Distributed Algorithm for En Route Aggregation Decision in Wireless Sensor Networks

Hong Luo, *Member*, *IEEE*, Yonghe Liu, *Member*, *IEEE*, and Sajal K. Das, *Member*, *IEEE*

Abstract—In wireless sensor networks, en route aggregation decision regarding where and when aggregation shall be performed along the routes has been explicitly or implicitly studied extensively. However, existing solutions have omitted one key dimension in the optimization space, namely, the aggregation cost. In this paper, focusing on optimizing over both transmission and aggregation costs, we develop an online algorithm capable of dynamically adjusting the route structure when sensor nodes join or leave the network. Furthermore, by only performing such reconstructions locally and maximally preserving existing routing structure, we show that the online algorithm can be readily implemented in real networks in a distributed manner, requiring only localized information. Analytically and experimentally, we show that the online algorithm promises extremely small performance deviation from the offline version, which has already been shown to outperform other routing schemes with static aggregation decision.

Index Terms—Data aggregation, sensor networks, routing, en route aggregation decision.

1 Introduction

Motivated by the scarce resource limitations in sensor networks [1], [2], extensive research work has been devoted to providing energy-efficient routing algorithms for correlated data gathering [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. By exploring data correlation and employing in-network processing, redundancy among sensed data can be curtailed, and hence, the network load can be reduced [4].

One fundamental question in routing schemes with aggregation is to determine when and where aggregation shall occur along the route in order to explore the data redundancy, which we term *en route aggregation decision*. A natural example for en route aggregation decision is the extensive set of work concerning clustering. There, it has been implicitly assumed that source data is aggregated at the cluster head from where it will be relayed directly to the sink [3], [16]. On the contrary, aggregation-driven routing algorithms [7], [8], [9], [10], [11], [12], [17] have adopted two schemes regarding en route aggregation: full aggregation strategy in which source data will be continuously aggregated on its path to the sink [7], [8], [9], [10], [17], and one-time aggregation where each node selects its best neighboring node for data aggregation and the aggregated data will be sent to the sink without further processing [11], [12]. Naturally, from the aggregation point of view, clustering techniques can be classified into the second category.

While extensive, existing strategies for information routing in sensor networks miss one key dimension in the optimization space for en route aggregation decision, namely the *data aggregation cost* [10]. Indeed, the cost for data aggregation may be negligible for certain applications. For example, sensor networks monitoring field temperature may use simple average, max, or min functions which essentially are of insignificant cost. However, other networks may require complex operations for data fusion.[1] Energy consumption of beamforming algorithm for acoustic signal fusion has been shown to be on the same order of that for data transmission [18]. Encryption and decryption at intermediate nodes will significantly increase aggregation cost in the hop-by-hop secure network since the computational cost is on the scale of nanojoule per bit [19]. Moreover, with increasingly expanding applications, multimedia data including image and video are also being served by wireless sensor networks. In these applications, techniques such as distributed wavelet processing and compressive sensing are intensively studied for in-network compression and processing [20], [21], [22]. As an example, we have studied image fusion when using wavelet-based schemes in [23]. Our results show that the fusion processes incur tens of nanojoule per bit energy consumption, which is on the same order as the communication cost reported in the literature [3], [18]. Similar results have also been reported in [24], where an improved JPEG compressing scheme is studied.

Fusion cost not only affects routing decisions when involving data aggregation but also significantly affects en route aggregation decision regarding when aggregation shall be performed and when it shall be disabled. This can be explained by the example below.

1.1 Motivation

Fig. 1 depicts a sensor network where sensor nodes are deployed on grid and sensed information of the source nodes is to be routed to sink t. Arrow lines form the aggregation tree in which nodes u and v initially aggregate

- *H. Luo is with the School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China. E-mail: luoh@bupt.edu.cn.*
- *Y. Liu and S.K. Das are with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019. E-mail: {yonghe, das}@cse.uta.edu.*

Manuscript received 12 Oct. 2006; revised 29 Oct. 2007; accepted 9 Apr. 2008; published online 19 May 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0276-1006.
Digital Object Identifier no. 10.1109/TMC.2008.82.

