Scalable Data Aggregation for Dynamic Events in Sensor Networks *

Kai-Wei Fan
Dept. of Computer Science and Engineering
The Ohio State University
fank@cse.ohio-state.edu

Sha Liu
Dept. of Computer Science and Engineering
The Ohio State University
liusha@cse.ohio-state.edu

Prasun Sinha
Dept. of Computer Science and Engineering
The Ohio State University
prasun@cse.ohio-state.edu

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 which is used by nodes to wait for packets from their children before forwarding the packet to the sink. Although structure-less approaches can address these issues, the performance does not scale well with the network size. We propose a semi-structured approach that uses a structure-less technique locally followed by *Dynamic Forwarding* on an implicitly constructed packet forwarding structure to support network scalability. The structure, *ToD*, is composed of multiple shortest path trees. 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 one of these trees in ToD, thus resulting in early aggregation of packets. Based on simulations on a 2000 nodes network and real experiments on a 105 nodes Mica2-based network, we conclude that efficient aggregation in large scale networks can be achieved by our semi-structured approach.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Performance

Keywords

Data Aggregation, Anycasting, Structure-free, ToD

*This work was partially supported by NSF's CAREER program (Grant number CNS 0546630) and NSF's RI program (Grant number CNS 0403342)

SenSys'06, November 1–3, 2006, Boulder, Colorado, USA.

1 Introduction

Data aggregation is an effective technique for conserving communication energy in sensor networks. In sensor networks, the communication cost is often several orders of magnitude larger than the computation cost. Due to inherent redundancy in raw data collected from sensors, in-network data aggregation can often reduce the communication cost by eliminating redundancy and forwarding only the extracted information from the raw data. As reducing consumption of communication energy extends the network lifetime, it is critical for sensor networks to support in-network data aggregation.

Various data aggregation approaches have been proposed for data gathering applications and event-based applications. These approaches make use of cluster based structures [1, 2] or tree based structures [3–8]. In data gathering applications, such as environment and habitat monitoring [9–12], nodes periodically report the sensed data to the sink. As the traffic pattern is unchanging, these structure-based approaches incur low maintenance overhead and are therefore suitable for such applications. However, in event-based applications, such as intrusion detection [13, 14] and biological hazard detection [15], the source nodes are not known in advance. Therefore the approaches that use fixed structures can not efficiently aggregate data, while the approaches that change the structure dynamically incur high maintenance overhead [4, 5]. *The goal of this paper is to design a scalable and efficient data aggregation protocol that incurs low maintenance overhead and is suited for event-based applications.*

Constructing an optimal structure for data aggregation for various aggregation functions has been proven to be an NP-hard problem [16, 17]. Although heuristics can be used to construct structures for data aggregation, another problem associated with the convergecast traffic pattern, where nodes transmit their packets to the cluster-head or parent in cluster or tree structures, results in low performance of structure based data aggregation protocols. In [18] the simulation results show that the packet dropping rate in Shortest Path Tree (SPT) is higher because of heavy contention caused by the convergecast traffic. This results in more packet drops and increased delays. As a result, enforcing a fixed order of packet transmissions becomes difficult, which impacts the performance of data aggregation in structured approaches. Typically, packets have to be transmitted in a fixed order

from leaves to the root in a tree-like structure to achieve maximum aggregation. Dropped packets not only make the optimal structure sub-optimal, but also waste energy on transmitting packets that are unable to reach the sink.

In [19] it shows that the performance gain by using heuristics to create the Steiner Minimum Tree (SMT) for aggregation is not significant compared with using only the Shortest Path Tree (SPT), not to mention that the overhead of constructing such a structure may negate the benefit resulting from data aggregation. However, their conclusions were based on the assumption of randomly located data sources, which is different from the scenarios in event-based sensor networks where a set of close-by nodes is expected to sense an event.

Realizing the shortcomings of structured approaches, [20] proposes an anycast based structure-less approach at the MAC layer to aggregate packet. It involves mechanisms to increase the chance of packets meeting at the same node (Spatial Aggregation) at the same time (Temporal Aggregation). As the approach does not guarantee aggregation of all packets from a single event, the cost of forwarding unaggregated packets increases with the scale of the network and the distance of the event from the sink.

To benefit from the strengths of the structured and the structure-less approaches, we propose a semi-structured approach in this paper. The main challenge in designing such a protocol is to determine the packet forwarding strategy in absence of a pre-constructed global structure to achieve early aggregation. Our approach uses a structure-less technique locally followed by *Dynamic Forwarding* on an implicitly constructed packet forwarding structure to support network scalability. The structure, *ToD* (Tree on Directed acyclic graph), is composed of multiple shortest path trees. After performing local aggregation, nodes dynamically decide the forwarding tree based on the location of the source nodes. 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. This paper makes the following contributions:

- We propose an efficient and scalable data aggregation mechanism that can achieve early aggregation without incurring any overhead of constructing a structure.
- We implement the *ToD* approach on TinyOS and compare its performance against other approaches on a 105 nodes 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 2000 nodes.

The organization of the rest of the paper is as follows. Section 2 presents background and related work. Section 3 presents the structure-less approach. Section 4 analyzes the performance of ToD in the worst case. The performance evaluation of the protocols using simulations and experiments is presented in Section 5. Finally Section 6 concludes the paper.

2 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 [21–24], some focus on how to construct and maintain a structure to facilitate data aggregation [1–8,17,25–30], and some focus on how to efficiently compress and aggregate data by taking the correlation of data into consideration [17, 31–34]. As our work focuses on how to facilitate data aggregation without incurring the overhead of constructing a structure, we briefly describe the structure-based as well as structure-less approaches in current research.

In [1,2], the authors propose the LEACH protocol to cluster sensor nodes and let the cluster-heads aggregate data. The cluster-heads then communicate directly with the base station. PEGASIS [26] extends LEACH by organizing all nodes in a chain and letting nodes be the head in turn. [26, 27] extend PEGASIS by allowing simultaneous transmission that balances the energy and delay cost for data gathering. Both LEACH and PEGASIS assume that any node in the network can reach the base-station directly in one-hop, which limits the size of the network for them to be applicable.

GIT [3] uses a different approach as compared to LEACH. GIT is built on top of a routing protocol, Directed Diffusion [21,22], which is one of the earliest proposed attribute-based routing protocols. In Directed Diffusion, data can be aggregated opportunistically when they meet at any intermediate node. Based on Directed Diffusion, the Greedy Incremental Tree (GIT) establishes an energy-efficient tree by attaching all sources greedily onto an established energy-efficient path and pruning less energy efficient paths. However due to the overhead of pruning branches, GIT might lead to high cost in moving event scenarios.

In [4, 5], the authors propose DCTC, Dynamic Convoy Tree-Based Collaboration, to reduce the overhead of tree migration in mobile event scenarios. DCTC assumes that the distance to the event is known to each sensor and uses the node near the center of the event as the root to construct and maintain the aggregation tree dynamically. However it involves heavy message exchanges which might offset the benefit of aggregation in large-scale networks. From the simulation results in DCTC [5], the energy consumption of tree expansion, pruning and reconfiguration is about 33% of the data collection.

In [8], the authors propose an aggregation tree construction algorithm to simultaneously approximate the optimum trees for all non-decreasing and concave aggregation functions. The algorithm uses a simple min-cost perfect matching to construct the tree. [7] also uses similar min-cost matching process to construct an aggregation tree that takes the data fusion cost into consideration. Other works, such as SMT (Steiner Minimum Tree) and MST (Multiple Shared Tree) for multicast algorithms which can be used in data aggregation [17, 19, 30], 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.

Moreover, fixed tree structure might have long stretch between adjacent nodes. 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 packets from adjacent nodes have to be forwarded many hops away before they can be aggregated. This problem has been studied as MSST (Minimum Stretch Spanning Tree) [35] and MAST (Minimum Average Stretch Spanning Tree) [36]. They are also NP-hard problems, and it has been shown that for any graph, the lower bound of the average stretch is $O(log(n))$ [36], and it can be as high as $O(n)$ for the worst case [37]. Even for a grid network, it has been shown that the lower bound for the worst case is $O(\sqrt{n})$ [36]. [38] proposes a polynomial time algorithm to construct a group-independent spanning tree that can achieve $O(log(n))$ stretch. However the delay in [38] is high in large networks if only nodes near the sink are triggered.

