World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:1, 2009

Tree-on-DAG for Data Aggregation in Sensor Networks

Prakash G L, Thejaswini M, S H Manjula, K R Venugopal, L M Patnaik

Abstract—Computing and maintaining network structures for efficient data aggregation incurs high overhead for dynamic events where the set of nodes sensing an event changes with time. Moreover, structured approaches are sensitive to the waiting time that is used by nodes to wait for packets from their children before forwarding the packet to the sink. An optimal routing and data aggregation scheme for wireless sensor networks is proposed in this paper. We propose Tree on DAG (ToD), a semistructured approach that uses Dynamic Forwarding on an implicitly constructed structure composed of multiple shortest path trees to support network scalability. The key principle behind ToD is that adjacent nodes in a graph will have low stretch in one of these trees in ToD, thus resulting in early aggregation of packets. Based on simulations on a 2,000-node Mica2-based network, we conclude that efficient aggregation in large-scale networks can be achieved by our semistructured approach.

Keywords—Aggregation, Packet Merging, Query Processing.

I. Introduction

THE main operation of a wireless sensor network (WSN) is to monitor the physical environment, process the sensed information, and deliver the results to some specic sink nodes. Sensor nodes are normally powered by batteries with limited energy resource. Therefore, the primary challenge for this energy-constrained system is to design energy-efcient protocols to maximize the lifetime of the network [1]. Since radio transmission is the primary source of power consumption, the design of communication protocols for topology management, transmission power control, and energy-efcient routing.

The basic idea is to route the packet through the minimum energy paths so as to minimize the overall energy consumption for delivering the packet from the source to the destination. The drawback of this approach is that it tends to overwhelm the nodes on the minimum energy path, which is undesirable for sensor networks since all sensor nodes are collaborating for a common mission and the duties of failed nodes may not be taken by other nodes. forwarding unaggregated packets increases with the scale of the network. To benefit from the strengths of structured and structureless approaches, we propose a semistructured approach.

In order to use the sensor network efciently, the data these sensors produce must be properly accessed and eventually propagated to the end user. One way this is done is through the organization of the sensor nodes into a network, which is treated and queried as a distributed sensor database. Such a sensor network allows for paths to be created from the root of the network to each and every sensor node in the network. However, sensor nodes are limited devices in terms of energy usage, node failures and lossy communication.

In order to alleviate these limitations, especially of power constraints and fault tolerance, several schemes have been developed. One scheme that has shown much promise is to organize the nodes into a tree and synchronize the sending and receiving of packets from children to parents in that tree. Doing this allows for in-network aggregation to occur, which has been shown to lower the amount of energy used in sensor networks, thus extending their lifetime and usefulness.

While query propagation and tree organization techniques have been shown to lower energy usage, they suffer in several respects, most importantly in the ability to deal with node crashes and failures. When one node fails all the nodes in its subtree will also be cut off from the network (due to the parent/child relationship) until the network can be reorganized. The problem is even more evident for nodes close to the root of the network, where a single node failure could cause a substantial portion of the tree to get isolated, dramatically decreasing the effectiveness of the data produced for the end user.

$challenges$: The main challenge in designing such a protocol is to determine the packet forwarding strategy in absence of a preconstructed global structure to achieve early aggregation. Our approach uses a structureless technique locally, followed by Dynamic Forwarding on Tree on DAG (ToD), an implicitly constructed packet forwarding structure to support network scalability. After performing local aggregation, nodes dynamically decide the forwarding tree based on the location of the sources. The key principle behind ToD is that adjacent nodes in a graph will have low stretch in at least one of these trees in ToD, thus resulting in early aggregation of packets.

$contributions$: This paper makes the following contributions:

- We propose an efficient and scalable data aggregation mechanism that can achieve early aggregation without

Prakash G L, Thejaswini M, S H Manjula and K R Venugopal are with the Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore 560 001, e-mail: glprakash@yahoo.com.