1. In this paper, we consider "aggregation" and "fusion" interchangeable, denoting the data reduction process on intermediate sensor nodes.

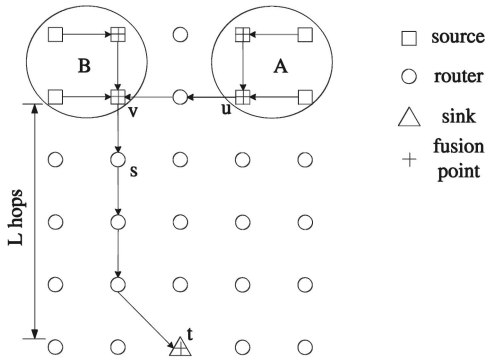


Fig. 1. Illustration of aggregation benefit/disadvantage.

data of areas A and B, respectively. As the sink is far away, u and v further aggregate their data at v and then send one fused data to the sink. Assume each hop has identical unit transmission cost c_0, the fusion cost is linear to the total amount of incoming data, and the unit fusion cost is q_0. Let $w(u)$ and $w(v)$, respectively, denote the amount of data at u and v before the aggregation between them. The amount of resultant aggregated data at v can be expressed as $(w(u) + w(v))(1 - \sigma_{uv})$, where σ_{uv} represents the data reduction ratio owing to aggregation. In this scenario, if v performs data fusion, the total energy consumption of the route from v to t, assuming there are L hops in between, is $Lc_0(w(u) + w(v))(1 - \sigma_{uv}) + q_0(w(u) + w(v))$. On the contrary, if v does not perform data fusion, the total energy consumption of the same route is simply the total relaying cost, $Lc_0(w(u) + w(v))$. To minimize the *total energy consumption of the network*, v should not perform data fusion as long as $\sigma_{uv} < \frac{q_0}{Lc_0}$.

From this example, we can see that neither the full aggregation nor the one-time aggregation can be the best solution, the decision at an individual node has to be based on data reduction ratio due to aggregation, its related cost, and its effect on the communication costs at the succeeding nodes.

1.2 Our Contribution

In our existing work [23], we proposed *Adaptive Fusion Steiner Tree* (AFST), a routing scheme that can dynamically assign fusion decisions to routing nodes during the route construction process by evaluating whether fusion is beneficial to the network based on fusion/transmission costs and network/data structures. But this offline solution cannot be readily implemented in a real network without incurring significant overhead, as it requires global information of the network and centralized derivation.

Our key task in this paper is then to design an online algorithm that can be implemented by individual sensor nodes in a distributed fashion relying on local information only. New sensor nodes can be deployed for enhanced coverage, and existing sensor nodes can become nonfunctional over time. Furthermore, nodes can be periodically scheduled into sleep mode in turn for prolonged network lifetime. Our proposed algorithm, termed *online AFST*, can effectively perform nodes adding and deleting only by modifying local routing structure without triggering extensive rerouting in the network. Such property can significantly reduce the overhead due to repetitive communication and calculation. Furthermore, due to this nature, the algorithm can be readily implemented in real environments based only on localized information. Through extensive simulations, we show that online AFST closely follows its offline version—within 15 percent performance deviation under a wide range of parameters.

The remainder of this paper is organized as follows: In Section 2, we describe the system model and formulate the routing problem. Section 3 describes the offline version of the randomized approximation algorithm AFST. Sections 4 and 5 present in detail the design and analysis of the proposed online AFST algorithm. Section 6 improves the online algorithm to a distributed algorithm. In Section 7, we experimentally study the performance of the online algorithm. Related work is discussed in Section 8, and Section 9 concludes this paper.