[20] is the first proposed structure-less data aggregation protocol that can achieve high aggregation without incurring the overhead of structure approaches. [20] 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 from a single event. 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 the control overhead.

In order to get benefit from structure-less approaches even in large networks, scalability has to be considered in the design of the aggregation protocol. In this paper, we propose a scalable structure-less protocol, ToD, that can achieve efficient aggregation even in large networks. ToD uses a semi-structure approach that does not have the long stretch problem in fixed structure nor incur structure maintenance overhead of dynamic structure, and further improves the performance of the structure-less approach.

3 Scalable Data Aggregation

As described before, 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 very large sensor networks.

Our protocol is based on the *Data Aware Anycast (DAA)* and *Randomized Waiting (RW)* approaches[1] proposed in [20]. There are two phases in our protocol: *DAA* and *Dynamic Forwarding*. In the first phase, packets are forwarded and aggregated to a selected node, termed aggregator, using DAA. In DAA [20], packets were destined to the sink, whereas in our approach they are destined to an aggregator. In the second phase, the leftover un-aggregated or partially aggregated packets are forwarded on a structure, termed *Tree on DAG (ToD)*, for further aggregation. First we briefly describe the DAA protocol proposed in [20].

3.1 Data Aware Anycast [20]

Data Aware Anycast is a structure-less protocol that aggregates packets by improving the *Spatial* and *Temporal* convergence. Spatial convergence and temporal convergence during transmission are two necessary conditions for aggregation. Packets have to be transmitted to the same node at the same time to be aggregated. Structured approaches achieve these two conditions by letting nodes transmit packets to their parents in the aggregation tree and parents wait for packets from all their children before transmitting the aggregated packets. Without explicit message exchanges in structure-less aggregation, nodes do not know where they should send packets to and how long they should wait for aggregation. Therefore improving spatial or temporal convergence is critical for improving the chance of aggregation.

Spatial Convergence is achieved by using anycast to forward packets to nodes that can achieve aggregation. Anycast is a routing scheme whereby packets are forwarded to the best one, or any one, of a group of target destinations based on some routing metrics. By exploiting the nature of wireless radio transmission in sensor networks where all nodes within the transmission range can receive the packet, nodes are able to tell if they can aggregate the transmitting packet, and the anycast mechanism allows the sender to forward packets to any one of them. Transmitting packets to nodes that can achieve aggregation reduces the number of remaining packets in the network, thereby reducing the total number of transmissions.

Temporal Convergence is used to further improve the aggregation. *Randomized Waiting* is a simple technique for achieving temporal convergence, in which nodes wait for a random delay before transmitting. In mobile event triggered networks, nodes are unable to know which nodes are triggered and have packets to transmit in advance. Therefore nodes can not know if they should wait for their upstream nodes and how long they should wait for aggregation. A naive approach of using a fixed delay depending on the distance to the sink may make the detection delay very high. For example, as shown in Fig. 1, nodes closer to the sink must wait longer for packets from possible upstream nodes if fixed waiting time is employed. When events are closer to the sink, the longer delay chosen by nodes closer to the sink is unnecessary. Random delay is used to avoid long delay in large networks while increasing the chance of aggregation.

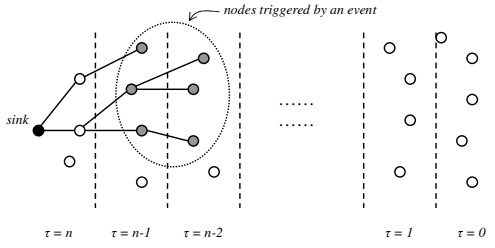


Figure 1. Longer delay is unnecessary but is inevitable using fixed delay when the event is closer to the sink. Nodes closer to the sink have longer delay (τ) because they have to wait for packets from possible upstream nodes.

When a node detects an event and generates a packet for reporting, it picks a random delay between 0 and τ before transmitting, where τ is a network parameter that specifies the maximum delay. After delaying the packet, the node

[1] In rest of this paper, we use DAA or Data Aware Anycast to refer to the combination of the two approaches.

broadcasts an RTS packet containing an *Aggregation ID*. In [20], the timestamp is used as the Aggregation ID, which means that packets generated at the same time can be aggregated. When a node receives an RTS packet, it checks if it has packets with the same Aggregation ID. If it does, it has higher priority for replying with a CTS than nodes that do not have packets for aggregation. The priority is decided by the delay of replying a CTS packet. Nodes with higher priority reply a CTS with shorter delay. If a node overhears any traffic before transmitting its CTS packet, it cancels the CTS transmission in order to avoid collision of multiple CTS responses at the sender. Therefore, nodes can send their packets for aggregation as long as at least one of its neighbors has a packet with the same Aggregation ID. More details and extensions of the DAA approach can be found in [20].

However, DAA can not guarantee that all packets will be aggregated into one packet. When more packets are transmitted from sources to the sink without aggregation, more energy is wasted. This effect becomes more severe when the network is very large and the sources are very far away from the sink. Therefore, instead of forwarding packets directly to the sink when DAA can not aggregate packets any more, we propose the use of Dynamic Forwarding for further packet aggregation. We now describe the Dynamic Forwarding and the construction of ToD.

3.2 Dynamic Forwarding over ToD

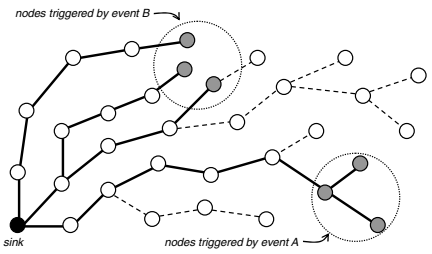


Figure 2. Fixed tree structure for aggregation can have long distance (link-stretch) between adjacent nodes, as in the case of nodes triggered by event *B*. In this example we assume that nodes in the range of event *B* are within transmission range of each other.

We adopt the pre-constructed structure approach in the second phase to achieve further aggregation. Having a structure to direct all packets to a single node is inevitable if we want to aggregate all packets into one. Constructing a structure dynamically with explicit message exchanges incurs high overhead. Therefore we use an implicitly computed pre-constructed structure that remains unchanged for relatively long time periods (several hours or days). However, using a fixed structure has the long stretch problem as described in Section 2. Take Fig. 2 as an example of precomputed tree structure where gray nodes are the sources. The fixed tree structure works well if the nodes that generate packets are triggered by event *A* because their packets can be aggregated immediately on the tree. However, if the nodes that generate packets are triggered by event *B*, their packets can not be aggregated even if they are adjacent to each other. Therefore we design a dynamic forwarding mechanism over *ToD*, to avoid the problem of long stretch.

3.2.1 ToD in One Dimensional Networks

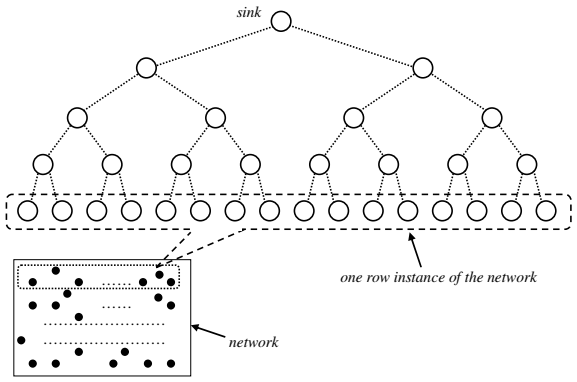


Figure 3. We illustrate the ToD construction from one row's point of view to simplify the discussion.

For illustrating the concept of ToD, we first describe the construction of ToD for a 1-D (a single row of nodes) network, as shown in Fig. 3. We assume that the nodes can communicate with their adjacent nodes in the same row through one hop.

