a two-phase pull diffusion algorithm proposed in [4] can be cast as a variant of DGA1. The main difference is that in the two-phase pull algorithm, a gradient (interest table entry) is maintained between each pair of neighbors for each data item with a direction associated with it. This is different from our basic algorithm which maintains a single entry for each pair of data item and source. [8] presented a comparative study of a set of data collection algorithms. However, the focus was on comparing the data push-based technique to the data pull-based technique. The focus of this paper is on a different set of techniques for data collection. [9] studies the problem of data collection in wireless sensor networks. However, their main focus is to study the delay rates (ratio of data size and the delay) and capacity of data collection protocols and provding theorectical upper and lower bounds. An alternative strategy in which a sink moves in a network and collect data from sensors has been studied in [10]–[12].

VI. Performance evaluation results

We evaluated the algorithms using TOSSIM. The evaluation was done starting with a square grid of size 6x6 and increasing the dimension in steps of 2 until size was 20x20 (total of 8 cases). Source and sinks are placed in the test topologies at opposite ends of the square grid. We also varied the number of consumers and producers. We will use xPyC to refer to the case with x producers and y consumers. With And-correlation, we assumed that each of the producers is producing a distinct data item, and all consumers are interested in all data items. For example in the case of 4P4C each of the four sources generate four different data items and each of the four consumers requests all of the four data types. The possible values of x and y considered were 1, 2 and 4 (total of 9 cases). Hence, 72 cases (8*9) were tested and results were averaged over several runs for each case. For each case, we

measured the number of interest messages propagated and the total cost of all trees constructed.

For Or-correlation, we assumed that all producers are producing the same data item, and the consumers are interested in receiving data from any one of them. We also assumed that the underlying protocol stack imposes an aggregation limit of 4 (that is, at most 4 interest messages can be aggregated into a single message).

A. And-correlation scenarios

1) Evaluation of DGA2: The performance of DGA2 which supports interest and data aggregation is evaluated in this section. For example, consider a network where a sink requires two types of data, say temperature and humidity readings and a source in the network generates both of these data. When there is no network aggregation, each request by the sink will be treated separately and may result in two independent paths between set up. With network aggregation, the paths can be shared to lower the cost. Fig. 4 and Fig. 5 show the performance in terms of the number of interest messages sent in the different algorithms. Fig. 4 shows

