

Distributed Online Data Aggregation for Large Scale Sensor Networks

Kai-Wei Fan and Prasun Sinha

Department of Computer Science and Engineering
The Ohio State University
Email: {fank, prasun}@cse.ohio-state.edu

Abstract—To benefit from data aggregation in large scale sensor networks, an aggregation point, i.e. the place where data are aggregated, must be close to sources. In event triggered sensor networks, this can be achieved by dynamically constructing a tree connecting the sources rooted at a nearby node. However, this incurs high control and maintenance overhead. With static trees, the distance (Δ) between sources and the aggregation point can be as high as $O(n)$ [1] where n is the number of nodes in the network. This diminishes the benefit of data aggregation, thereby limiting the scalability of static trees. In this paper we propose AFT, a structure with multi-level overlapping clusters. Packet forwarding decisions on AFT are made on the fly when packets are being forwarded and it bounds the distance between the aggregation point and sources by $O(\delta)$ irrespective of network size, where δ is the diameter of the event. This guarantees that packets can be aggregated near sources without the overhead of constructing a dynamic structure and therefore is scalable. We prove that in the worst case, AFT guarantees aggregation at a node that is at most $2(1 + \sqrt{13})\delta$ away from the sources.

I. INTRODUCTION

With advances in sensor and wireless technologies, large-scale city-wide deployment of sensor networks will become feasible in the near future. Such sensor networks can provide platforms for various applications, including fire detection [2], vehicle tracking [3], and biochemical hazards detection [4]. Sensors collaborate on sensing and reporting tasks which provide an autonomous surveillance system to monitor our surrounding environment. One key to the success for such a large sensor network deployment will be scalability since sensors are highly resource constrained devices.

In this paper we focus on data aggregation for event-based applications in large-scale sensor networks. 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.

Various data aggregation approaches have been proposed for data gathering applications and event-based applications. In data gathering applications, such as environment and habitat monitoring [5], [6], nodes periodically report sensed data to the sink. As the traffic pattern is unchanging, fixed structure-based approaches [7]–[17] incur low maintenance overhead and are

therefore suitable for such applications. However, in event-based applications, such as intrusion detection [18], [19] and biochemical hazard detection [4], the sources are not known in advance. Therefore the approaches that use fixed structures can not efficiently aggregate data [17], while the approaches that change the structure dynamically incur high maintenance overhead [20], [21]. More recently two other paradigms for aggregation have been proposed, namely, structure-free [22], [23] and semi-structure [24] approaches. However the structure-free approach is not scalable in large sensor networks [24] and the semi-structure approach requires the knowledge of maximum event size for optimal performance.

Observing the insufficiency of current approaches, we propose AFT, *Alternative Forwarding Tree*, for event-based applications to guarantee scalable data aggregation irrespective of network size, event size, and event location. AFT uses a multi-level, interleaved cluster-based structure, with exponentially increasing cluster size at each level. At each level, packets will be forwarded to an upper level cluster which covers adjacent clusters that have packets for aggregation. Forwarding decisions are made by the nodes on the fly solely based on local information. This guarantees that packets will be aggregated near the sources without incurring high control overhead as dynamic-structured approaches do, and therefore is scalable to any network or event size. This paper makes the following contributions:

- We propose a scalable data aggregation structure that achieves early aggregation for event-based applications.
- We prove that the distance the packets traveled before they are aggregated is bounded by a constant factor, which is $2(1 + \sqrt{13})$, of the event diameter.
- We conduct extensive large-scale simulations and show that AFT does guarantee the bound for aggregation.

The organization of the rest of the paper is as follows. Section II presents background and related work. Section III presents the AFT structure and forwarding rules. Section IV discusses implementation and design issues. The performance evaluation of the protocols using simulations is presented in Section V. Finally Section VI concludes the paper.

II. RELATED WORK

Current works on forwarding data to facilitate data aggregation can be sorted into three categories: static structure

[7]–[14], [16], [17], dynamic structure [20], [21], [25]–[27] and structure-free/semi-structure approaches [22]–[24]. In this section we briefly review the pros and cons for each category.

Static structure approaches create a tree structure in advance to forward packets. As the tree is static, it incurs low maintenance overhead. If sources are known in advance, an optimal tree can be constructed for data aggregation. However in event triggered networks, sources are not known in advance. A tree might have long stretch between adjacent nodes [28] [29]. 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 are aggregated. It has been shown that for any graph, the lower bound of the average stretch is $O(\log(n))$ [29], and it can be as high as $O(n)$ for the worst case [1].

Dynamic structure-based approaches create a tree dynamically when forwarding packets to the sink. As the structure is created dynamically, it can be optimized according to the locations of the sources. However in mobile event scenarios, sources change as the event moves, and the structure has to be adjusted to accommodate the new set of sources. The adjustment involves heavy message exchanges which might offset the benefit of aggregation in large-scale networks. In spatial database field, such as [25]–[27], extensive query message propagation or communication between sources incurs high control overhead. In GIT [20] which based on Directed Diffusion, the interests have to be flooded to entire network periodically, even if there is no event. For DCTC, the energy consumption of tree expansion, pruning and reconfiguration is about 33% of the data collection [21].

DAA [22] is the first proposed structure-free data aggregation protocol that can achieve high aggregation without incurring the overhead of structure-based 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 failed to get aggregated will negate the benefit of energy savings resulted from eliminating the control overhead.

In order to get benefit from structure-free approach even in large networks, ToD [24] uses an implicit structure to forward packets that are not aggregated by structure-free approach. ToD uses a flat and interleaved clustered structure to create a multi-tree graph, and forwards packets on one of the trees based on where packets originated from. This guarantees that packets will be aggregated near the sources. However, ToD requires the knowledge of maximum event size to partition a network into clusters for optimal performance, which limits its applicability.

AFT is different from the above approaches as AFT guarantees the aggregation of packets within a fixed distance to the sources without incurring high control overhead. The distance between the aggregation point and the sources in AFT is a small constant factor of event size. This property makes AFT

a scalable structure for event-based applications.

III. AFT - ALTERNATIVE FORWARDING TREE

In this section we describe the construction of AFT, Alternative Forwarding Tree, and forwarding rules on AFT. The goal of AFT is to *guarantee aggregation of packets near sources for event-triggered applications irrespective of event size, shape, and location, without the overhead of constructing a dynamic structure*. AFT achieves this goal by forwarding packets on a fixed hybrid structure with low maintenance overhead. First we briefly describe ToD, a multi-tree static structure which can achieve the same goal with fixed event size.

A. Tree-on-DAG

Tree-on-DAG (ToD) [24] is a scalable data aggregation structure. ToD guarantees the aggregation of packets near sources without incurring communication overhead for creating a dynamic structure. In ToD, the network is partitioned into cells, first-level clusters (F-clusters), and second-level clusters (S-clusters), as shown in Fig. 1. When nodes are triggered by an event, they collect readings of the event and send these readings to their F-clusterhead. When a F-clusterhead receives these packets, it can learn which cells these packets originated from. Because ToD assumes a maximum event size and defines a cell to be greater than the maximum event size, an event will only trigger nodes in the same cluster or only adjacent clusters. Therefore an F-clusterhead can conjecture which cluster might cover the event, and forward the aggregated packet accordingly for further aggregation. In [24] four basic forwarding rules are defined to guarantee that packets can be fully aggregated within two steps of forwarding. These forwarding decisions are made solely based on the received data packets, without regarding any communication between neighboring nodes.