We define a cell as a square with side length Δ where Δ is greater than the maximum diameter of the area that an event can span. 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 1-D cells in the network. All nodes in F-clusters send their packets to their cluster-heads, called F-aggregators (First-level aggregators). Note that nodes in the F-cluster can be multiple hops away from the F-aggregator. The formation of the clusters and the election of the aggregators are discussed later in Section 3.2.3. Each F-aggregator then creates a shortest path to the sink. Therefore the structure is a shortest path tree where the root is the sink and the leaves are F-aggregators. We call this tree an *F-Tree*. Fig. 4(a) shows the construction of the F-Tree.

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 shortest path tree in the network. We call it *S-Tree*. The illustration of an S-Tree is shown in Fig. 4(b). 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 Fig. 4(c). For example, in Fig. 4(c), the F-aggregator $F4$ connects to S-aggregators $S3$ and $S4$ because its F-cluster overlaps with S-cluster 3 and 4. Thus, the combination of F-Tree and S-Tree creates a Di-

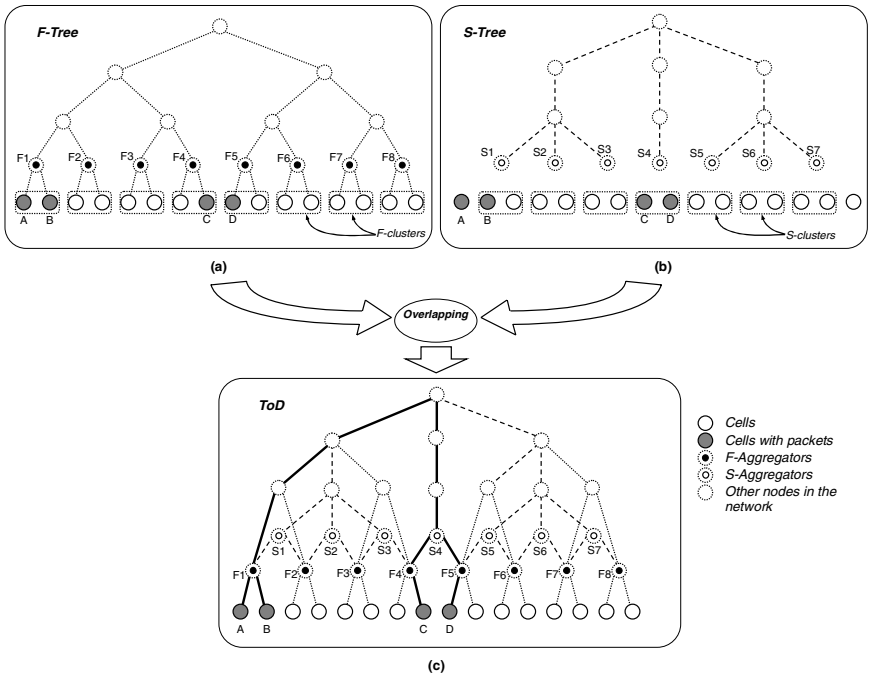


Figure 4. **The construction of F-Tree, S-Tree, and ToD. (a) Leaf nodes are cells. Pairs of neighbor cells define F-clusters. Each F-cluster has an F-aggregator, and F-aggregators form the F-Tree. (b) Each pair of adjacent cells not in the same F-cluster form an S-cluster. Each S-cluster has an S-aggregator, and S-aggregators form the S-Tree. Nodes on the network boundary do not need to be in any S-cluster. (c) Each F-aggregator connects to two S-aggregators of S-clusters which its F-cluster overlaps with. This structure called the Tree on DAG or ToD. 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.**

rected Acyclic Graph, which we refer to as the ToD (Tree on DAG).

Nodes first use the *Data Aware Anycast* (DAA) approach to aggregate as many packets as possible. When no further aggregation can be achieved by DAA, nodes forward their packets to the F-aggregator in its F-cluster. If an event only triggers nodes within a single F-cluster, its packets can be aggregated at the F-aggregator, and be forwarded to the sink using the F-Tree. However, in case the event spans multiple F-clusters, the corresponding packets will be forwarded to different F-aggregators. As we assumed that the event size is not larger than the size of a cell, an event on the boundary of F-clusters will only trigger nodes in cells on the boundary of the F-clusters. By the construction of S-clusters, adjacent cells on the boundary of F-clusters belong to the same S-cluster. Thus, F-aggregators can exploit the information collected from received packets to select the S-aggregator that is best suited for further aggregation. This information is obtained from the source of traffic that can be encoded in the packets. Often such information is readily available in the packet. Otherwise, 4 extra bits can be used to indicate which cell the packet comes from.

Consider the example in Fig. 4(c). Since the maximum number of cells an event can span is two, either these two cells are in the same F-cluster, or they are in the same S-cluster. If they are in the same F-cluster, their packets can be aggregated at the F-aggregator. For example, if the event spans A and B, $F1$ knows that no other F-cluster has packets for aggregation, and it can forward the packets using the F-Tree. If the event spans two cells that are in different F-clusters, the two F-aggregators in the two F-clusters will receive packets only from one of their cells. The F-aggregators then conjecture which F-cluster might also have packets based on which cells the packets come from. For example, if the event spans C and D, $F4$ will only receive packets from C. Therefore $F4$ can know either the event happens only in C, or the event spans C and D. Consequently, $F4$ can forward packets to $S4$, the S-aggregator of its overlapped S-clusters covering C. Also $F5$ will forward its packets to $S4$ if packets only come from D. 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.

The ToD for the one dimensional network has the following property.

Property 1. *For any two adjacent nodes in ToD in one dimensional network, their packets will be aggregated either at a first level aggregator, or will be aggregated at a second level aggregator.*

Proof. There are only three possibilities when an event triggers nodes to generate packets. If only nodes in one cell are triggered and generate the packets, their packets can be aggregated at one F-aggregator since all nodes in a cell reside

in the same F-cluster, and all packets in an F-cluster will be aggregated at the F-aggregator.

If an event triggers nodes in two cells, and these two cells are in the same F-cluster, the packets can be aggregated at the F-aggregator as well.

If an event triggers nodes in two cells, but these two cells are in different F-clusters, they must reside in the same S-cluster because S-clusters and F-clusters are interleaved. Moreover, packets in one F-cluster will only originate from the cell that is closer to the other F-cluster that also has packets. Therefore the F-aggregator can forward packets to the S-aggregator for aggregation accordingly, and packets will be aggregated at the S-aggregator.

Since the cell is not smaller than the maximum size of an event, it is impossible for an event to trigger more than two cells, and this completes the proof.

□

3.2.2 ToD in Two Dimensional Networks

Section 3.2.1 only demonstrates the construction for one row of nodes to illustrate the basic idea of dynamic forwarding, and it works because each cell is only adjacent to one (or none, if the cell is on the boundary of the network) of the F-clusters. Therefore if an event spans two cells, the two cells are either in the same F-cluster or in the same S-cluster, and the F-aggregator can conjecture whether to forward the packets to the S-aggregator, or to the sink directly. When we consider other cells and F-clusters in the adjacent row, a cell on the boundary of an F-cluster might be adjacent to multiple F-clusters. If an event spans multiple cells, each F-aggregator may have multiple choices of S-aggregators if the cells in their F-cluster are adjacent to multiple F-clusters. If these F-aggregators select different S-aggregators, their packets will not be aggregated. However, the ideas presented in 1D networks can be extended for the 2D networks. But instead of guaranteeing that packets will be aggregated within two steps as in the 1D case (aggregating either at an F-aggregator or an S-aggregator), the ToD in 2D guarantees that the packets can be aggregated within three steps.

We first define the cells and clusters in two dimensions. For the ease of understanding, we use grid clustering to illustrate the construction. As defined before, the size of a cell is not less than the maximum size of an event, and an F-cluster must cover all the cells that an event might span, which is four cells in 2D grid-clustering. Therefore the entire network is divided into F-clusters, and each F-cluster contains four cells. The S-clusters have to cover all adjacent cells in different F-clusters. Each F-cluster and S-cluster also has a cluster-head acting as the aggregator to aggregate packets. Fig. 5 shows a 5×5 network with its F-clusters and S-clusters.