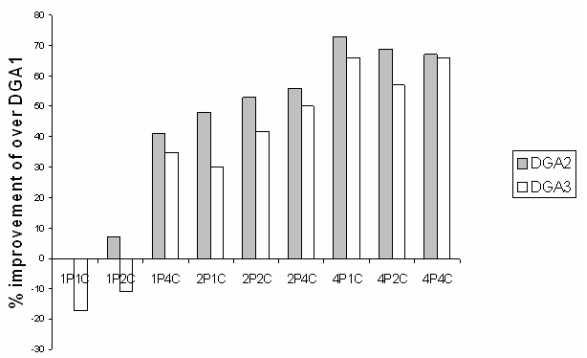

Fig. 4. Comparison of number of messages

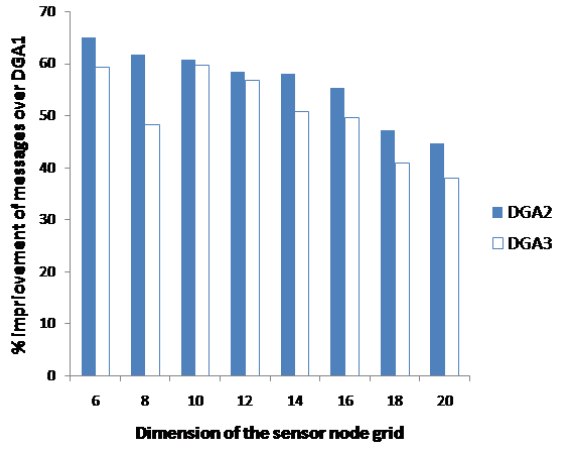

Fig. 5. Evaluation with respect to network size

the percentage improvement of each algorithm over DGA1. For each algorithm and each case (xPyC), the percentage improvement for different network sizes was computed and the average is presented in Fig. 4. As can be seen in Fig. 4, DGA2 outperforms DGA1 in all cases as the number of consumers and producers are varied. Note that for case 1P1C, there is no scope for aggregation, and hence both DGA1 and DGA2 behave the same. As discussed earlier, interest aggregation can take place at the sink nodes (where the interest messages for different data types are aggregated) and at the relay nodes (where aggregation happens when messages arrive simultaneously to be forwarded over common links). For cases 2P1C and 4P1C, we see significant improvements of 48.33% and 73% respectively, which is a result of sink aggregation (no relay aggregation happens with one consumer). As we increase the number of consumers, cases 2P2C and 2P4C show an improvement of 53.72% and 56.32% respectively. This incremental improvement is due to aggregation at the relay nodes. In fact, for 4P2C and 4P4C, a drop from 73% (for 4P1C) to 69.44% and 67% respectively was observed. In these cases, since the aggregation is limited to 4 messages, no aggregation at relay nodes is possible. Fig. 5 shows that the relative performance of DGA2 over DGA1 as the network size increases for the case 2P4C. As the number of consumers is kept constant and the network size is increased, there is a reduced probability of relay aggregation and therefore, the relative performance of DGA2 over DGA1 drops with network size. A similar trend is observed in all other cases.

Fig. 6 shows the performance improvements in tree cost for different algorithms (averaged over different network sizes). As can be seen, DGA2 results in lower cost trees as compared to DGA1 (except for test case 1P1C where no interest aggregation is possible). This improvement is primarily due to reduction in the traffic along shared paths on which aggregated interest and data messages are sent. In particular, in the single consumer cases, sink aggregation results in a significant improvement. Since interest messages for different data items for a sink v are aggregated at the sink itself, if a sub-path is selected for one data item $d1$ in $v.consume$ and it is also a possible sub-path for another data item $d2$ in $v.consume$, then that sub-path is also likely to be selected for $d2$ due to simultaneous flow of interest messages. However, as the number of consumers increase, we see a relatively lower improvement due to the interest aggregation limit of 4. When we repeated the experiments for the cases with 4 consumers by increasing the aggregation limit to 8, we observed a continuous increase in the performance of DGA2 over DGA1 as the number of consumers increase.

2) Evaluation of DGA3: In this section, we report the results of our experiments evaluating the performance of DGA3. As can be seen in Fig. 4 and Fig. 5, DGA3 uses more messages as compared to DGA2 as it may propagate additional interest messages. The comparison of DGA3 and DGA1 is more interesting. There are three factors impacting the number of messages when comparing DGA1 and DGA3: (a) increase due to propagation of lower cost interest messages, (b) reduction due to aggregation, and (c) reduction due to the greedy heuristics. For cases 1P1C and 1P2C (see Fig. 4), the number of interest messages generated by DGA3 is higher (rel-