2 System Model and Problem Formulation

2.1 Network Model

We model a sensor network as a graph $G = (V, E)$, where V denotes the node set and E the edge set representing the communication links between node-pairs. We assume a set $S \subset V$ of k nodes are data sources of interests and the sensed data needs to be gathered at a special sink node $t \in V$.

For a node $v \in S$, we define node weight $w(v)$, denoting the amount of information outgoing from v. Evidently, various data fusion algorithms will result in different weights, and therefore, a node's weight is *dynamic* in the process of data fusion. In order to avoid confusion, we use $\widetilde{w}(\cdot)$ to denote the temporary weight of a node *before data fusion* and use $w(\cdot)$ to denote the weight of a node *after data fusion*.

An edge $e \in E$ is denoted by $e = (u, v)$, where u is the start node and v is the end node. Then, the data amount transmitting on edge e, denoted by $w(e)$, is equivalent to the weight of its start node, i.e., $w(e) = w(u)$. Two metrics, $t(e)$ and $f(e)$, are associated with each edge, describing the transmission cost and fusion cost on the edge, respectively.

2.2 Transmission and Fusion Costs

Transmission cost $t(e)$ denotes the cost for transmitting $w(e)$ amount of data from u to v. Let $c(e)$ denote the cost (e.g., power) when transmitting unit data on edge e, the transmission cost $t(e)$ of edge e is then given by $t(e) = w(e)c(e)$.

Fusion cost $f(e)$ denotes energy consumption for fusion process at the *end* node v. $f(e)$ depends on the amount of data to be fused as well as the algorithms utilized. In this paper, we use $q(e)$ to abstract the unit fusion cost. Then, the cost for fusing the data of nodes u and v at node v is given by $f(e) = q(e)(w(u) + \widetilde{w}(v))$.

2.3 Correlation and Data Aggregation

Key to a sensor data routing protocol is the data aggregation ratio. Regardless of the application scenarios, we use an abstract parameter ρ, the correlation coefficient, to denote the data redundancy between nodes. If node v is responsible for fusing node u's data (denoted by $w(u)$) with its own, we have $w(v) = \widetilde{w}(v) + w(u)(1 - \rho_{uv})$, where $\widetilde{w}(v)$ and $w(v)$ denote the data amount of node v before and after fusion.

Due to aggregation cost, node v may choose to simply relay the incoming data of node u instead of performing data aggregation in order to realize maximum energy saving. In this case, the new weight of node v is simply $w(v) = w(u) + \widetilde{w}(v)$ and there is no fusion cost on edge (u, v) obviously.

2.4 Problem Formulation

Given the source node set S and sink t, our objective is to design a routing algorithm that minimizes the energy consumption when delivering data from all source nodes in S to sink t. Not only do we need to design routing paths back hauling sensed information driven by information aggregation, but also we have to optimize over the decisions as to whether aggregation shall occur or not on a particular node.

Mathematically, a feasible routing scheme is a connected subgraph $G' = (V', E')$, where $G' \subset G$ contains all sources $(S \subset V')$ and the sink $(t \in V')$. Depending on whether fusion is performed or not, the edge set E' can be divided into two subsets E'_f and E'_n, where E'_f includes all fusion edges and E'_n represents the unfusion edge set. Our goal is to find a feasible subgraph G^* such that

$$G^* = \text{argmin}_{G'} \sum_{e \in E'_f} (f(e) + t(e)) + \sum_{e \in E'_n} t(e). \quad (1)$$

In [23], we have described an offline solution, termed AFST, for the above problem. However, this solution requires deriving of the routing structure to be performed at the sink based on the global information of the network. Our approach toward a distributed solution can be outlined as follows: Based on the offline algorithm, we first propose an online algorithm that can 1) efficiently incorporate a newly coming node into the network by connecting it to the existing routing tree and 2) efficiently disconnect a node from the network without affecting other part of the routing tree. Our design utilizes only local information in the existing routing structure to perform the above tasks while maintaining the energy efficiency achieved by the offline algorithm. Given this property, the online algorithm can then be readily implemented in a distributed fashion as local information about the routing structure can be fully obtained by communicating with neighbor nodes.