Since the size of a cell (one side of the square cell) must be greater or equal to the maximum size of an event (diameter of the event), an event can span only one, two, three, or four cells as illustrated in Fig. 6. If the event only spans cells in the same F-cluster, the packets can be aggregated at the F-aggregator. Therefore we only consider scenarios where an event spans cells in multiple F-clusters.

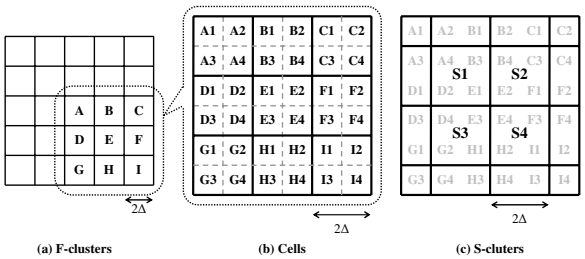


Figure 5. Grid-clustering for a two-dimension network. (a) The network is divided into 5×5 F-clusters. (b) Each F-cluster contains four cells. For example the F-cluster A in (a) contains cell $A1$, $A2$, $A3$, and $A4$. (c) The S-clusters have to cover all adjacent cells in different F-clusters. Each S-cluster contains four cells from four different F-clusters.

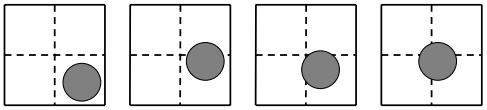


Figure 6. The possible numbers of cells an event may span in 2×2 cells, which are one, two, three, and four from left to right. The four cells in each case are any instance of four cells in the network. They may be in the same F-cluster or different F-clusters.

Fig. 7 shows four basic scenarios that an F-aggregator may encounter when collecting all packets generated in its F-cluster. All other scenarios are only different combinations of these four scenarios. If packets originate from three or four cells in the same F-cluster, the F-aggregator knows that no other nodes in other F-clusters have packets, and it can forward the packets directly to the sink. If only one or two cells generate packets, it is possible that other F-clusters also have packets. We assume that the region spanned by an event is contiguous. So simultaneous occurrence of scenarios of (a) and (c), or (b) and (d), is impossible in the F-cluster. However, such scenarios are possible in presence of losses in a real environment where packets from third or fourth cluster are lost. In such cases the F-aggregator can just forward the packets directly to the sink because no other F-cluster will have packets from the same event.

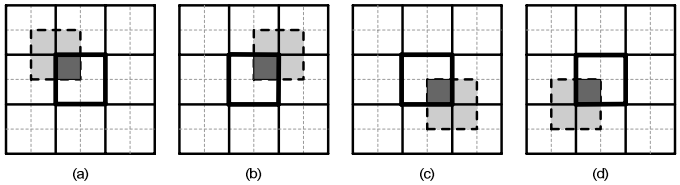


Figure 7. All possible scenarios in an F-aggregator's point of view. Each case shows 3×3 F-clusters, and the aggregator of the center F-cluster is making the decision. The dark grayed squares are cells that generate packets, and the light grayed squares represent the corresponding S-cluster of the dark grayed cells.

When the F-aggregator collects all packets within its cluster, it knows which cells the packets come from and forwards the packets to best suited S-aggregator for further aggregation. For example, if the packets only come from one cell as in case (a) in Fig. 7, the F-aggregator can forward the packet to the S-aggregator of the S-cluster that covers that

cell. However, if packets come from two cells in an F-cluster, the two cells must be in different S-clusters. For example, as in Fig. 8 where the F-aggregator of F-cluster X receives packets from two cells, is the combination of case (a) and (b) in Fig. 7. It is possible that the F-aggregator of F-cluster Y may receive packets from cells as in Fig. 7 (c), (d), or both. Since the F-aggregator of F-cluster X does not know which case the F-aggregator of F-cluster Y encounters, it does not know which S-aggregator to forward packets to. To guarantee the aggregation, the F-aggregator of F-cluster X forwards the packet through two S-aggregators that covers cell $C1$ and $C2$, therefore packets can meet at least at one S-aggregator. If both F-aggregators receive packets from two cells in its cluster, to guarantee that the packets can meet at least at one S-aggregator, these two F-aggregators must select the S-aggregator deterministically. The strategy is to select the S-aggregator that is closer to the sink. If the packets meet at the first S-aggregator, it does not need to forward packets to the second S-aggregator. The S-aggregator only forwards packets to the second S-aggregator if the packets it received only come from two cells in one F-cluster. We will present a simplified construction later (in Section 3.2.3) for the selection of S-aggregators.

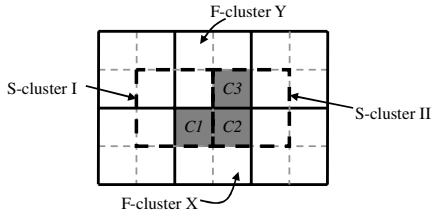


Figure 8. **The F-aggregators have two choices for S-aggregators if they receive packets from two cells.**

To guarantee that the packets can meet at least at one S-aggregator, the second S-aggregator must wait longer than the first S-aggregator. Therefore, if the S-aggregator receives packets from only one cell, it waits longer to wait for possible packets forwarded by the other S-aggregator because it could be the second S-aggregator of the other F-aggregator. Fig. 9 shows an example of one F-aggregator sending packets to the first S-aggregator and then the second S-aggregator, while the other F-aggregator sends packets directly to the second S-aggregator. As long as the second S-aggregator waits sufficiently longer than the first S-aggregator the packets can be aggregated at the second S-aggregator.

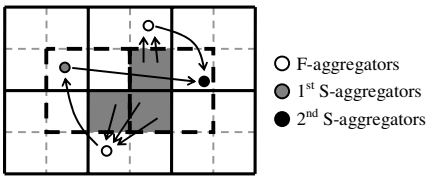


Figure 9. **Depending on how many cells generate packets in its F-cluster, one F-aggregator sends packets to two S-aggregators while the other F-aggregator sends packets to only one S-aggregator. We assume that the sink is located at bottom-left of the network.**

The ToD for the two dimension networks has the following property.

Property 2. *For any two adjacent nodes in ToD, their packets will be aggregated at the F-aggregator, at the* 1^{st} *S-aggregator, or at the* 2^{nd} *S-aggregator.*

Proof. First we define the F-aggregator X as the aggregator of F-cluster X and S-aggregator I as the aggregator of S-cluster I, and so forth.

For packets generated only in one F-cluster, their packets can be aggregated at the F-aggregator since all packets in the F-cluster will be sent to the F-aggregator.

If an event triggers nodes in different F-clusters, there are only three cases. First, only one cell in each F-cluster generates packets. In this case, all cells having packets will be in the same S-cluster since the adjacent cells in different F-clusters are all in the same S-cluster. Therefore their packets can be aggregated at the S-aggregator.

Second, the event spans three cells, $C1$, $C2$, and $C3$, and two of them are in one F-cluster and one of them is in the other F-cluster. Without loss of generality, we assume that $C1$ and $C2$ are in the same F-cluster, F-cluster X, and $C3$ is in the other F-cluster, F-cluster Y. Moreover $C3$ must be adjacent to either $C1$ or $C2$, and let us assume that it is $C2$. From the ToD construction we know that $C2$ and $C3$ will be in the same S-cluster, S-cluster II, and $C1$ will be in another S-cluster, S-cluster I. Fig. 8 illustrates one instance of this case. First the F-aggregator X will aggregate packets from $C1$ and $C2$ because they are in the same F-cluster, and forward the aggregated packets through S-aggregator I to S-aggregator II, or the other way around, because $C1$ is in S-cluster I and $C2$ is in S-cluster II. F-aggregator Y will aggregate packets from $C3$ and forward packets to S-aggregator II because $C3$ is in S-cluster II. Because packets of F-aggregator Y only come from $C3$, they will have longer delay in S-aggregator II in order to wait for packets being forwarded through the other S-aggregator. In the mean time, if F-aggregator X forwards packets to S-aggregator II first, the packets can be aggregated at S-aggregator II. If F-aggregator X forwards packets to S-aggregator I first, S-aggregator I will forward packets to S-aggregator II with shorter delay because the packets come from two cells in one F-cluster, therefore their packets can also be aggregated at S-aggregator II.