L M Patnaik is a Vice Chancellor, Defence Institute of Advanced Technology(Deemed University), Pune, India.

incurring overhead of constructing a structure.
- We implement the ToD on TinyOS and compare it against other approaches on a 105-node sensor network.
- For studying the scalability aspects of our approach, we implement ToD in the ns2 simulator and study its performance in networks of up to 2,000 nodes.

$Organization$: The organization of this paper is as follows: Section 2 presents related work. Section 3 presents the problem definition. Section 4 presents the design. The performance evaluation of the protocols using experiments and simulations is presented in Section 5. Finally, Section 6 concludes this paper.

II. RELATED WORK

Data aggregation has been an active research area in sensor networks for its ability to reduce energy consumption. Some works focus on how to aggregate data from different nodes [**?**], some focus on how to construct and maintain a structure to facilitate data aggregation [2], [3] and some focus on how to efficiently compress and aggregate data by taking the correlation of data into consideration [4]. As our work focuses on how to facilitate data aggregation without incurring the overhead of constructing a structure, we briefly describe the structure-based approaches in current research.

In [5], the authors propose an aggregation tree construc- tion algorithm to simultaneously approximate the optimum trees for all nondecreasing and concave aggregation functions. The algorithm uses a simple min-cost perfect matching to construct the tree. Other works, such as Steiner Minimum Tree (SMT) and Multiple Shared Tree (MST) for multicast algorithms which can be used in data aggregation , build a structure in advance for data aggregation. In addition to their complexity and overhead, they are only suitable for networks where the sources are known in advance. Therefore, they are not suitable for networks with mobile events.

In addition, fixed tree structures also have the long stretch problem. A stretch of two nodes u and v in a tree T on a graph G is the ratio between the distance from node u to v in T and their distance in G. Long stretch implies that packets from adjacent nodes have to be forwarded many hops away before aggregation. This problem has been studied as Minimum Stretch Spanning Tree (MSST) and Minimum Average Stretch Spanning Tree (MAST) [6].

Data Aware Anycast (DAA) [7] is the first proposed structureless data aggregation protocol that can achieve high aggregation without incurring the overhead of structure approaches. DAA uses anycast to forward packets to one-hop neighbors that have packets for aggregation. It can efficiently aggregate packets near the sources and effectively reduce the number of transmissions. However, it does not guarantee the aggregation of all packets. As the network grows, the cost of forwarding packets that were unable to be aggregated will negate the benefit of energy saving resulted from eliminating

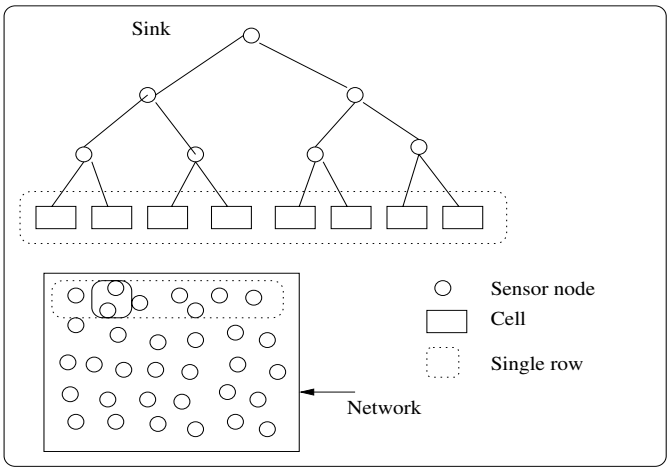

Fig. 1: The ToD construction from one rows point of view.

the control overhead. In order to get benefit from structureless approaches even in large networks, scalability has to be considered in the design of the aggregation protocol. In this paper, we propose a scalable semistructured protocol, ToD, that can achieve efficient aggregation even in large networks.

To support queries over sensor networks, the distributed data over sensor nodes need to be processed. This poses up an implicit requirement for aggregating the data. Optimal Data Aggregation is an NP-HARD problem [8].