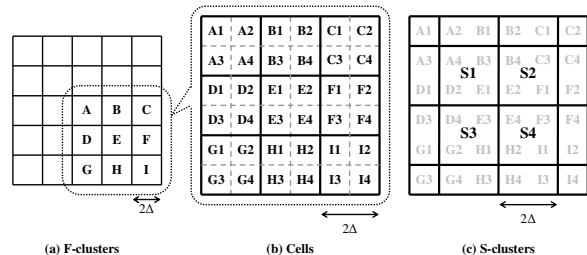


Fig. 1. F-clusters, cells, and S-clusters in ToD. Δ is the diameter of maximum event size. (a) The network is divided into 5×5 F-clusters. (b) Each F-cluster contains four cells. For example the F-cluster A in (a) contains cell $A1$, $A2$, $A3$, and $A4$. (c) The S-clusters have to cover all adjacent cells in different F-clusters. Each S-cluster contains four cells from four different F-clusters.

However, ToD guarantees early aggregation when the maximum event size is known. For some applications, such as intrusion detection, the event size can be determined because it depends on the sensing range of equipped sensors, such as PIR and magnetometer [30]. For other applications, such as fire detection, the size of an event can not be determined. ToD does not perform well when the event size is not known in advance. This leads us to design the AFT, a more flexible and low

control overhead protocol that guarantees early aggregation irrespective of event size.

B. Alternative Forwarding Tree

Alternative Forwarding Tree (AFT) is a multi-level structure that recursively splits nodes into different overlapping clusters at different levels based on their locations. At each level, a Q-cluster is composed of four Q-clusters at lower level. An A-cluster is also composed of four Q-clusters at lower level, but Q-clusters and A-clusters at the same level are overlapping with each other. The Q-clusters resemble the hierarchy of a Quad-tree [31] therefore they are named Q-clusters. A-clusters serve as *Alternative Clusters* to provide alternative choices for packet forwarding and therefore are named A-clusters. Before we start describing the construction of AFT, we define these two terms that will be used throughout the paper.

Definition 1: Let l be the number of levels of an AFT. $Q_{i,j}$ is the j^{th} Q-cluster at level i , $1 \leq i \leq l$ and $A_{i,j}$ is the j^{th} A-cluster at level i , $2 \leq i < l$. When a specific cluster at level i is not of concern, we use Q_i and A_i to represent a Q-cluster and A-cluster at level i .

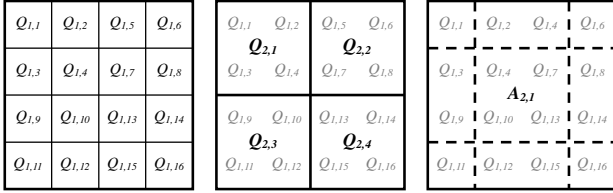


Fig. 2. The illustration for Q-clusters and A-clusters in a 3-level AFT. $Q_{i,j}$ is a Q-cluster at level i and $A_{i,j}$ is an A-cluster at level i . A-clusters interleave with Q-clusters at the same level

A_i clusters are of the same size as Q_i (except for boundary A_i clusters), but they are interleaved as shown in Fig. 2¹. Each A_i covers four Q_{i-1} from four different Q_i . Therefore each Q_i has two parents, one Q_{i+1} and one A_{i+1} cluster.

For each A_i , there are three cases. (a) It is fully covered by a Q_{i+1} cluster (Fig. 3a). (b) It is fully covered by an A_{i+1} cluster (Fig. 3b). (c) It is covered by two Q_{i+1} clusters, $Q_{i+1,a}$ and $Q_{i+1,b}$, also by two A_{i+1} clusters, $A_{i+1,c}$ and $A_{i+1,d}$. (Fig. 3c). For case (a) and (b), the A_i just selects the Q_{i+1} (case (a)) or A_{i+1} (case (b)) which covers it, as its parent respectively. For case (c), the A_i will select the two Q-clusters and two A-clusters that cover it as its parent clusters.

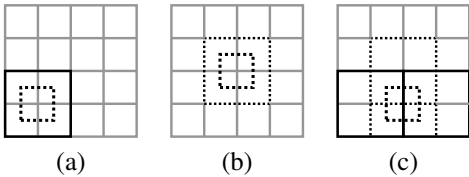


Fig. 3. The three possibilities of selecting parents for an A-cluster.

The overview of a four level AFT is shown in Fig. 4. It is a directed acyclic graph composed of multiple overlapping

¹Throughout the paper, solid lines are used to indicate Q-clusters and dotted/dashed lines are used to indicate A-clusters

trees. At each level packets alternate between Q-clusters and A-clusters, thus the name Alternative Forwarding Tree, AFT. Note that switching between Q-clusters and A-clusters may not happen at each level and is governed by the forwarding rules to be discussed later.

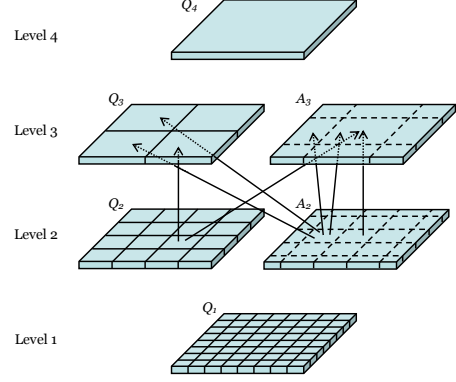


Fig. 4. The overview of a four level AFT for a square network. Each slab represents a partition of the entire network at different levels for either Q-clusters or A-clusters. Each cluster at level i has one to four parents at level $i + 1$. For example, in this figure, a Q_2 cluster has two parents, a Q_3 and an A_3 ; an A_2 cluster has one A_3 parent, and another A_2 cluster has four parents (shown using arrows).

C. Alternative Forwarding on AFT

In AFT, nodes first aggregate their packets within their level one Q-clusters. After that, packets will be forwarded up on the AFT for further aggregation of packets from different clusters. In this section we define the forwarding rules used by AFT that are designed to guarantee that the number of steps of forwarding packets will be bounded by a constant factor of the event diameter.

1) *Assumptions:* First, we assume that there is a cluster-head in each cluster at each level and each cluster-head knows its parent cluster-heads in the higher level. Second, the cluster-head has the ability to know which neighboring clusters at the same level have packets for aggregation. We will describe how cluster-heads achieve these in more details in Section IV. Third we assume that an event triggers nodes in contiguous clusters. For contiguous we mean that if we create a graph with level one Q-clusters as its vertices, there is an edge between two vertices if these two vertices represent two adjacent Q_1 clusters, the vertices which represent Q_1 clusters that are triggered by an event are connected. Here two Q_1 clusters, $Q_{1,i}$ and $Q_{1,j}$, are adjacent only if $Q_{1,j}$ is the left, right, top, or bottom adjacent cluster of $Q_{1,i}$.