In the third case, the event spans four cells. Two of them will be in one F-cluster and the other two will be in the other F-cluster. Without loss of generality, we can assume that cells $C1$ and $C2$ are in F-cluster X and cells $C3$ and $C4$ are in F-cluster Y, and $C1$ and $C3$ are adjacent, $C2$ and $C4$ are adjacent. From the ToD construction, $C1$ and $C3$ will be in one S-cluster, S-cluster I, and $C2$ and $C4$ will be in the other S-cluster, S-cluster II. Because from S-aggregator I and II, F-aggregator X and Y choose one that is closer to the sink as the first S-aggregator, they will choose the same S-aggregator. Therefore their packets can be aggregated at the first S-aggregator, and this completes the proof. □

Though in this section we assume that the size of an event is smaller than the size of the cell, our approach can still work

correctly and perform more efficiently than DAA even if the size of the event is not known in advance. This is because the nodes will use Dynamic Forwarding over ToD only at second phase where the aggregation by DAA is no longer achievable. Therefore at worst our approach just falls back to DAA. Section 5.1 shows that in experiments, ToD improves the performance of DAA by 27% even if the size of the event is greater than the size of a cell.

3.2.3 Clustering and Aggregator Selection

In this paper we use grid-clustering to construct the cells and clusters. Although other clustering methods, such as clustering based on hexagonal or triangular tessellation, can also be used, we do not explore them further in this paper. In principle any clustering would work as long as they satisfy the following conditions. First, the size of the cell must be greater than or equal to the maximum size of an event. Second, the F-cluster and S-cluster must cover the cells that an event may span, and the S-cluster must cover the adjacent cells in different F-clusters.

As opposed to defining an arbitrary clustering, using grid-clustering has two advantages. First, the size of the grid can be easily determined by configuring the grid size as a network parameter. Second, as long as the geographic location is known to the node, the cell, F-cluster and S-cluster it belongs to can be determined immediately without any communication. Geographic information is essential in sensor networks, therefore we assume that sensor nodes know their physical location by configuration at deployment, a GPS device, or localization protocols [39, 40]. As a consequence, all the cells, F-clusters, and S-clusters can be implicitly constructed.

After the grids are constructed, nodes in an F-cluster and S-cluster have to select an aggregator for their cluster. Because the node that acts as the aggregator consumes more energy than other nodes, nodes should play the role of aggregator in turn in order to evenly distribute the energy consumption among all nodes. Therefore the aggregator selection process must be performed periodically. However the frequency of updating the aggregator can be very low, from once in several hours to once in several days, depending on the capacity of the battery on the nodes. Nodes can elect themselves as the cluster-head with probability based on metrics such as the residual energy, and advertise to all nodes in its cluster. In case two nodes select themselves as the cluster-head, the node-id can be used to break the tie.

The other approach is that the nodes in a cluster use a hash function to hash the current time to a node within that cluster, and use that node as the aggregator. Nodes have to know the address of all nodes in its F-cluster and sort them by their node id. A hash function hashes the current time to a number k from 1 to n where n is the number of nodes in its cluster, and nodes use the k^{th} node as the aggregator. Because the frequency of changing the aggregator could be low, the time used could be in hours or days, therefore the time only needs to be coarsely synchronized, and the cluster-head election overhead can be avoided.

However, the *Dynamic Forwarding* approach requires that each F-aggregator knows the location of S-aggregators of S-clusters that its F-cluster overlaps with. Therefore each time the S-aggregator changes, it has to notify the F-aggregators. 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. Instead of selecting a node as the S-aggregator and changing it periodically for an S-cluster, 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 Fig. 10(a), assuming that the sink is located on the bottom-left corner. Therefore as the F-aggregator changes, the corresponding S-aggregator changes as well. When an F-aggregator forwards a packet to an S-aggregator, it forwards the packet toward the Aggregating Cluster of that S-aggregator. When the packet reaches the Aggregating Cluster, nodes in that F-cluster know the location of its F-aggregator and can forward the packet to it. Therefore no aggregator update has to be propagated to neighboring clusters.

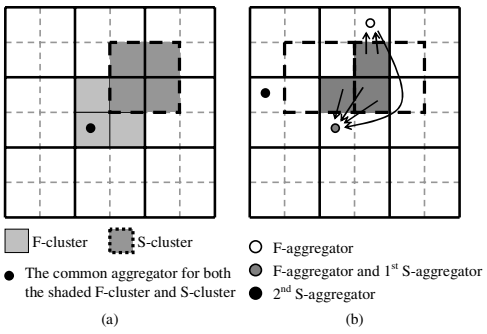


Figure 10. (a) The S-cluster selects the F-cluster closest to the sink among its overlapped F-clusters, assuming that the sink is located at the bottom-left corner of the network. (b) The white F-aggregator selects the F-cluster containing the gray F-aggregator as the aggregating cluster.

Now the role of S-aggregators is passed on to the F-aggregators, and the F-cluster selected by an S-aggregator is the one closer to the sink. When an F-aggregator wants to forward packets to both S-aggregators, it selects the F-cluster that is closer to itself as the aggregating cluster of the first S-aggregator (could be itself) to reduce the number of transmissions between aggregators, as shown in Fig. 10(b). This selection does not affect the property that packets will eventually be aggregated at one aggregator because the S-clusters that cover the cells in two F-clusters are the same, therefore the aggregating cluster selected by two F-aggregators will be the same.

The benefits of using this approach are five-fold. First, no leader election is required for S-clusters, which eliminates the leader election overhead. Second, nodes only need to know the F-aggregator of its F-cluster, which make this approach very scalable. Third, when the F-aggregator changes, the S-aggregator changes as well, but the change does not need to be propagated to other F-clusters or S-clusters. Fourth, if nodes choose the aggregator by hashing current time to get a node id of the aggregator in its cluster,

only nodes within the same F-cluster need to be synchronized with each other. And last, since the Aggregating Clusters of S-clusters are statically computed, there is no packet overhead for computing the Aggregating Clusters.

4 Performance Analysis

In this section we show that the maximum distance between any two adjacent nodes in *ToD* only depends on the size of the cells, and is independent of the size of the network. We ignore the cost from the aggregator to the sink since for perfect aggregation, only one packet will be forwarded to the sink from the aggregator, therefore the cost is comparatively small. Compared to the lower bound $O(\sqrt{n})$ [36] of the grid network for a fixed tree, *ToD* can achieve constant factor even in the worst case.

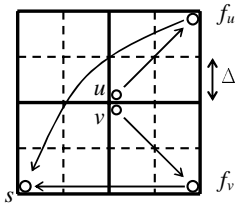


Figure 11. The worst case scenario for ToD.

The worst case in *ToD* is illustrated in Fig. 11 where only two adjacent nodes, u and v, in the corner of two different F-clusters generate packets, and their F-aggregators, f_u and f_v, are located at the opposite corner. We assume a dense deployment of sensor nodes, therefore the distance between two nodes can be transferred to the cost of transmitting a packet between these nodes. Fig. 11 is the worst case since if more nodes are generating packets in one cluster, it will only amortize the cost of sending packets from the F-aggregator to the S-aggregator, and more nodes in multiple F-clusters generating packets will only lower the average distance.

We assume that the length of one side of the cell is Δ, and two nodes are adjacent if their distance is less than a unit of distance. Therefore in Fig. 11 the distance that packets from u and v have to be forwarded before they are aggregated at s is the sum of distances between u to f_u to s and v to f_v to s, and is $(2\Delta\sqrt{2}+4\Delta\sqrt{2})+(2\Delta\sqrt{2}+4\Delta) = 8\Delta\sqrt{2}+4\Delta$. Therefore in the optimal approach, only one transmission is required because u and v are adjacent. In ToD, $8\Delta\sqrt{2}+4\Delta$ number of transmission is required for the worst case.