TAG and COUGAR are tightly coupled with the underlying aggregation schemes. [9] proposes a Query Agent that provides application independent query interface and an API support to map the user specied queries to lower level semantics corresponding to underlying routing and aggregating protocols. It supports different communication models anycast , unicast , multicast and broadcast. Query agent will support a wide variety of routing and aggregation protocols selecting the best combination based on the type of the query.

An enhancement of the same is CLUDDA [10], where the authors propose to use clustering for more efcient use of resources and extend the query format to support dynamic aggregation. The idea of clustering is to prevent the ooding during interest propagation, which in this case is limited to the cluster heads. The regular nodes do not participate unless they can support some request. The revised format of interest allows the node to handle unfamiliar queries and formation of data aggregation. The cluster heads act as dynamic data aggregation points thereby ensuring aggregation as close to the source as possible.

III. PROBLEM DEFINITION

Cosider the network as shown in figure 1, the network is divided into cells. These cells are grouped into clusters, called

F-clusters (first-level clusters). The size of the F-clusters must be large enough to cover the cells an event can span, which is two when we only consider 1D cells in the network. All nodes in F-clusters send their packets to their cluster-heads, called F-aggregators. Nodes in the F-cluster can be multiple hops away from the F-aggregator.

The goal of our protocol is to achieve aggregation of data near the sources without explicitly constructing a structure for mobile event scenarios. Aggregating packets near the sources is critical for reducing the number of transmissions. Aggregating without using an explicit structure reduces the overhead of construction and maintenance of the structure. In this section, we propose a highly scalable approach that is suitable for large sensor networks.

In DAA [7], packets were destined to the sink when no further aggregation can be achieved locally. This incurs higher transmission cost when the distance from sources to the sink is longer. To remedy this problem, we forward packets on a structure when no further aggregation can be achieved instead of forwarding them to the sink directly.

IV. DESIGN

A. Network Model

The construction of F-Tree, S-Tree, and ToD is shown in figure 2. Leaf nodes are cells. Pairs of neighbor cells define F-clusters. Each F-cluster has an F-aggregator, and F-aggregators form the F-Tree. Each pair of adjacent cells not in the same F-cluster form an S-cluster. Each S-cluster has an S-aggregator, and S-aggregators form the S-Tree. Nodes on the network boundary do not need to be in any S-cluster. ToD is created by connecting each F-aggregator to two S-aggregators of S-clusters which its F-cluster overlaps with. F-aggregator in ToD uses Dynamic Forwarding to forward packets to the root, or through an S-aggregator in the S-Tree based on where the packets come from. S-Tree creates a Directed Acyclic Graph, which we refer to as the ToD.

In addition to the F-clusters, we create the second type of clusters, S-clusters (second-level clusters) for these cells. The size of an S-cluster must also be large enough to cover all cells spanned by an event, and it must interleave with the F-clusters so it can cover adjacent cells in different F-clusters. Each S-cluster also has a cluster-head, S-aggregator, for aggregating packets. Each S-aggregator creates a shortest path to the sink, and forms a second SPT in the network. We call it S-Tree. The illustration of an S-Tree is shown in Figure 2. For all sets of nearby cells that can be triggered by an event, either they will be in the same F-cluster, or they will be in the same S-cluster. This property is exploited by Dynamic Forwarding to avoid the long stretch problem discussed earlier.

After the S-Tree is constructed, the F-aggregators connect themselves to the S-aggregators of S-clusters which its F-cluster overlaps with, as shown in Figure 2.

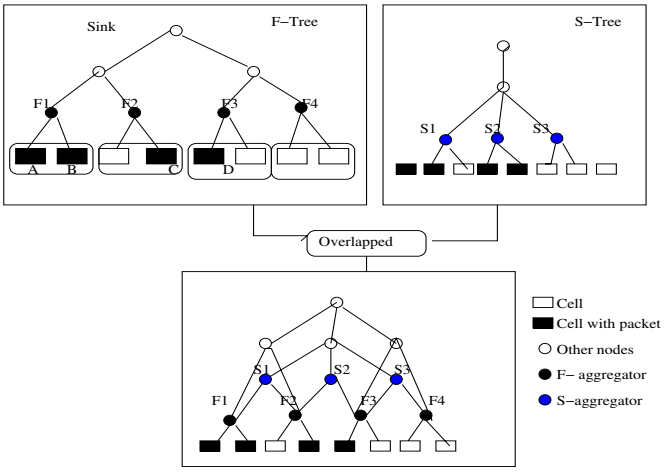

Fig. 2: The construction of F-Tree, S-Tree, and ToD.