2) *AFT Forwarding Rules:* The aggregators will forward packets based on which neighboring clusters have packets. At each level, aggregators will forward packets only to an upper level cluster that covers some of neighboring clusters that have packets. The intuition is, if packets are forwarded to an upper level cluster that covers a neighboring cluster that has packets, their packets will be aggregated if both clusters forward their packets to that upper level cluster. Due to the construction of AFT, such an upper level cluster always

exists for two neighboring clusters, either a Q-cluster or an A-cluster. The challenge is how to forward packets solely based on information collected from data packets, not on extensive communications between neighboring nodes, so the forwarding decisions can be made on the fly. Since clusters have only local view of which neighboring clusters have packets, they must follow identical forwarding rules based on local information to achieve global aggregation. We define some other terms we used in describing the forwarding rules.

Definition 2: $P_Q(X_i)$ is a Q_{i+1} cluster which is a parent of X_i . $P_A(X_i)$ is a A_{i+1} cluster which is a parent of X_i .

Definition 3: $N_Q(Q_{i,j})$ is $Q_{i,j}$'s neighboring Q_i clusters whose parent is $P_Q(Q_{i,j})$, i.e. sibling Q-clusters at level i with the same Q-cluster parent at level $i + 1$. $N_A(Q_{i,j})$ is $Q_{i,j}$'s neighboring Q_i clusters whose parent is $P_A(Q_{i,j})$, i.e. sibling Q-clusters at level i with the same A-cluster parent at level $i + 1$.

Algorithm 1 is the pseudo-code for the AFT forwarding rules. For a $Q_{i,j}$ cluster, there are three scenarios:

- 1) **QR1:** No $Q_i \in N_Q(Q_{i,j}) \cup N_A(Q_{i,j})$ has packets: In this case, the event only triggers nodes in $Q_{i,j}$, and all packets will be aggregated at the aggregator of $Q_{i,j}$. Thus the packet can be forwarded to the sink directly.
- 2) **QR2:** At least one $Q_{i,k} \in N_Q(Q_{i,j})$ has packets: In this case, $Q_{i,j}$ and $Q_{i,k}$ have the same Q_{i+1} parent cluster. Packets are forwarded to the aggregator of $P_Q(Q_{i,j})$.
- 3) **QR3:** At least one $Q_{i,k} \in N_A(Q_{i,j})$ has packets: In this case, $Q_{i,j}$ and $Q_{i,k}$ have the same A_{i+1} parent cluster. Packets are forwarded to the aggregator of $P_A(Q_{i,j})$.

For an $A_{i,j}$ cluster, there are three scenarios:

- 1) **AR1:** It receives packets from all child Q_{i-1} clusters that have packets: In this case, all packets will be aggregated by the aggregator, and the aggregator will send the aggregated packets to the sink directly.
- 2) **AR2:** It only receives packets from some of its child Q_{i-1} clusters that have packets, and it has only one parent: The aggregator will forward packets to its parent cluster.
- 3) **AR3:** As in AR2, but it has four parents, two Q_{i+1} clusters and two A_{i+1} clusters: There are three sub cases in this scenario: (**AR3.1**) If all child Q_{i-1} clusters that have packets are only covered by one of the Q_{i+1} clusters, as shown in Fig. 5a, forward packets to that Q_{i+1} cluster. (**AR3.2**) If all child Q_{i-1} clusters that have packets are only covered by one of the A_{i+1} clusters, as shown in Fig. 5b, forward packets to that A_{i+1} cluster. (**AR3.3**) The child Q_{i-1} clusters that have packets may be covered by two Q_{i+1} parent clusters, as shown in Fig. 5c. In this case, forward packets to the Q_{i+1} parent cluster which covers the Q_{i-1} clusters that have packets but are not received by $A_{i,j}$, or randomly forward packets to one of its two Q_{i+1} parent clusters if both cover such Q_{i-1} .

Algorithm 1 Alternative Forwarding Rules

// Def: $c_i \subset X$: child cluster i is covered by a cluster X

Each aggregator maintains following variables

$Rcv[1..4]$: i^{th} value is 1 if pkts have been received from c_i

$Exp[1..4]$: i^{th} value is 1 if child cluster c_i has pkts

$Nbr[1..8]$: i^{th} value is 1 if neighboring cluster i has pkts

procedure *AltForwarding*(C)

```

1: if  $C$  is a  $Q$  cluster then
2:   if ( $Nbr[i] = 0, \forall i$ ) then
3:     Forward to sink
4:   else if ( $\exists i : Nbr[i] \neq 0 \ \& \ i \in N_Q(C)$ ) then
5:     Forward to  $P_Q(C)$ 
6:   else
7:     Forward to  $P_A(C)$ 
8:   end if
9: else
10:   $Missing[] \leftarrow Exp[] \ \& \ !Rcv[]$ 
11:  if ( $Missing[i] = 0, \forall i$ ) then
12:    Forward to sink
13:  else if ( $C$  has only one parent) then
14:    Forward to  $C$ 's parent cluster
15:  else
16:    //  $Q_{i+1,a}, Q_{i+1,b}, A_{i+1,c}$ , and  $A_{i+1,d}$  are  $C$ 's parents
17:    if ( $\forall i$  where  $Exp[i] = 1, c_i \subset Q_{i+1,a}$ ) then
18:      Forward to  $Q_{i+1,a}$ 
19:    else if ( $\forall i$  where  $Exp[i] = 1, c_i \subset Q_{i+1,b}$ ) then
20:      Forward to  $Q_{i+1,b}$ 
21:    else if ( $\forall i$  where  $Exp[i] = 1, c_i \subset A_{i+1,c}$ ) then
22:      Forward to  $A_{i+1,c}$ 
23:    else if ( $\forall i$  where  $Exp[i] = 1, c_i \subset A_{i+1,d}$ ) then
24:      Forward to  $A_{i+1,d}$ 
25:    else
26:      Forward to  $Q_{i+1,a}$  or  $Q_{i+1,b}$  depending on which one
        covers at least some  $c_i$  where  $Missing[i] = 1$ 
27:    end if
28:  end if
29: end if

```

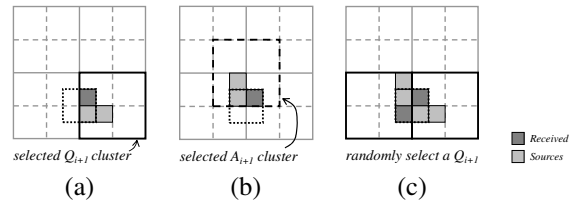


Fig. 5. The forwarding decisions for an A-cluster with four parents. The aggregator receives packets only from dark gray clusters. (a) The aggregator selects the Q_{i+1} cluster that covers all Q_{i-1} clusters that have packets. (b) The aggregator selects the A_{i+1} cluster that covers all Q_{i-1} clusters that have packets. (c) The aggregator randomly selects one Q_{i+1} cluster that covers a Q_{i-1} cluster that has packets but whose data has not been received.