However, since we use *DAA* as the aggregation technique, packets from adjacent nodes will be aggregated immediately. Therefore for the worst cast to happen, the distance between u and v must be at least 2 units, and our protocol has $4\Delta\sqrt{2}+2\Delta \simeq 7.66\Delta$ times number of transmissions than optimal. The upper bound is only dependent on the size of a cell, and the size of the cell is dependent on the size of an event. This value is independent of the size of the network and therefore is very suitable for large-scale networks.

On average, the number of transmissions will be much less than $4\Delta\sqrt{2}+2\Delta$ because first, typically there will be many nodes generating packets. Second, the distance between a node and its F-aggregator is not always $2\Delta\sqrt{2}$, and the distances between the F-aggregators and the S-aggregator are shorter, too. Third, the *DAA* approach can efficiently aggregate packets from adjacent nodes thereby further reducing the number of transmissions. Therefore we expect the average distance for nodes generating packets to be much less than the worst case.

5 Performance Evaluation

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

5.1 Testbed Evaluation

We conduct experiments with 105 Mica2-based nodes on a sensor testbed. 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 [41] running Linux, which has an Ethernet interface and a serial port for connecting a mote. The Stargates are connected to the server using wired Ethernet. Therefore we can program these motes and send messages and signals to them through Stargates via Ethernet connection. The 105 nodes form a 7×15 grid network with 3 feet spacing. The radio signal using default transmission power covers a lot of nodes in the testbed. In our experiments we do not change the transmission power but limit nodes only to receive packets from nodes within two grid neighbors away, i.e. each node has maximum 12 neighbors.

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. The Anycast MAC implements the RTS-CTS-DATA-ACK for anycast. 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.

First we evaluate the following protocols on the testbed and the codes are available on-line [2]:

- **Dynamic Forwarding over ToD (ToD)**. The semi-structured 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.
- **Data Aware Anycast (DAA)**. The structure-less approach proposed in [20].
- **Shortest Path Tree (SPT)**. Nodes send packets to the sink through the shortest path tree immediately after sensing an event. Aggregation is opportunistic and happens only if two packets are at the same node at the same time. The shortest path tree is constructed immediately after the network is deployed. A message is

[2] http://www.cse.ohio-state.edu/~fank/research/tod.tar.gz

broadcast from the sink and flooded into the network to create a shortest path tree from all nodes to the sink.

- **Shortest Path Tree with Fixed Delay (SPT-D)** Same as the SPT approach, but nodes delay their transmission according to their height in the tree to wait for packets from their children.

Due to the scale of the testbed, in ToD we only divide the network into two F-clusters, which forces the smallest cell to have only 9 sensor nodes. However we do not limit the size of an event to be smaller than the cell size. The event size is larger than the cell size in all following experiments.

We use the normalized number of transmissions as the metric to compare the performance of these protocols. The normalized number of transmissions is the average number of transmissions performed in the entire network to deliver one unit of useful information from sources to the sink. It can be converted to the normalized energy consumption if we know the sending and receiving cost of one transmission, therefore the energy spent on data collection for one packet can be derived. We do not consider energy consumption on idle listening here since all nodes are fully active for all protocols in the experiments and simulations, and the idle energy consumption would be similar for all protocols. To reduce the energy consumption on idle listening, various duty cycling protocols have been proposed. However, due to the page limitation, we are unable to describe how to integrate those works.

Fig. 12 shows the normalized number of transmissions for different event sizes. We fixed the location of the event and vary its diameter from 12 ft 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. Actually DAA can perform best since all packets can be aggregated because all triggered nodes are within transmission range of each other.

All protocols have better performance when the size of the event increases because packets have more chances of being aggregated. ToD performs best 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. In SPT-D, when the event size is smaller, the long stretch effect is more significant than in larger event scenario. When event size is large, for example, two-third of nodes in the network are triggered when the diameter of the event is 36 feet, most of the packets can be aggregated to their parent with one transmission. This indicates that in applications where most nodes are transmitting, the fixed structure such as SPT-D is better, but when only a small subset of nodes are transmitting, their performance degrades because of the long stretch problem.

We notice that the variance of some results in SPT and SPT-D is very high. For example, when the event size is 12 feet in diameter, the maximum normalized number of transmissions in SPT-D is 3.41, and the minimum value is 2.41. By tracing into the detail experiment logs we found that the high variance is because of the different shortest path trees. The tree is re-constructed for each experiment, and therefore may change from experiment to experiment. We found that SPT-D always gets better performance in one tree where all sources are under the same subtree, and performs badly in the other tree where sources are located under two or three different subtrees. This further supports our claims that the long stretch problem in fixed structured approaches affects their performance significantly.

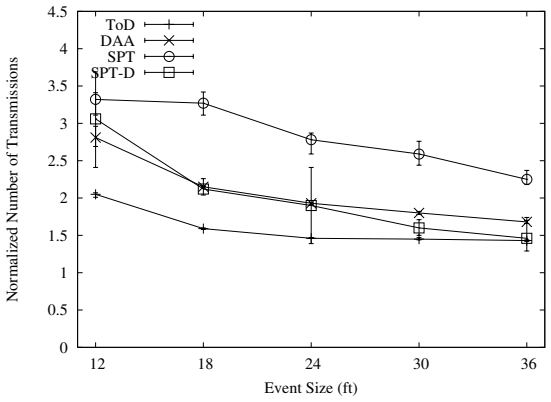


Figure 12. The normalized number of transmissions for different event sizes from experiments on 105 sensors.

The second experiment evaluates the performance of these protocols for different values of maximum delay. We vary the delay from 0 to 8 seconds, and all nodes in the network generate one packet every 10 seconds. Fig. 13 shows the results. As we described, the performance of the structure-based approaches heavily depends on the delay. The SPT-D performs worse than ToD when the maximum delay is less than five seconds, and the performance increases as the delay increases. On the contrary, the performance of ToD and DAA does not change for different delays, which is different from results observed in [20]. We believe that this is because with the default transmission power, a large number of nodes are in interference range when nodes transmit. Therefore even if nodes do not delay their transmissions, only one node can transmit at any given time. Other nodes will be forced to delay, which has the same effect as the Randomized Waiting.

5.2 Large Scale Simulation

To evaluate and compare the performance and scalability of ToD with other approaches requires a large sensor network, which is currently unavailable in real experiments. Therefore we resort to simulations. In this section we use the *ns*2 network simulator to evaluate these protocols. Besides ToD, DAA, and SPT, we evaluate OPT, Optimal Aggregation Tree, to replace the SPT-D protocol.

In OPT, nodes forward their packets on an aggregation tree rooted at the center of the event. Nodes know where to forward packets to and how long to wait. The tree is constructed in advance and changes when the event moves assuming the location and mobility of the event are known.

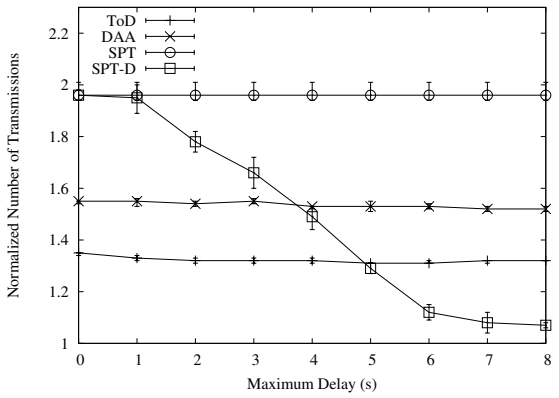


Figure 13. The normalized number of transmissions for different maximum delays from experiments on 105 sensors.