Consider the example in Figure 2. If the event spans A and B, F1 knows that no other F-cluster will have packets for aggregation since the maximum number of cells an event can span is two; hence, it can forward the packets using the F-Tree. If the event spans two cells in two different F-clusters, for example, C and D, F4 will only receive packets from C, and F5 will only receive packets from D. F4 can know either the event happens only in C, or it spans D as well. Consequently, F4 can forward packets to S4, the S-aggregator of its overlapped S-clusters covering C, and so is F5. Therefore, these packets can be aggregated at S4.

Note that we do not specifically assign cells on the boundary of the network to any S-cluster. They do not need to be in any S-cluster if they are not adjacent to any other F-cluster, or they can be assigned to the same S-cluster as its adjacent cell.

B. Algorithm

The Dynamic Forwarding approach requires that each F-aggregator knows the location of S-aggregators of S-clusters that its F-cluster overlaps with. To simplify the cluster-head selection process and avoid the overhead of propagating the update information, we delegate the role of S-aggregators to F-aggregators. We choose an F-cluster, called Aggregating Cluster, for each S-cluster, and use the F-aggregator of the Aggregating Cluster as its S-aggregator. The Aggregating Cluster of an S-cluster is the F-cluster which is closest to the sink among all F-clusters that the S-cluster overlaps with, as shown in Figure 3, assuming that the sink is located on the bottom-left corner. When an F-aggregator forwards packets to an S-aggregator, it forwards them toward the aggregating cluster of that S-aggregator. When packets reach the aggregating cluster, nodes in that F-cluster know the location of their F-aggregator and can forward packets to it.

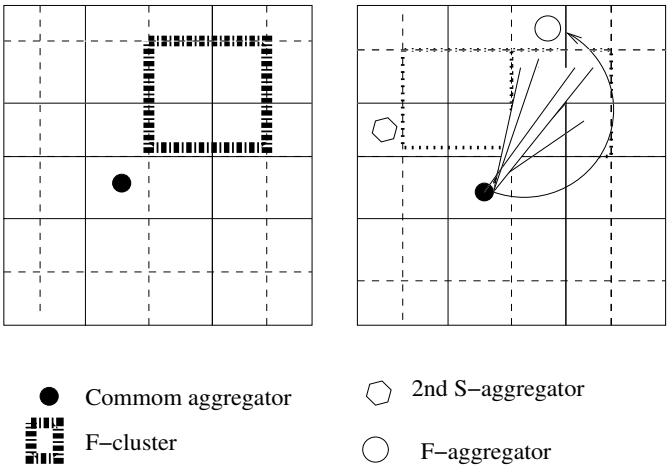

Fig. 3: The construction of F-Tree, S-Tree, and ToD.

TABLE I: AGGREGATING CLUSTER SELECTION

```
// Cells of sources can be represented by bit-mask
of four bits.
// Topleft cell is bit 1, topright cell is bit 2,
// bottomleft is bit 3 and bottomright cell is bit 4.
Aggregating_Cluster(Packet p)
cells = cells of sources in p
begin
      if (cells==topleft) then
    1st_fcluster=2nd_fcluster=left F-cluster
      else if (cells==topright) then
    1st_fcluster=2nd_fcluster=my F-cluster
      else if (cells==bottomleft) then
    1st_fcluster=2nd_fcluster=bottom left F-cluster
      else if (cells==bottomright) then
    1st_fcluster=2nd_fcluster=bottom F-cluster
      else if (cells==(topleft | topright)) then
    1st_fcluster=my F-cluster
    2nd_fcluster=left F-cluster
      else if (cells==(bottomleft | bottomright)) then
    1st_fcluster=bottom F-cluster
    2nd_fcluster=bottom_left F-cluster
      else if (cells==(topleft | bottomleft)) then
    1st_fcluster=left F-cluster
    2nd_fcluster=bottom_left F-cluster
      else if (cells==(topright | bottomright)) then
    1st_fcluster=my F-cluster
    2nd_fcluster=bottom F-cluster
      else
     dst=sink

end
```