Following these forwarding rules at each level, the packets will be forwarded between the Q-clusters and A-clusters to achieve early aggregation without control overhead. In the next section we will show that using these rules, packets can be aggregated at or before level $i + 2$ cluster if the size of the area of sources can be covered by the size of a level i cluster.

D. Guaranteed Early Aggregation

In this section we are going to show that if the area of an event can be covered by a cluster of size equal to the size

of clusters at level i , the packets can be aggregated within a cluster before or at level $i + 2$.

Property 1: In AFT, at each level, packets will only be forwarded to clusters that cover at least part of the event.

This is evident because the forwarding rules in Section III-C for Q-clusters and A-clusters at each level will only forward packets to their parent clusters that cover at least some of clusters that have packets.

Definition 4: A most constrained cluster, Q_i or A_i , for an event is the smallest Q-cluster or A-cluster that covers entire area of the event.

Lemma 1: If the most constrained cluster is a Q-cluster Q_i at level i , the packets can be aggregated at Q_i .

Proof: As shown in Fig. 6, suppose Q_i is the most constrained cluster. Because of Property 1, at level $i - 1$, packets can only come from four Q_{i-1} clusters, $Q_{i-1,1}$ to $Q_{i-1,4}$, and five A_{i-1} clusters, $A_{i-1,1}$ to $A_{i-1,5}$. Packets can not come from $A_{i-1,6}$ to $A_{i-1,9}$ because packets will be forwarded to these four clusters only if some Q_{i-2} clusters in these four A_{i-1} clusters but not covered by the Q_i have packets, which violates that Q_i is the most constrained cluster.

For Q_{i-1} clusters, at least two Q_{i-1} clusters have packets because Q_i is the most constrained cluster. Therefore packets from Q_{i-1} will be forwarded to Q_i (Rule QR2).

For $A_{i-1,5}$, because of the construction of AFT, it has only one parent cluster, which is Q_i . Therefore its packets will be forwarded to Q_i (Rule AR2).

For $A_{i-1,1}$ to $A_{i-1,4}$, if they receive packets, we show that both of its Q_{i-2} child clusters that are covered by Q_i must have packets. Take $A_{i-1,3}$ as an example. If only one of the two Q_{i-2} clusters, say $Q_{i-2,j}$, has packets, one of the $N_Q(Q_{i-2,j})$ not covered by the $A_{i-1,3}$ cluster, say $Q_{i-2,k}$, must have packets. This is because the event is contiguous and Q_i is the most constrained cluster. According to rule QR2, $Q_{i-2,j}$ will forward packets to the $Q_{i-1,2}$ cluster, not the $A_{i-1,3}$ cluster. Therefore both of its Q_{i-2} child clusters covered by the Q_i must have packets, and one of which forwards packets to the $A_{i-1,3}$. Therefore the packets will be forwarded to Q_i (Rule AR3.1), and this completes the proof. ■

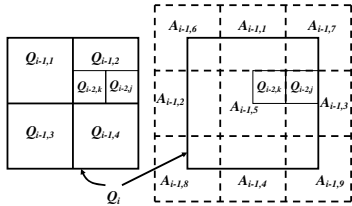


Fig. 6. All possible Q_{i-1} and A_{i-1} that might have packets for Q_i .

Lemma 2: If the most constrained cluster is an A-cluster A_i at level i , the packets can be aggregated at A_i .

Proof: As shown in Fig. 6, but now the Q_i is an A_i cluster and is the most constrained cluster. Packets can only come from four Q_{i-1} clusters (which are covered by four different Q_i clusters), $Q_{i-1,1}$ to $Q_{i-1,4}$, or five A_{i-1} clusters, $A_{i-1,1}$

to $A_{i-1,5}$ because of Property 1. Using the same argument as in Lemma 1, packets can not come from $A_{i-1,6}$ to $A_{i-1,9}$. Because A_i is the most constrained cluster, no cluster in $N_Q(Q_{i-1,j})$ (sibling clusters with the same Q_i cluster parent), for $1 \leq j \leq 4$, will have packets, and at least two of the Q_{i-1} clusters will have packets. Therefore their packets will be forwarded to A_i (Rule QR3).

Using the same argument as in Lemma 1, packets from $A_{i-1,5}$ will be forwarded to A_i (Rule AR2) and packets from $A_{i-1,1}$ to $A_{i-1,4}$ will be forwarded to A_i (Rule AR3.2), and this completes the proof. ■

Lemma 3: For an event of size at most the size of Q_i , it is fully covered by a cluster at level at most $i + 2$.

Proof: If the event E is fully covered by any Q_{i+2} or A_{i+2} cluster, the proof is done. Suppose that event E of size at most the size of Q_i is not fully covered by any $i + 2$ level cluster. E must overlap with the boundary of some Q_{i+2} and A_{i+2} clusters, as shown in Fig. 7a. Let their intersection point be Y . Let $A_{i+1,k}$ be the cluster that contains Y . As the event contains Y and its size is not larger than the size of Q_i , it is fully contained in $A_{i+1,k}$ cluster. This completes the proof. ■

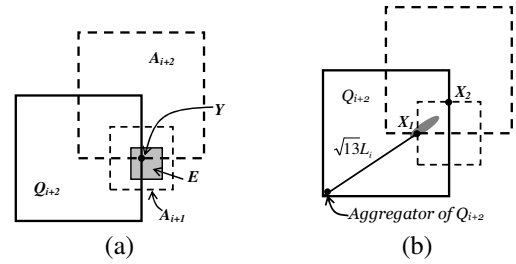


Fig. 7. (a) An event E of size of Q_i that can not be fully covered by a level $i + 2$ cluster. (b) The worst case where the distance between the aggregator and the event is $2(1 + \sqrt{13})$ of the event diameter.

Theorem 1: For an event which can be covered by a square of size of Q_i , packets of the event can be aggregated at or before a level $i + 2$ cluster. Assume that the network can be partitioned into as many levels as possible such that Q_1 clusters are smaller than Q_i for any event, then $\Delta < 2(1 + \sqrt{13})\delta$ where Δ is the distance between the sources and the aggregator where all packets are aggregated, and δ is the event diameter.

Proof: From Lemmas 1, 2, and 3, packets from an event can be aggregated at or before a level $i + 2$ cluster if the area of the event can be covered by a square of size of Q_i .

To prove the bound of Δ , we show that in the worst case, $\Delta < 2(1 + \sqrt{13})\delta$. The worst case happens when packets are aggregated at a level $i + 2$ cluster. For packets to be aggregated at a level $i + 2$ cluster, the event can not be fully covered by any level i or $i + 1$ clusters, else packets will be aggregated before a level $i + 2$ cluster (Lemma 1 and 2). For an event not to be fully covered by any level i or $i + 1$ clusters, the event must cover an intersection point of Q_{i+1} and A_{i+1} clusters, such as X_1 or X_2 in Fig. 7b. Fig. 7b shows the worst case where the aggregator of Q_{i+2} is at the bottom-left corner of Q_{i+2} .