Ideally only $n-1$ transmissions are required for n sources. This is the lower bound for any structure, therefore we use it as the optimal case. This approach is similar to the aggregation tree proposed in [4] but without its tree construction and migration overhead. We do not evaluate SPT-D in simulation because SPT-D is not practical in the large scale network. In the largest simulation scenario, the network is a 58-hop network. According to the simulation in smaller network, SPT-D gets best performance when the delay of each hop is about 0.64 seconds. This makes nodes closer to the sink have about 36 seconds delay in SPT-D, which is not advisable.

We perform simulations of these protocols on a $2000m \times 1200m$ grid network with $35m$ node separation, therefore there are a total of 1938 nodes in the network. The data rate of the radio is $38.4Kbps$ and the transmission range of the nodes is slightly higher than $50m$. An event moves in the network using the random way-point mobility model at the speed of $10m/s$ for 400 seconds. The event size is $400m$ in diameter. The nodes triggered by an event will send packets every five seconds to the sink located at $(0,0)$. The aggregation function evaluated here is perfect aggregation, i.e. all packets can be aggregated into one packet without increasing the packet size.

5.3 Event Size

We first evaluate the performance for these protocols on different number of nodes generating the packets. This simulation reflects the performance of each protocol for different event sizes. We study the performance for 4 mobility scenarios and show the average, maximum, and minimum values of the results.

Fig. 14(a) shows the result of normalized number of transmissions. ToD improves the performance of DAA and SPT by 30% and 85%, and is 25% higher than OPT. However OPT has the best performance by using the aggregation tree that keeps changing when event moves but its overhead is not considered in the simulation. SPT has very poor performance since its aggregation is opportunistic. Except the SPT, the performance of all other protocols is quite steady. This shows that they are quite scalable in terms of the event size.

Fig. 14(b) and (c) show the total number of transmissions and total units of useful information received by the sink. As observed in [20], 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.

5.4 Scalability

In this set of simulations we evaluate the scalability of our protocol since our goal is to design a scalable data aggregation protocol. If a protocol is not scalable, its performance will degrade as the size of the network increases.

To evaluate the scalability of a protocol, we limit an event to move only in a bounded region at a certain distance from the sink to simulate the effect of different network sizes. We limit an event to move within a $400m \times 1200m$ rectangle, and change the distance of the rectangle to the sink from $200m$ to $1400m$, as shown in Fig. 15. In order to be fair to all scenarios, we limit the event not to move closer than $200m$ to the network boundary such that the number of nodes triggered by the event does not change drastically.

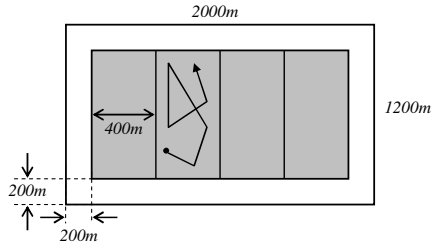
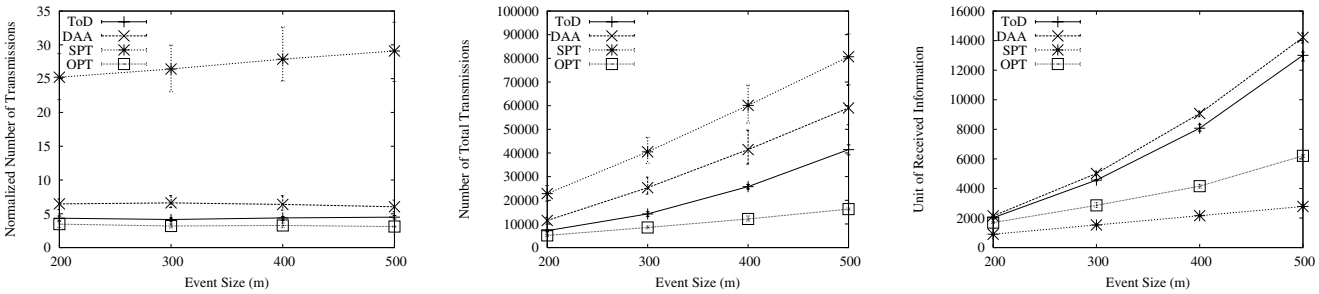


Figure 15. The simulation scenario for scalability. The event is limited to move only within a small gray rectangle in each simulation.

Fig. 16 shows the results of the scalability simulation. The performance of ToD and OPT remains steady, and ToD is 22% higher than OPT. This shows that ToD is quite scalable as its performance does not degrade as the size of the network increases. 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 rectangle (to the sink) to the farthest rectangle.

Fig. 16(c) 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. Though 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.

Ideally the number of received packets at sink is 1, if all packets can be aggregated at the aggregator. However the number of received packets at sink is higher than 1 in ToD



(a) Normalized number of transmissions (b) Number of transmissions (c) Unit of received information

Figure 14. The simulation results for different event sizes.

and OPT. This is because the delay in CSMA-based MAC protocol can not be accurately predicted therefore the aggregator might send the packet to the sink before all packets are forwarded to it. Though the cost of forwarding the unaggregated packets from aggregator to the sink in ToD and OPT also increases when the size of the network increases, the increase is comparably smaller than DAA and SPT because few packets are forwarded to the sink without aggregation.

We observe that the number of received packets at the sink in ToD is higher when the event is closer to the sink. In our simulation, nodes in the same F-cluster as the sink will always use sink as the F-aggregator because we assume that the sink is wire powered therefore there is no need to delegate the role of aggregator to other nodes in order to evenly distribute the energy consumption.

5.5 Cell Size

The above simulations use the maximum size of an event as the cell size. As we described in Section 3.2.2, this ensures that the Dynamic Forwarding can aggregate all packets at an S-aggregator, and the cost of forwarding the aggregated packets to the sink is minimized. However, large cell size increases the cost of aggregating packets to the aggregator as we use DAA as the aggregation technique in an F-cluster. In this section we evaluate the impact of the size of a cell on the performance of ToD.

We vary the cell size from $50m \times 50m$ to $800m \times 800m$ and run simulations for three different event sizes, which are $200m$, $400m$, and $600m$ in diameter. The results are collected from five different event mobility patterns and shown in Fig. 17.

When the size of cell is larger than the event size, the performance is worse because the cost of aggregating packets to F-aggregator increases, but the cost of forwarding packets from S-aggregator does not change. When the size of cell is too small, the cost of forwarding packets to sink increases because packets will be aggregated at different F-aggregators and more packets will be forwarded to the sink without further aggregation. In general, when the size of the F-cluster is small enough to only contain one node, or when the size of the F-cluster is large enough to include all nodes in the network, ToD just downgrades to DAA.

ToD has the best performance when the cell size is $100m \times 100m$ (F-cluster size is $200m \times 200m$) when the event size is $200m$ in diameter. When the diameter of an event is $400m$ and $600m$, using $200m \times 200m$ as the cell size has the best performance (F-cluster size is $400m \times 400m$). This shows that the ToD performance can be further optimized by selecting the appropriate cell size. To explore the relation between the event and cell size for optimization will be part of our future work.

6 Conclusion

In this paper we propose a semi-structured approach that locally uses a structure-less 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 construction and maintenance of dynamic structures. We evaluate its performance using real experiments on a testbed of 105 sensor nodes and simulations on 2000 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.

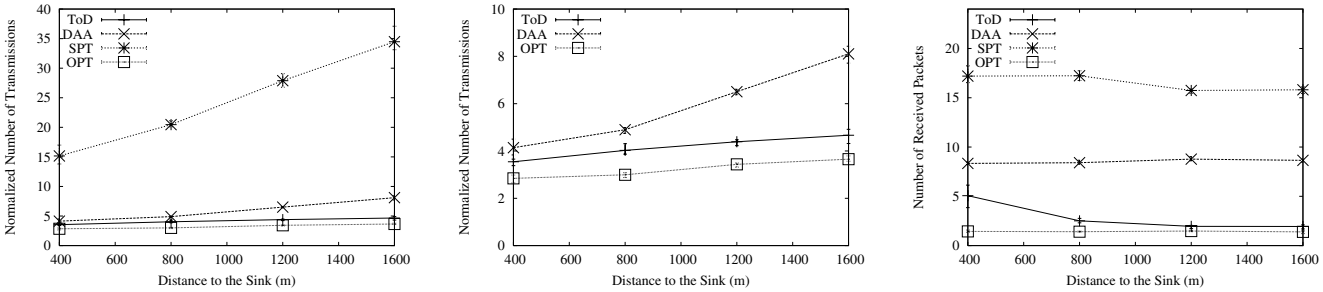
7 References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 2, January 2000.