Before detailing the online algorithm design, we will first give a brief overview of the offline version and quantify its performance.

3 Brief Overview of Offline AFST

Offline AFST is a hierarchical matching algorithm and runs in "stages." In each stage, we first pair up source nodes (or source with the sink) based on the defined metrics and then randomly select a fusion node from the node-pair. The fusion node evaluates the benefit of fusion for the matching. If fusion is deemed worthy, the weight of the nonfusion node will be transferred to the fusion node, paying appropriate transmission and fusion costs on that edge. Subsequently, the nonfusion node will be eliminated and the fusion node with aggregated weight will become an element of the new source set. If fusion is deemed unworthy, both nodes will be linked to the sink via their respective shortest paths, paying appropriate transmission cost, and both nodes will be eliminated. This process will then be repeated on the new source set composed of chosen fusion nodes until only the sink remains. The algorithm is detailed below.

OFFLINE AFST ALGORITHM:

1) Initialize the stage loop index $i = 0$. Define source set $S_0 = S \cup \{t\}$, and $E^* = \emptyset$. Let w_{v_0} for any $v \in S$ equals to its original weight, and let $w_{t_0} = 0$.
2) In each stage, there are three steps: matching, decision, and fusion.
   a) In the matching step, for every pair of nodes $(u, v) \in S_i$, find the minimum cost path (u, v) in G according to the new metric

   $$M(e) = q(e)(w_{u_i} + w_{v_i}) + \alpha(w_{u_i}, w_{v_i})c(e),$$

   where $\alpha(w_{u_i}, w_{v_i})$ is the expected amount of data transmitted between node u and v. Intuitively, $M(e)$ denotes the expected transmission and fusion costs on edge e in this stage. Given the minimum cost paths, minimum-cost perfect matching is then performed between nodes in S_i according to the metric $M(e)$.
   b) In the decision step, for each matched pair (u, v), the fusion decision is made by calculating the fusion benefit according to

   $$\Delta_{v_i} = w_{u_i}\rho_{u_i v_i} s(v_i, t) - q_{u_i v_i}(w_{u_i} + w_{v_i}), \quad (2)$$

   where the first part of (2) denotes the energy saving on the shortest path from v_i to the sink t, and the second part of (2) is the corresponding fusion cost. We call (u, v) a nonfusion pair if there is no fusion benefit regardless which node is selected as the fusion point, i.e., $\Delta_{u_i} < 0$ and $\Delta_{v_i} < 0$. Otherwise, we call it fusion pair.
   c) In the fusion step, for each fusion pair (u, v), add those edges that are on the minimum-cost perfect matching path to set E^*_f. Randomly choose one node to be the fusion node. Remove the nonfusion node from S_i and the weight of nonfusion node will be transferred to its corresponding fusion node. For each nonfusion pair (u, v), add those edges that are on the shortest paths of (u, t) and (v, t) to set E^*_n. Remove both u and v from S_i.
3) All remaining fusion nodes induce S_{i+1}.
4) If S_{i+1} has no more nodes left besides the sink, the algorithm stops. Otherwise, increment i and return to step 2.

It can be shown that the resultant routing tree of AFST can be divided into two parts. The lower part consists of subtrees inside which aggregation is always performed, and the upper part is always employing shortest paths from all roots of the full fusion subtrees to sink t.