Without loss of generality, we assume that the event covers X_1 , and the level $i + 2$ cluster is a Q cluster, Q_{i+2} , since Q_{i+2} fully covers the event. Assume the length of one side of Q_i is L_i . The distance between the aggregator and X_1 is $\sqrt{13}L_i$. Assume the diameter of the event, δ , is also L_i . Therefore the distance between the farthest source and the aggregator is $(1 + \sqrt{13})L_i$, and $\Delta \leq (1 + \sqrt{13})L_i$. However, for an event to be covered by a square of size of Q_i but not Q_{i-1} , the smallest diameter of the event could be $L_{i-1} + \epsilon = L_i/2 + \epsilon$ where $\epsilon > 0$. Such Q_{i-1} always exists since we assume that Q_1 is smaller than Q_i . Therefore $\Delta < 2(1 + \sqrt{13})\delta$, or $\Delta < 9.22\delta$. ■

IV. DISCUSSION

A. Construction and Maintenance

In AFT, nodes need to know which cluster they belong to at first level so they can aggregate their packets to the corresponding aggregator. Furthermore, aggregators need to know their parent aggregators at higher level. In this section we first describe how clusters are created and then describe how the AFT is constructed.

We assume that nodes have the ability to know their physical location. Sensors can obtain their physical location by configuration at deployment, a GPS device, or localization protocols [32], [33]. We also assume that nodes know the physical location of the sink. Without loss of generality, we assume the sink is located at $(0, 0)$. In AFT we use grid-clustering with exponentially increased cluster size at each level; therefore nodes can determine which clusters they belong to at each level without any communication, given that their physical location and the size of a level one cluster are known.

Once clusters are determined, cluster-head selection protocols can be invoked to elect aggregators. After the aggregators at level i are selected, four level i aggregators can elect one of them as the aggregator at level $i + 1$. Cluster-head selection protocols are not in the scope of this paper. As our approach does not rely on any specific algorithm, many cluster-head selection algorithms for multi-level clusters, such as [17], [34], or hash-based techniques like GHT [35] or [24], can be used.

To balance the energy consumption of nodes, the role of the aggregator has to be rotated among nodes. If hash-based approaches are used, such as the approach used in [24], the changes of aggregators only incur restricted local synchronization overhead. Otherwise the new aggregator needs to inform its parent and child aggregators the update, which requires a constant number of messages with cost proportional to the size its cluster.

B. Irregular Network Topology

In this paper we assume that the network is a square for ease of description. However AFT can still be applied to amorphous network topology. This is attributed to the forwarding rules used in AFT. First, since the clusters are partitioned based on their physical location, nodes can determine their clusters irrespective of the network topology. The only difference is that there might be some clusters that do not have any node

at all. For example, in simulations we have nodes randomly deployed in the network, and the random deployment makes void clusters. However, as described in Property 1, the forwarding rules will only forward packets to clusters that cover at least part of the event, i.e. clusters that have nodes being triggered by the event, those empty clusters do not play a role in packet forwarding and do not have any impact on AFT. The only impact is that hash-based cluster-head selection, such as GHT, may not be adequate because it may hash the cluster-head to a location where this is no node at all.

C. Implementation of AFT

In AFT, we assume that aggregators at each level know which neighboring clusters have packets. For an aggregator at level one, this can be achieved by piggybacking neighboring cluster information in the packet from boundary nodes of the cluster. Boundary nodes can overhear packet transmission activities in neighboring clusters or through explicit announcements to learn if its neighboring cluster has packets. For an aggregator at higher levels, the information can be derived from aggregated packets from lower level aggregators.

Because each Q_i or A_i cluster has four child Q_{i-1} clusters and eight neighboring Q_{i-1} clusters, we use three bytes to represent child Q_{i-1} clusters (4 bits) and neighboring Q_{i-1} clusters (8 bits) from which packets are being received, and child Q_{i-1} clusters (4 bits) and neighboring Q_{i-1} clusters (8 bits) from which packets are expected. Aggregators at higher level can infer which neighboring clusters have packets for aggregation from these three bytes received from lower level aggregators. However the precision of the information will be lower when they are propagated up on the AFT. For example, for an event that spans multiple clusters as shown in Fig. 8, packets in cluster Q_1 will be sent to A_2 (Rule $QR3$), and might be forwarded to bottom Q -cluster ($AR3.3$). If four bits are used to represent if packets are being received, or not received, from the four child clusters, Q_3 can know precisely that it has received packets from $Q_{2,2}$ but not from $Q_{2,1}$. However, when Q_3 forwards the information to Q_4 , Q_4 can only determine that either packets from Q_3 have been received, or not, by using only four bits, one of which is for Q_3 . If Q_4 determines that packets have been received from Q_3 , in some scenarios it may result in packets being forwarded to the sink earlier before they are fully aggregated. On the other hand, if Q_4 determines that packets have not been received from Q_3 , in some other scenarios it may result in continued packet forwarding up on the AFT because the aggregators fail to conclude that all packets have been aggregated.

Fig. 9 shows the percentage of cases in which packets are not fully aggregated among one million randomly generated events when we use three bytes to represent cluster states. The simulation is conducted on a 512×512 level-one clusters network by randomly selecting 2 to 512 contiguous level-one clusters as sources. In over 99% of the cases we can still aggregate all packets with only three bytes of information.

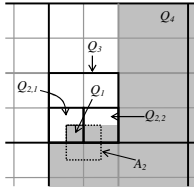


Fig. 8. An event (gray area) spans multiple clusters. Packets from Q_1 will be forwarded A_2 , and Q_3 knows packets are received from $Q_{2,2}$ but not $Q_{2,1}$. This information will be lost when packets are forwarded to Q_4 if only 4 bits are used to represent from which child clusters packets are being received.

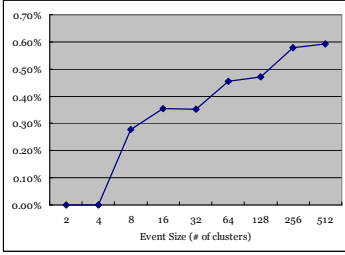


Fig. 9. Percentage of one million cases that does not aggregate all packets

V. PERFORMANCE EVALUATION

In this section we use simulations to evaluate the performance of AFT. Because our goal is to achieve efficient aggregation without incurring heavy control overhead, we compare AFT with two other approaches, Tree-on-DAG (ToD) and Quad-Tree (QT). Since the goal of AFT is to achieve scale-free data aggregation, we evaluate these protocols for large network deployment scenarios.