Algorithm shows the pseudocode for selecting the first and second aggregating clusters based on where the packets come from. The pseudocode only shows the simplified procedure and does not show the aggregating cluster selection code when the F-cluster is located at the boundary of the network, where the left, bottom, or bottom-left F-cluster does not exist. If the next aggregating cluster is the left or bottom F-cluster that does not exist, the F-aggregator will choose itself as the next aggregator. If the next aggregating cluster is the bottom-left F-cluster that does not exist, the F-aggregator chooses the bottom or left F-cluster if either of them exists, or selects itself if none of them exists.

V. EVALUATION

In this section, we use experiments and simulations to evaluate the performance of our semistructured approach and compare it with other protocols.

A. Testbed Evaluation

We conduct experiments on Kansei sensor testbed to show the advantage and practicability of the ToD approach. The testbed consists of 105 Mica2-based motes and each mote is hooked onto a Stargate. The Stargate is a 32-bit hardware device from CrossBow running Linux. The 105 nodes form a 7 X 15 grid network with 3-*ft* spacing. The radio signal using default transmission power covers most nodes in the testbed.

We implement an Anycast MAC protocol on top of the Mica2 MAC layer. The Anycast MAC layer has its own backoff and retransmission mechanisms and we disable the ACK and backoff of the Mica2 MAC module. An event is emulated by broadcasting a message on the testbed to the Stargates, and the Stargates send the message to the Mica2 nodes through serial port. The message contains a unique ID distinguishing packets generated at different time.

When a node is triggered by an event, an event report is generated. If the node has to delay its transmission, it stores the packet in a report queue. Both the application layer and Anycast MAC layer can access the queue; therefore, they can check if the node has packets for aggregation, or aggregate the received packets to packets in the queue. We evaluate the following protocols1 on the testbed:
$DynamicForwardingoverToD(ToD)$: The semistructured approach we proposed in this paper. DAA is used to aggregate packets in each F-cluster, and aggregated packets are forwarded to the sink on ToD.
DAA : The structureless approach proposed in [7].
SPT : Nodes send packets to the sink through the SPT immediately after sensing an event. Aggregation is opportunistic and happens only if two packets are at the same node at the same time.

B. Simulation Results

Figure 4, shows the normalized number of transmissions for different event sizes. We fixed the location of the event and vary its diameter from 12 to 36 *ft* where nodes within two grid-hops to six grid-hops of the event will be triggered, respectively, and send packets to the sink located at one corner of the network. We use 6 seconds as maximum delay for all protocols except SPT. For event size less than 12 ft, there are too little nodes been triggered (less than five), and all triggered nodes are within transmission range. Data aggregation is not so interesting in such scenario; therefore, we do not evaluate it. All protocols have better performance when the size of the event increases because packets have more chances of being aggregated. ToD performs best

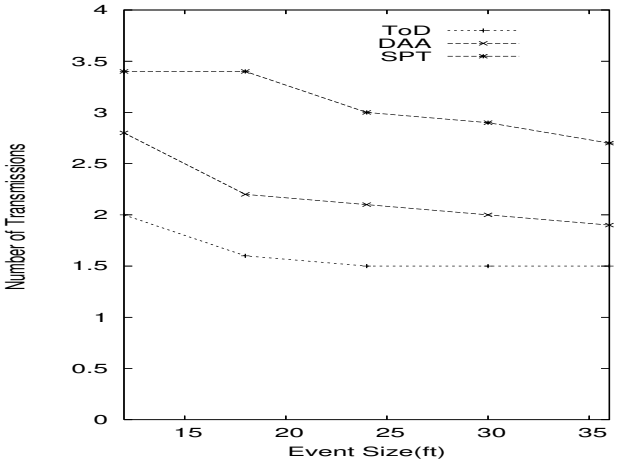

Fig. 4: The normalized number of transmissions for different event sizes.