[2] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," in *IEEE Transactions on Wireless Communications*, vol. 1, October 2002, pp. 660–670.

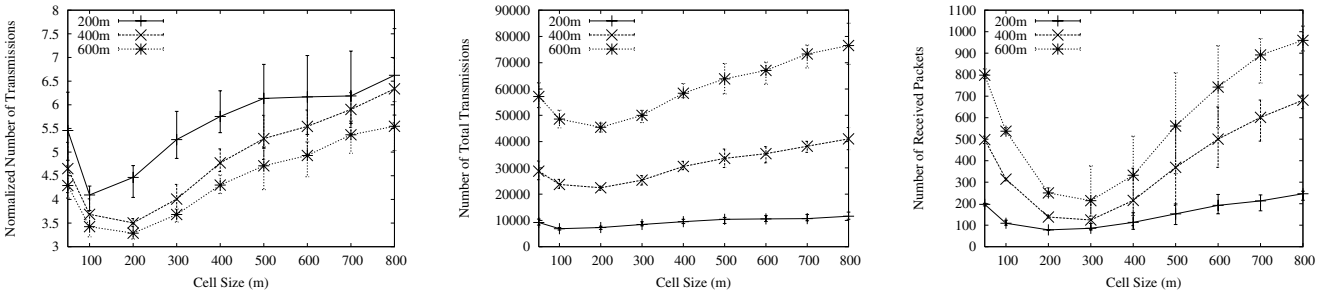
[3] C. Intanagonwiwat, D. Estrin, and R. Goviindan, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," in *Technical Report 01-750, University of Southern California*, November 2001.

[4] W. Zhang and G. Cao, "Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks," in *Proceedings of INFOCOM 2004*, vol. 4, March 2004, pp. 2434–2445.



(a) Normalized number of transmission (b) Zoom in of Fig. 16(a) (c) Number of received packets

Figure 16. **The simulation results for difference distances from the event to the sink.**



(a) Normalized number of transmissions (b) Number of transmissions (c) Number of received packets

Figure 17. **The simulation results for difference cell sizes.**

[5] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-based Collaboration for Target Tracking in Sensor Networks," in *IEEE Transactions on Wireless Communications*, vol. 3, September 2004, pp. 1689–1701.

[6] M. Ding, X. Cheng, and G. Xue, "Aggregation Tree Construction in Sensor Networks," in *Proceedings of the 58th IEEE Vehicular Technology Conference*, vol. 4, October 2003, pp. 2168–2172.

[7] H. Luo, J. Luo, and Y. Liu, "Energy Efficient Routing with Adaptive Data Fusion in Sensor Networks," in *Proceedings of the Third ACM/SIGMOBILEe Workshop on Foundations of Mobile Computing*, August 2005.

[8] A. Goel and D. Estrin, "Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[9] "Networked Infomechanical Systems," http://www.cens.ucla.edu .

[10] "Center for Embedded Networked Sensing at UCLA," http://www.cens.ucla.edu .

[11] J. Polastre, "Design and Implementation of Wireless Sensor Networks for Habitat Mon itoring," Master's Thesis, University of California at Berkeley, Spring 2003.

[12] A. Mainwaring, R. Szewczyk, J. Anderson, and J. Polastre, "Habitat Monitoring on Great Duck Island," http://www.greatduckisland.net.

[13] A. Arora, P. Dutta, and S. Bapat, "Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking," OSU-CISRC-12/03-TR71, 2003.

[14] "ExScal," http://www.cast.cse.ohio-state.edu/exscal/.

[15] S. Corporation, "Chemical/Bio Defense and Sensor Networks," http://www.sentel.com/html/chemicalbio.html .

[16] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[17] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On Network Correlated Data Gathering," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2571–2582.

[18] K. W. Fan, S. Liu, and P. Sinha, "Structure-free Data Aggregation in Sensor Networks," in *OSU-CISRC-4/06-TR35, Technical Report, Dept of CSE, OSU*, April 2006.

[19] Y. Zhu, K. Sundaresan, and R. Sivakumar, "Practical Limits on Achievable Energy Improvements and

Useable Delay Tolerance in Correlation Aware Data Gathering in Wireless Sensor Networks," in *IEEE Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, September 2005.

[20] K. W. Fan, S. Liu, and P. Sinha, "On the potential of Structure-free Data Aggregation in Sensor Networks," in *To be appear in Proceedings of INFOCOM 2006*, April 2006.

[21] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, August 2000, pp. 56–67.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed Diffusion for Wireless Sensor Networking," in *IEEE/ACM Transactions on Networking*, vol. 11, February 2003, pp. 2–16.

[23] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *Proceedings of the 5th symposium on Operating systems design and implementation*, December 2002, pp. 131–146.

[24] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, "Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks," in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, June 2004, pp. 49–58.

[25] S. Lindsey, C. S. Raghavendra, and K. M. Sivalingam, "Data Gathering in Sensor Networks using the Energy*Delay Metric," in *Proceedings 15th International Parallel and Distributed Processing Symposium*, April 2001, pp. 2001–2008.

[26] S. Lindsey and C. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *Proceedings of IEEE Aerospace Conference*, vol. 3, March 2002, pp. 1125–1130.

[27] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks Using Energy Metrics," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, September 2002, pp. 924–935.

[28] J. Wong, R. Jafari, and M. Potkonjak, "Gateway placement for latency and energy efficient data aggregation," in *29th Annual IEEE International Conference on Local Computer Networks*, November 2004, pp. 490–497.

[29] B. J. Culpepper, L. Dung, and M. Moh, "Design and Analysis of Hybrid Indirect Transmissions (HIT) for Data Gathering in Wireless Micro Sensor Networks," in *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, January 2004, pp. 61–83.

[30] H. F. Salama, D. S. Reeves, and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-time Communication on High-speed Networks," in *IEEE Journal on Selected Area in Communications*, vol. 15, April 1997.

[31] A. Scaglione and S. D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, September 2002, pp. 140–147.

[32] A. Scaglione, "Routing and Data Compression in Sensor Networks: Stochastic Models for Sensor Data that Guarantee Scalability," in *Proceedings of IEEE International Symposium on Information Theory*, June 2003, p. 174.

[33] R. Cristescu and M. Vetterli, "Power Efficient Gathering of Correlated Data: Optimization, NP-Completeness and Heuristics," in *Summaries of MobiHoc 2003 posters*, vol. 7, July 2003, pp. 31–32.

[34] S. Pattern, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, April 2004, pp. 28–35.

[35] L. Cai and D. Corneil, "Tree Spanners," in *SIAM Journal of Discrete Mathematics*, vol. 8, 1995.

[36] N. Alon, R. M. Karp, D. Peleg, and D. West, "A graph theoretic game and its application to the k-server problem," in *SIAM Journal of Computing*, vol. 24, 1995.

[37] D. Peleg and D. Tendler, "Low Stretch Spanning Trees for Planar Graphs," in *Technical Report MCS01-14, Mathematics & Computer Sience, Weizmann Institute Of Sience*, 2001.

[38] L. Jia, G. Noubir, R. Rajaraman, and R. Sundaram, "GIST: Group-Independent Spanning Tree for Data Aggregation in Dense Sensor Networks," in *International Conference on Distributed Computing in Sensor Systems*, June 2006.

[39] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Localization For Very Small Devices," in *IEEE Personal Communications, Special Issue on "Smart Spaces and Environments"*, vol. 7, October 2000.

[40] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust Distributed Network Localization with Noisy Range Measurements," in *Proceedings of 2nd ACM Sensys*, pp. 50–61.

[41] Crossbow, "Crossbow," http://www.xbow.com.