In all the simulations, unless otherwise mentioned, we randomly deploy 32,768 sensor nodes whose transmission range is $50m$ in a $4096m \times 4096m$ network in order to create a connected network. Therefore each node has roughly 15 neighbors within transmission range. With our most powerful server which has two 64-bit 3GHz CPUs and 4GB memory, the *ns2* simulator could not handle such large deployments. We use a custom-built simulator that does not simulate detailed packet-level behavior, such as collisions and queue-drops. This simulator allows us to trade off the level of detail with the size of the simulation while preserving accurate high level behavior of these protocols. All the simulation results are averaged from 10 different random network topologies.

A. Baseline Simulations

First we create an event of size $\delta \times \delta$ and move the event by $50m$ in X-axis or Y-axis each time, from $(0, 0)$ to $(4096 - \delta, 4096 - \delta)$. This simulates an event of size $\delta \times \delta$ triggering nodes at different locations in the network. The purpose of this simulation is to find out how often a “bad case” (i.e. an event triggering nodes at locations that make static structure approaches fail to aggregate packets near sources) occurs, and how much our approach can improve it. We use a hash-based cluster-head selection algorithms similar to that in [24]. For a cluster at level i , we select a node in a level-one cluster which is closest to the sink and is covered by the level i cluster

as the cluster-head. In case of an aggregator failure, explicit notification has to be broadcast to nodes within the level-one cluster to re-elect a new aggregator. Since the overhead of re-election is the same for the three evaluated approaches, we do not particularly consider node failures in our simulation.

Fig. 10 shows the CDF of ratio of number of transmissions between QT and AFT for three different event sizes. In the simulation we use $64m \times 64m$ ($\sigma = 64m$) as the size of level one clusters. From the figure we can see that when the event size is $100m$, in about 59.5% of the scenarios QT has trouble in aggregating packets near sources (43.2% of the cases have ratio greater than 1.05), and AFT can reduce the number of transmissions by up to four times. When event size increases, the percentage of “bad cases” decreases and the improvement also decreases. This is because when the size of an event is large, there are more nodes transmitting, and most of the transmissions are contributed by transmitting within level-one clusters. When we normalize the number of transmission, the extra transmissions in “bad cases” are amortized.

There are cases where AFT performs worse than QT in the simulation (1% ~ 2% of the cases the ratio is less than 0.95). This is because there are some nodes that do not have neighbors in transmission range in adjacent clusters, or some clusters do not have any node, due to random deployment. Therefore nodes can not learn whether there are sources in neighboring clusters. This may lead to imperfect forwarding in AFT which leads to higher number of transmissions. Though as described in Section IV-C that neighboring information might be lost when they are propagated on the tree, we do not observe this phenomenon in this simulation due to the regular shape of the event. However on average AFT never performs worse than QT (Fig. 14a and Fig. 14b).

Fig. 11 shows results of the same simulation with ToD and AFT. We show the results of $\sigma = 256m$ as the cluster size. We use larger cluster size to favor ToD since AFT performs better in smaller cluster size while ToD performs very bad in small cluster in large event size scenarios. For example, when the cluster size is $\sigma = 64m$ and the event size is $\delta = 500m$, ToD performs very bad compared to AFT. (Fig. 13).

Fig. 11a shows that when the event size is small, in 55.4% of the cases ToD performs better than AFT (36.9% with ratio smaller than 0.95). However when the event size increases, ToD performs worse than AFT in 94.1% of the cases (89.7% with ratio greater than 1.05). In ToD, the cluster size has to be determined in advance based on the size of events to achieve optimal performance. This simulation shows that the performance of ToD highly depends on the size of clusters and the size of events and is not applicable to all scenarios.

To have better insight on why AFT has lower number of transmissions, we conduct the same set of simulations on a grid network with 100×100 nodes in 4096×4096 area (to eliminate the effect of missing neighboring information in random topology network as described above), and collect Δ , the distance between the node at which packets are aggregated and the center of the event, and collect the CDF of Δ/δ , the ratio between the distance and the event size. Fig. 12

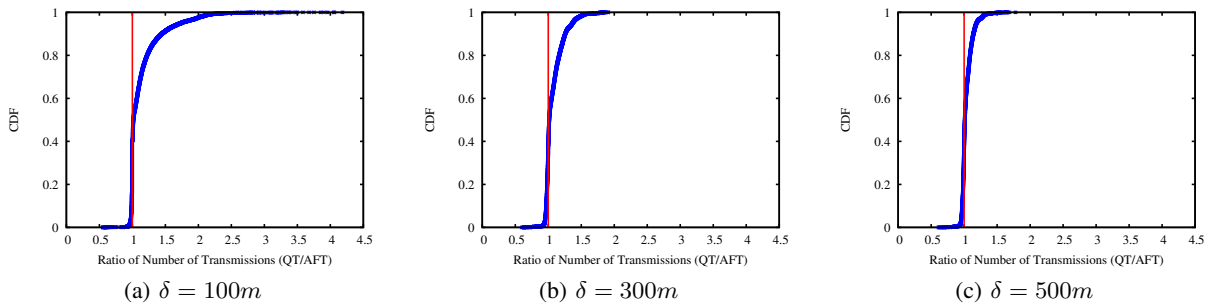


Fig. 10. CDF of ratio of number of transmissions between QT and AFT using $64m \times 64m$ ($\sigma = 64m$) as level-one cluster size.

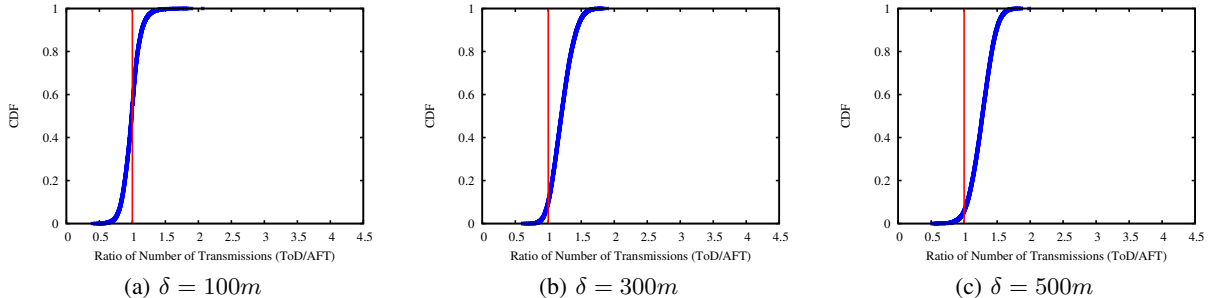


Fig. 11. CDF of ratio of number of transmissions between ToD and AFT using $\sigma = 256m$ as the level-one cluster size.

shows the CDF of Δ/δ in AFT, QT, and ToD. We can see that in AFT, Δ/δ is always bounded by 4, while in QT and ToD they are unbounded (44.8 and 55.8 respectively in this scenario). This shows that AFT can guarantee the aggregation of packets near the sources and therefore effectively reduce the number of transmissions. We do not observe the ratio to reach $2(1 + \sqrt{13}) \simeq 9.22$ in this scenario. Therefore we deliberately create scenarios with $\sigma = 80m$ and event size $\delta = 81m$ to simulate the worst case scenarios. We do observe that the ratio could be as high as 7.293. However the ratio is higher than 4 only in less than 1% of the cases.