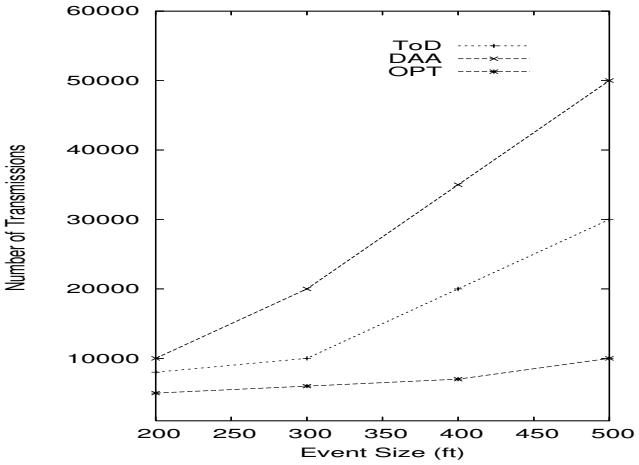

Fig. 5: Number of transmissions.

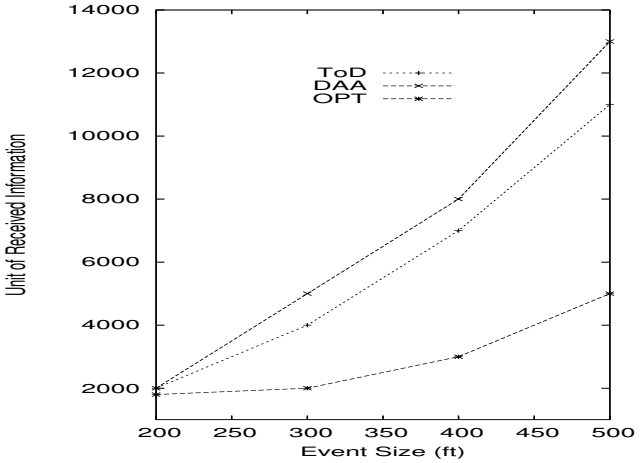

Fig. 6: Unit of received information.

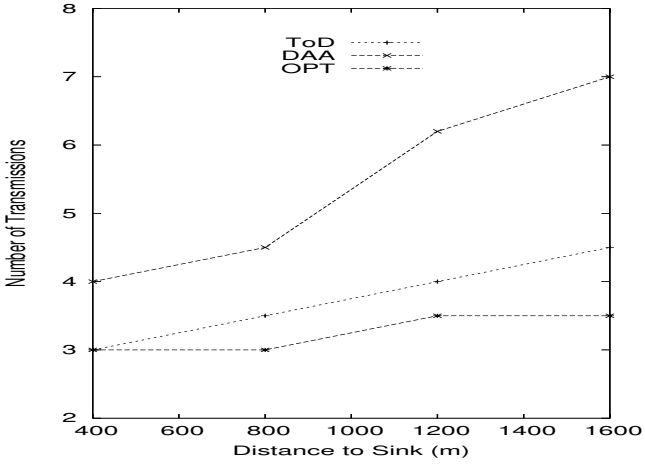

Fig. 7: Number of Transmission.

among all protocols in all scenarios. This shows that DAA can efficiently achieve early aggregation and the Dynamic Forwarding over ToD can effectively reduce the cost of directly forwarding unaggregated packets to the sink in DAA.

Figures 5, show the total number of transmissions and total units of useful information received by the sink. DAA and ToD have higher number of received packets than OPT due to the ability of structureless aggregation to aggregate packets early and scatter them away from each other to reduce contention. ToD performs better than DAA in terms of the normalized number of transmissions because of its ability to aggregate packets at nodes closer to the source, and thus, it reduces the cost of forwarding packets from sources to the sink. It has slightly lower number of units of received information than DAA. From the simulation logs, we found that most dropped packets in ToD are packets forwarded from sources to their F-aggregators. We believe that the convergecast causes higher contention, thus leading to higher dropping rate.