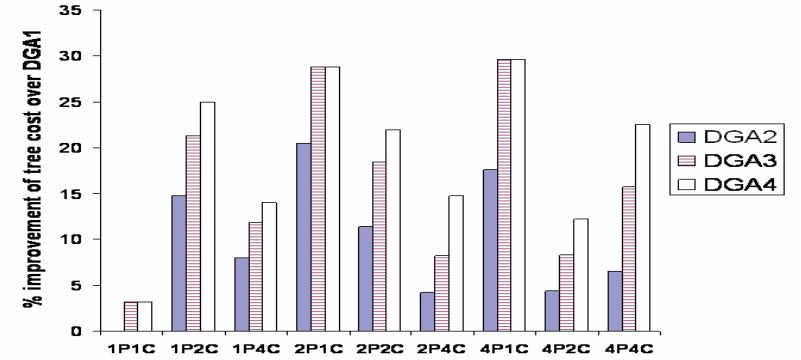

Fig. 6. Comparison of tree cost

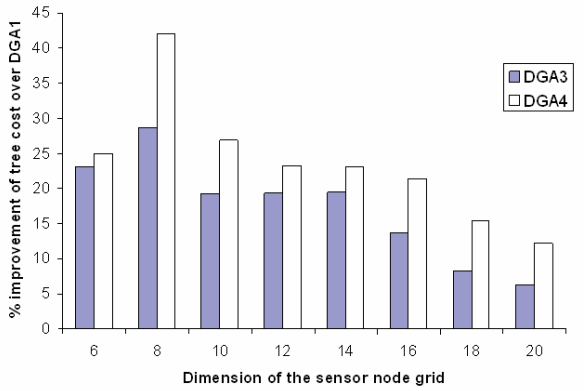

Fig. 7. Tree cost comparison for case 4P4C

ative improvement of -17% and -12.58% respectively). Here, we find that factor (a) is dominant as not much aggregation is possible. The results for other cases show the opposite: DGA3 outperforms DGA1 (for example, there is a 35% improvement for 1P4C). In these cases, factor (b) dominates (a). However, for the test cases 4P1C, 4P2C and 4P4C, the improvements observed are 66.91%, 57.69% and 66.14% respectively. In these cases, the aggregation reaches the saturation level as the aggregation limit is 4. Here, factor (b) reaches its limit and we see a drop in relative performance from 4P1C to 4P2C. However, the greedy heuristic factor (c) offsets this drop when number of consumers is increased to 4 in 4P4C.

As shown in Fig. 6, DGA3 outperforms both DGA1 and DGA2 with respect to tree cost criteria. This was expected as DGA3 attempts to reduce tree cost at the expense of more messages. As the network size is increased by keeping the number of consumers constant, the density of consumers decreases. This decreases the probability of receiving messages over longer paths before those on shorter paths increases. Hence, the relative advantage of DGA3 drops. This is reflected in Fig. 7 which shows the tree cost for the case 4P4C for different number of nodes. As can be seen, as the network size increases, the percentage improvement of DGA3 over DGA1 decreases.

3) Evaluation of DGA4: We did not evaluate DGA4 for test cases with one sink (which includes the cases 1P1C, 2P1C and 4P1C) as there is no room for cost division heuristic with a single consumer. Furthermore, there is no change in the number of interest messages sent as compared to DGA3. Hence, we only evaluated the tree cost and not the number of messages. As can be seen in Fig. 6, DGA4 results in lower tree cost as compared to all other algorithms. We also find that DGA4 is more effective when the density of consumers is high.

For example, as the number of consumers increases, the relative advantage of DGA4 over DGA3 in same sized network improves (as more consumers imply more possibility of shared paths). For a network with grid size of 20x20, DGA4 outperformed DGA3 by 2.85% for 2 consumers, and by 6.39% for 4 consumers. Similarly, as the number of consumers is kept constant and the network size increased, the possibility of shared paths reduces, and hence, DGA4 becomes less effective. This is reflected in Fig. 7 which shows a relative decrease in the improvement of DGA4 over DGA1 as the network size increases.

4) Comparison of Steiner tree cost: For each of the test cases, we also computed the optimal tree cost (which is the cost of the Steiner tree for the graph). The percentage deviation from the Steiner tree cost for the various algorithms is shown in Fig. 8. Note that lower the deviation, the better is the performance. It can be observed that the deviation from the Steiner tree cost increases as the number of producers and consumers is increased. This reflects the increased difficulty of finding an optimal solution in more complex configurations. It can also be observed that for each test case, the tree cost deviation decreases as we move from DGA1 to DGA4.

B. Sinks with Or-correlation logic

We also experimented with Or-correlation logic at sinks. In this case, the sink is interested in getting data from any one of the sources (thus, a path has to be established for each sink with respect to one source only). Test cases with one source or one sink are not performed as they can be reduced to cases already analyzed for And-correlation. The results for the test cases for Or-correlation are shown in Fig. 9. The results showed similar trends as observed in the And correlation case.