B. Cluster Size

Fig. 14a and 14b shows the average normalized number of transmissions for AFT, QT, and ToD in random deployment scenarios with different cluster size when the event size is $\delta = 100m$ and $\delta = 500m$. OPT is an off-line algorithm that computes the shortest path tree for data collection. The shortest path tree contains all source nodes and is rooted at a source node closest to the sink. Data are collected and aggregated from leaves to the root on the tree and are forwarded to the sink thereafter. We use it as the optimal an online protocol can achieve. Normalized number of transmissions is the number of total transmissions in the network divided by the number of packets received at the sink. We can see that when the event size is $100m$, AFT with cluster size $\sigma = 64m$ performs best among all scenarios, 14.11% better than QT with $\sigma = 64m$ (the best among all cluster sizes for QT) and 9.81% better than ToD with $\sigma = 256m$ (the best among all cluster sizes for ToD). When the event size is $500m$, AFT with $\sigma = 64m$ performs similar (2.63% improvement) to QT with $\sigma = 64m$, and is 45.44% better than ToD with $\sigma = 512m$. This shows

that AFT is resilient to the size of the event and can perform better than QT and ToD in any circumstance.

C. Amorphous Event

In the next set of simulations we randomly generate events that cover 4, 16, and 64 randomly selected but contiguous level-one clusters (because we assume an event triggers nodes in contiguous clusters) of size $64m \times 64m$ in random deployment scenarios. This simulation simulates scenarios where events are amorphous. Fig. 15 shows the CDF of Δ/δ for AFT, QT, and ToD. We can see that in scenarios with amorphous event, AFT can bound the ratio to 4 in more than 90% of the cases. For the cases where the ratio exceeds 4 in AFT, most of them are because of the lack of direct connectivity between boundary nodes in adjacent clusters due to the random deployment. In grid network deployment, though with the loss of detailed lower level cluster information as described in Section IV-C, AFT can still bound the ratio within 4 (Fig. 16) over 99% of the cases.

D. Packet Loss

In the last set of simulations we evaluate the impact of packet loss rate on these protocols. All the three evaluated protocols depend on information piggybacked in the packet to determine where to forward packets to. Therefore if packets are lost, the piggybacked information will be lost and it impacts the forwarding decision. Fig. 17 shows the normalized number of transmissions for different packet loss rates in random deployment networks. In the simulation, we use four as the maximum number of retransmissions if packets are lost. From the figure we can see that the trends are similar. The normalized number of transmissions increases as the

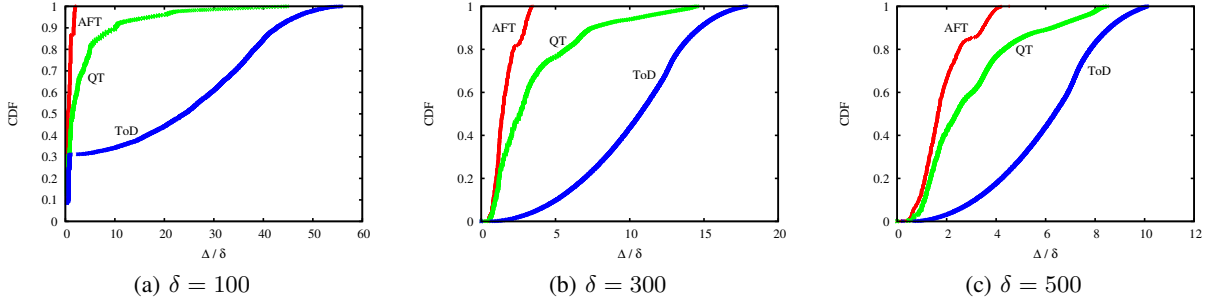


Fig. 12. CDF of ratio between Δ (distance between the aggregation point and the center of the event) and δ (event size) with $\sigma = 64$ in a grid network.

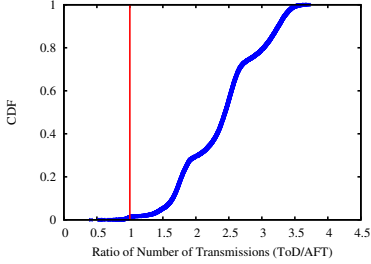


Fig. 13. CDF of ToD/AFT with $\sigma = 64m$

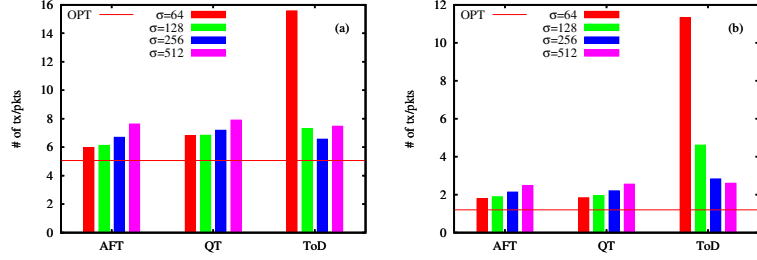


Fig. 14. Average normalized number of transmissions when (a) $\delta = 100m$. (b) $\delta = 500m$.

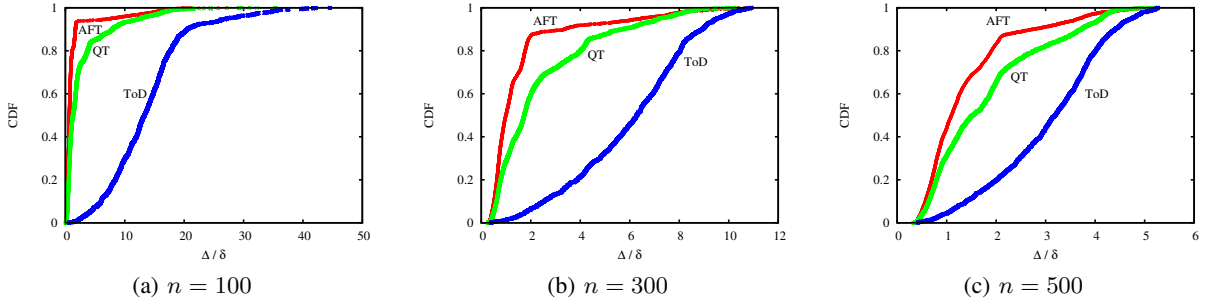


Fig. 15. CDF of ratio between Δ (the distance between the aggregation point and the center of the event) and δ (event size) with $\sigma = 64$ in random topology networks with amorphous event. n is number of randomly selected clusters of size $64m \times 64m$.

packet loss rate increases. It is quite intuitive since packet loss increases the number of transmissions and reduces the number of received packets. However AFT does not deteriorate any faster than QT or ToD.