Figure 6, shows the results of scalability simulations. The performance of ToD and OPT remains steady. This shows that ToD is quite scalable as its performance does not degrade as the size of the network increases.

Figure 7, shows the performance of both DAA and SPT degrades as the size of the network increases. The normalized number of transmissions for DAA and SPT doubled when the event moves from the closest to the sink to the farthest.

Figure 8, shows the number of packets received at the sink per event. If all packets can be aggregated near the event and forwarded to the sink, the sink will receive only one packet. Conversely, more packets received at the sink shows that fewer aggregations happened in the network. The cost of forwarding more packets to the sink increases rapidly as the size of the network increases. We can see that, in both DAA and SPT, the sink receives many packets. Although the number of packets received at the sink remains quite steady, the total number of transmissions increases linearly as the distance from the sources to the sink increases.

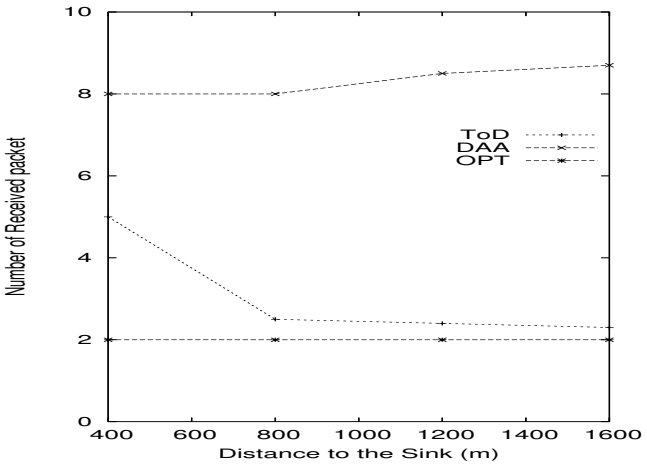

Fig. 8: Number of received packets.

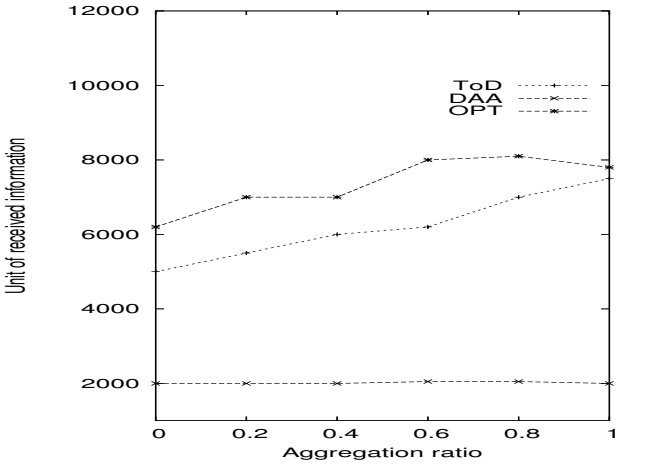

Fig. 9: Unit of received information.

As shown in Figure 9, ToD improves the normalized number of transmissions of DAA, but the improvement decreases as the aggregation ratio decreases. This is because when the aggregation ratio decreases, packet size increases after aggregation. Packets cannot be aggregated anymore when they reach maximum payload even if they meet. Both ToD and DAA perform better than OPT when the aggregation ratio is not 1 because the packet dropping rate in OPT is very high. OPT only receives less than 2,000 units of information, compared to more than 5,000 in ToD and DAA. We believe the high dropping rate is because of the convergecast traffic in OPT. When aggregation ratio decreases, more packets with larger size is forwarded to the root of the aggregation tree, which results in high contention and leads to high dropping rate.