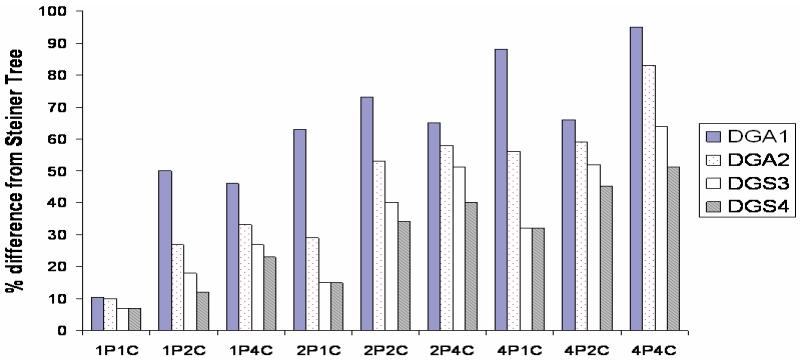

Fig. 8. Difference from the Steiner tree cost

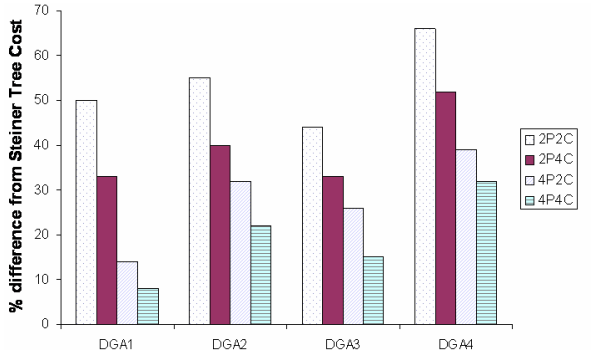

Fig. 9. Tree cost for Or-correlation

VII. Conclusion

We have presented a configurable infrastructure for implementation of data collection algorithms. We proposed and implemented four algorithms for data collection in this infrastructure. We performed a comprehensive set of experiments to evaluate their performance by varying several factors such as network size, number of producers and consumers and aggregation limit. We presented several conclusions from our experiments. For example, we showed that aggregation not only reduces the number of messages, but also lowers the tree cost when aggregation is performed during the tree construction phase. We also showed that various heuristics such as the shortest path and the cost-division heuristics result in lower cost trees and are more effective when the density of consumers is high.

Acknowledgement

This work was supported by NSF grants CNS0615337 and CNS0551626 and DARPA contract F33615-00-C-3044.

References

[1] Y. Snakarasubramaniam, I. F. Akayildiz, W. Su, and E. Caryirci. A survey of sensor networks. *IEEE Communications*, 40(8):102–114, August 2002.

[2] Chee-Yee Chong and S.P. Kumar. Sensor networks: evolution, opportunities and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[3] B. Krishnamachari, D. Estrin, and S. Wicker. Modeling data-centric routing in wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, 2002.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, 2000.

[5] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, 2003.

[6] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

[7] B. Krishnamachari and J. Heidemann. Application-specific modelling of information routing in sensor networks. In *Proceedings of the Workshop on Multiop Wireless Networks in conjunction with the Itnernational Performance Computing and Communications Conference*, 2004.

[8] J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, 2003.

[9] Siyuan Chen, Yu Wang, Xiang-Yang Li, and Xinghua Shi. Order-optimal data collection in wireless sensor networks: Delay and capacity. In *IEEE Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2009.

[10] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Sink mobility protocols for data collection in wireless sensor networks. In *Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*, 2006.

[11] G. Anastasi, M. Conti, and M. Di Francesco. Reliable and energy-efficient data collection in sparse sensor networks with mobile elements. *Performance Evaluation*, 66(12):791–810, December 2009.

[12] L. He, J. Pan, and J. Xu. Reducing data collection latency in wireless sensor networks with mobile elements. In *Proceedings of the International Workshop on Sensor, Actuator and Robot Networks in conjunction with IEEE INFOCOM*, 2011.