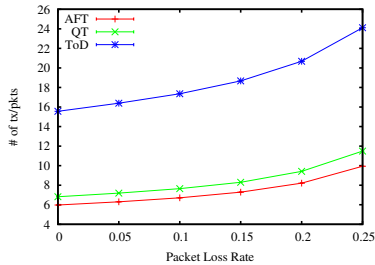


Fig. 17. The normalized number of transmission for different packet loss rates in random deployment network with $\sigma = 64$ and $\delta = 100$.

VI. CONCLUSION

In this paper we propose AFT, Alternative Forward Tree, and its forwarding rules to bound the distance between an

aggregation point and sources within a constant factor, which is $2(1 + \sqrt{13})$, of the event size. AFT is a structure with multi-level overlapping clusters that does not incur high maintenance overhead as dynamic structures. These properties guarantee that AFT is a scalable data aggregation structure irrespective of network size and event size. We evaluate its performance by simulations on 32,768 nodes random topology networks and a 10,000 nodes grid network. We show in simulations that AFT does guarantee the bound when neighboring cluster information is available, while other static structure approaches do not. How to guarantee the bound in random deployment with amorphous event where neighboring information can not be reliably collected is our future work.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants CNS-0546630 (CAREER Award), CNS-0721434, CNS-0721817 and CNS-0403342.

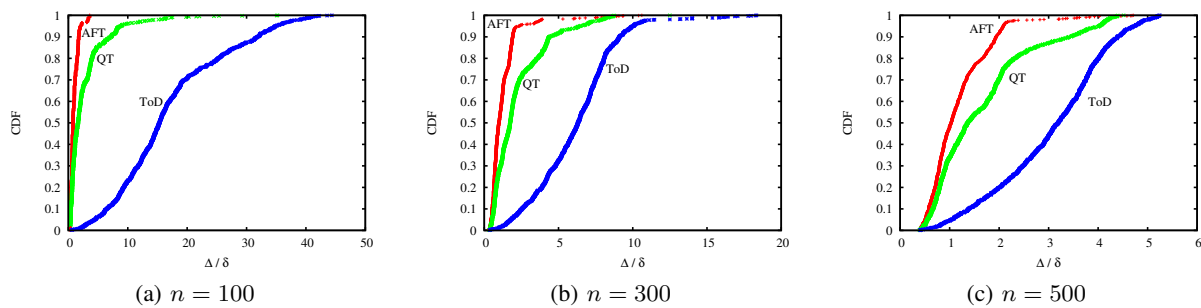


Fig. 16. CDF of ratio between Δ (distance between the aggregation point and the center of the event) and δ (event size) with $\sigma = 64$ in a grid network with amorphous event. n is number of randomly selected clusters of size $64m \times 64m$.

REFERENCES

- [1] D. Peleg and D. Tandler, "Low Stretch Spanning Trees for Planar Graphs," in *Technical Report MCS01-14, Mathematics & Computer Science, Weizmann Institute Of Science*, 2001.
- [2] L. Yu, N. Wang, and X. Meng, "Real-time Forest Fire Detection with Wireless Sensor Networks," *International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, pp. 1214–1217, Sep. 2005.
- [3] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, E. Shih, Y. Zhang, H. Balakrishnan, and S. Madden, "CarTel: A Distributed Mobile Sensor Computing System," pp. 125–138, Nov. 2006.
- [4] Sentel Corporation, "Chemical/Bio Defense and Sensor Networks," <http://www.sentel.com/html/chemicalbio.html>, 2007.
- [5] J. Polastre, "Design and Implementation of Wireless Sensor Networks for Habitat Monitoring," Master's Thesis, Dept. of Electrical Engineering and Computer Sciences, Univ. of California at Berkeley, 2003.
- [6] A. Mainwaring, R. Szewczyk, J. Anderson, and J. Polastre, "Habitat Monitoring on Great Duck Island," <http://www.greatduckisland.net>.
- [7] 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. 8, Jan. 2000, p. 8020.
- [8] —, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," in *IEEE Transactions on Wireless Communications*, vol. 1, Oct. 2002, pp. 660–670.
- [9] 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, Sep. 2002, pp. 924–935.
- [10] 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*, Nov. 2004, pp. 490–497.
- [11] 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, no. 1, Jan. 2004, pp. 61–83.
- [12] M. Ding, X. Cheng, and G. Xue, "Aggregation Tree Construction in Sensor Networks," in *Proceedings of the 58th IEEE Vehicular Technology Conference*, vol. 4, Oct. 2003, pp. 2168–2172.
- [13] H. Luo, J. Luo, and Y. Liu, "Energy Efficient Routing with Adaptive Data Fusion in Sensor Networks," in *Proceedings of the Third ACM/SIGMOBILE Workshop on Foundations of Mobile Computing*, Aug. 2005, pp. 80–88.
- [14] 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, Mar. 2004, pp. 2571–2582.
- [15] 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, no. 3, Apr. 1997, pp. 332–345.
- [16] 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, pp. 499–505.
- [17] 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*, Jun. 2006, pp. 282–304.
- [18] 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.
- [19] "ExScal," <http://www.cast.cse.ohio-state.edu/exscal/>.
- [20] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," in *Proceedings of 22nd International Conference on Distributed Computing Systems*, Jul. 2002, pp. 457–458.
- [21] 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, no. 5, Sep. 2004, pp. 1689–1701.
- [22] K. W. Fan, S. Liu, and P. Sinha, "On the potential of Structure-free Data Aggregation in Sensor Networks," in *INFOCOM 2006*, Apr. 2006.
- [23] K.-W. Fan, S. Liu, and P. Sinha, "Structure-free Data Aggregation in Sensor Networks," in *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, Aug. 2007, pp. 929–942.
- [24] —, "Scalable Data Aggregation for Dynamic Events in Sensor Networks," in *SenSys 2006*, Nov. 2006, pp. 181–194.
- [25] 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*, Dec. 2002, pp. 131–146.
- [26] M. Sharifzadeh and C. Shahabi, "Supporting Spatial Aggregation in Sensor Network Databases," in *Proceedings of ACM Geographic Information System 2004*, 2004, pp. 166–175.
- [27] A. Soheili, V. Kalogeraki, and D. Gunopulos, "Spatial Queries in Sensor Networks," in *Proceedings of ACM Geographic Information System 2005*, 2005, pp. 61–70.
- [28] L. Cai and D. Corneil, "Tree Spanners," in *SIAM Journal of Discrete Mathematics*, vol. 8, no. 3, 1995, pp. 359–387.
- [29] 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, no. 1, Feb. 1995, pp. 78–100.
- [30] A. Arora et al., "ExScal: Elements of an Extreme Scale Wireless Sensor Network," in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2005.
- [31] R. A. Finkel and J. L. Bentley, "Quad Trees: A Data Structure for Retrieval on Composite Keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974.
- [32] 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, Oct. 2000.
- [33] 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.
- [34] E. M. Belding-Royer, "Multi-Level Hierarchies for Scalable Ad Hoc Routing," *Wireless Networks*, vol. 9, no. 5, pp. 461–478, Sep. 2003.
- [35] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," vol. 8, no. 4, Aug. 2003, pp. 427–442.