VI. CONCLUSIONS

A semistructured approach are proposed that locally uses a structureless technique followed by Dynamic Forwarding on an implicitly constructed packet forwarding structure, ToD, to support network scalability. ToD avoids the long stretch problem in fixed structured approaches and eliminates the overhead of constructing and maintaining dynamic structures. We evaluate its performance using simulations on 2,000 node networks. Based on our studies, we find that ToD is highly scalable and it performs close to the optimal structured approach. Therefore, it is very suitable for conserving energy and extending the lifetime of large-scale sensor networks.

REFERENCES

[1] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for selforganization of a wireless sensor network," *IEEE Communication*, vol. 7, pp. 16–27, oct. 2007.
[2] M. Ding, X. Cheng, and G. Xue, "Aggregation Tree Construction in Sensor Networks," in *Proc. 58th IEEE Vehicular Technology Conf. (VTC 03)*, vol. 4, pp. 2168–2172, Oct. 2003.
[3] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Aggregation Tree Construction in Sensor Networks," in *Data Gathering Algorithms in Sensor Networks Using Energy Metrics*, vol. 13, pp. 924–935, Sept. 2002.
[4] R. Cristescu and M. Vetterli, "Power Efficient Gathering of Correlated Data: Optimization, NP-Completeness and Heuristics," in *Summaries of MobiHoc 2003 Posters*, vol. 7, pp. 31–32, July 2003.
[5] A. Goel and D. Estrin, "Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk," in *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2003.
[6] N. Alon, R.M. Karp, D. Peleg, and D. West, "A Graph Theoretic Game and Its Application to the K-Server Problem," in *SIAM J. Computing*, vol. 24, 1995.
[7] K.W. Fan, S. Liu, and P. Sinha, "On the Potential of Structure-Free Data Aggregation in Sensor Networks," in *Proc. IEEE INFOCOM 06*, Apr 2006.
[8] Krishanamachari, Estrin D., Wicker S, "The Impact of Data Aggregation in Wireless Sensor Networks," in *IEEE Confernece on Distributed Comput- ing systems Workshop*, 2002.
[9] Weisong Shi, Sivakumar Sellamuthu, Kewei Sha and Loren Schwiebert, "Query Agent: A General Query Processing Tool for Sensor Networks," in *IEEE Conference on Parallel Computing Workshop*, September 2004.
[10] S. Chatterjea, P. Havinga, "A Dyanamic Data Aggregation Scheme for Wireless Sensor Networks," *ProRISC, Veldhoven, The Netherlands*, 26-27 November 2003.

Prakash G L is an Assistant Professor with the Department of Computer Science and Engineering of Alpha College of Engineering, Bangalore, India. He received his B.E and M.E degrees in Computer Science and Engineering from Bangalore University, Bangalore. He is presently pursuing his Ph.D programme in the area of Wireless Sensor Networks in Bangalore University.

Thejaswini M is a Faculty with the Department of Computer Science and Engineering of Sri Jagadguru Mallikarjuna Murugharajendra Institute of Technology, Chitradurga, India. She received her B.E in Information Science and M.E degrees in Computer Science and Engineering from Visveswaraiah Technological University and Bangalore University respectively. Her research interests include Wireless Sensor Networks.

Manjula S H is an Assistant Professor with the Department of Computer Science and Engineering of University Visvesvaraya College of Engineering, Bangalore, India. She received her B.E and M.E degrees in Computer Science and Engineering from Bangalore University, Bangalore. She is presently pursuing her Ph.D programme in the area of Wireless Sensor Networks.

K R Venugopal is currently the Principal and Dean, Faculty of Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Comput- er Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D. in Economics from Bangalore University and Ph.D. in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored 27 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ etc. He has been serving as the Professor and Chairman, Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. During his three decades of service at UVCE he has over 200 research papers to his credit. His research interests include computer networks, parallel and distributed systems, digital signal processing and data mining.

L M Patnaik is a Vice Chancellor, Defence Institute of Advanced Technology(Deemed University), Pune, India. During the past 35 years of his service at the Indian Institute of Science, Bangalore. He has over 500 research publications in refereed International Journals and Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to high performance computing and soft computing. His areas of research interest have been parallel and distributed computing, mobile computing, CAD for VLSI circuits, soft computing, and computational neuroscience.