3.1 Performance of AFST

The main idea of AFST is to perform data aggregation when there exists sufficient data redundancy and the fusion benefit can justify corresponding cost. Therefore, when the correlation coefficient ρ is small or the unit fusion cost q is

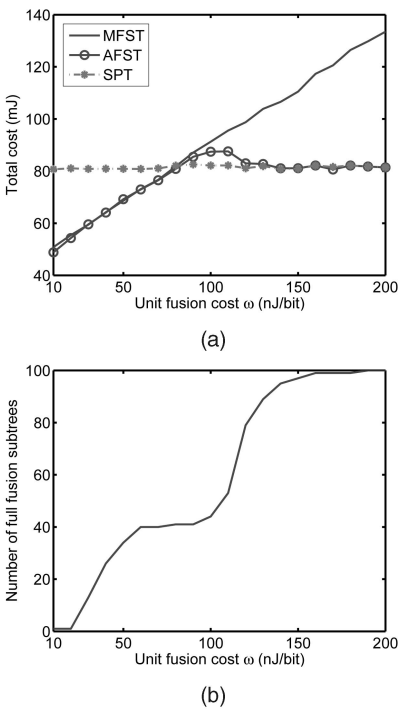


Fig. 2. Impact of unit fusion cost to energy consumption. (a) Total cost. (b) Number of subtrees.

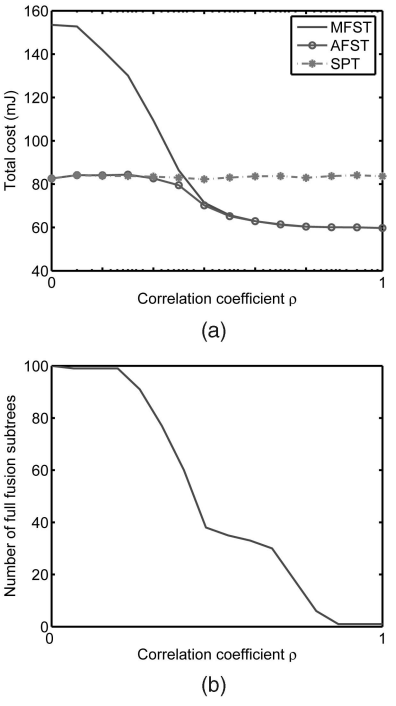


Fig. 3. Impact of average correlation coefficient to energy consumption. (a) Total cost. (b) Number of fusion subtrees.

large, AFST leans toward simple SPT without data fusion; on the contrary, if the correlation coefficient ρ is large and the unit fusion cost q is small, the result of AFST leans toward a full data aggregation tree.

Fig. 2 compares the performance of three algorithms, namely SPT, MFST, and AFST, with varying unit fusion cost. MFST is a full aggregation tree proposed in [10], which jointly optimizes over both the fusion and transmission costs and achieves $\frac{5}{4}\log(k+1)$ approximation ratio to the optimal solution. In the simulation, we have 100 nodes uniformly deployed in a 50 $\times$ 50 m square, each with maximal communication range of 20 m. The data correlation coefficient among nodes is set to be inversely proportional to their distance.

As we can see in Fig. 2a, the total cost of MFST increases unboundedly along with the increase of unit fusion cost since it cannot stop doing fusion even at high fusion cost. At the same time, the cost of SPT is constant since no data fusion occurs and, hence, no influence of unit fusion cost. However, this also accounts for its high energy cost when the unit fusion cost is low. On the contrary, AFST follows MFST first when unit fusion cost is low by jointly considering the effects of both data redundancy and fusion cost. As fusion cost increases, it leans toward SPT, the optimal solution when fusion cost is high. The figure can be best explained when we jointly examine it with Fig. 2b, which depicts the number of full fusion subtrees in AFST. When the unit fusion cost is very small, there are only two full fusion subtrees. This denotes that data fusion is performed almost at all nodes, benefiting from the low fusion cost. When the unit fusion cost increases, AFST reduces the number of full fusion subtrees due to reduced fusion benefit in order to balance the fusion cost and transmission cost. When the unit fusion cost is too large, AFST can achieve the same constant cost as SPT by completely stopping data fusion.

Fig. 3a shows the performance of the three algorithms with different data correlation under moderate unit fusion cost. Still there are 100 nodes with maximal communication range of 20 m. The unit fusion cost is set to 50 nJ/bit.

When the data correlation in the network is very weak $(\rho \to 0)$, AFST follows SPT, the optimal solution, by avoiding any data aggregation. When the data correlation in the network increases, AFST dynamically adjusts its decisions accordingly by performing data fusion partially in order to benefit from data aggregation and resultant data reduction. When the data in the network is highly correlated $(\rho \to 1)$, AFST makes perfect matching and performs data fusion in each stage like MFST to pursue the most energy saving. This dynamic adjustment process can be explicitly illustrated in Fig. 3b which depicts the number of full fusion subtrees changing from 100 (no aggregation in network) to 1 (full aggregation) as a result of increase value of ρ. Due to this dynamic adjustment, the cost of AFST is extremely smooth in the whole range of the correlation coefficient and steadily outperforms others.

As fusion cost and data correlation may vary widely from network to network, from application to application. AFST is the only algorithm that can adapt best to a wide range of scenarios and, hence, be applicable to a variety of applications.

4 Design of Online AFST Algorithm

In this section, we present an online version of AFST algorithm that can dynamically adjust the routing structure when new nodes arrive and existing nodes leave the network. When a source node joins the network, it must

find the best point in the existing tree to establish a connection. This can be either a fusion point to combine its data or a relay path to the sink due to the lack of fusion benefit. When a source node leaves the network, its direct children will have to reconnect to the existing tree accordingly. Nonetheless, our goal in designing these algorithms for joining and leaving is to confine the structure change to be local so that the impact to the remaining routing tree will be minimal, and at the same time, optimize the energy consumption of the new routing tree.

Before getting into details about our reasoning and philosophy in the design, we first present below the algorithms for source arrival and departure separately.

4.1 Algorithm for New Source Arrival

In offline AFST, in a certain stage, a sensor node will eventually become a nonfusion node and its data will be aggregated at the corresponding fusion node. We denote this stage as a node's "level." Evidently, for each source node, it can fuse data from nodes with lower levels, and its data will be fused at a node with higher level. Our key idea in accepting new arriving node is to randomly assign a level to it and connect it to the existing tree. In doing so, we also simultaneously best preserve the existing level structure and minimize the additional cost introduced by the newcomer.

Formally, given the existing data routing tree T for source set S and the network graph $G = (V, E)$, our objective is to construct a new routing structure T' for a new source set $S' = S \bigcup \{u\}$, where u is the new arriving source node. Our designed algorithm is detailed below.

ONLINE ALGORITHM FOR SOURCE ARRIVAL:

1) Set $T' = T$.
2) Assign a level i, $1 \leq i \leq \log(k+1)$, to the arrival node u with probability
$$1/2^i. \tag{3}$$
3) From every source node in T with level higher than i and being "unmatched" at level i, select the best fusion point v for node u so that the total additional cost on the entire network is minimized. The total additional cost on path from u to t via v, denoted by $\triangle C(P_{ut})|_v$, is given by
$$\triangle C(P_{ut})|_v = q_{uv}(w(u) + \widetilde{w}(v)) + c_{uv}w(u) + w(u)(1-\rho_{uv})g(v). \tag{4}$$
Here, the first two parts of (4) are the fusion cost and transmission cost on establishing the edge (u, v). $g(v)$ denotes the total cost of delivering unit data from v to sink t in the existing aggregation tree, and hence, the last part of (4) is the additional cost on path (v, t) if u employs v as its connection point to the existing tree.
4) If $\triangle C(P_{ut})|_v$ is less than the transmission cost on the shortest path from u to sink t,
   - Chosen v as the best fusion point for u, and add the edge (u, v) to T' as a new fusion edge.
   - Set $g(u) = g(v) + c_{uv} + q_{uv}$, and increase the weight of all nodes on path (v, t) by $w(u)(1-\rho_{uv})$, i.e., for each node s on path (v, t), set $w(s) = \widetilde{w}(s) + w(u)(1-\rho_{uv})$.
   - Set both u and v as "matched" on level i, and set u as "unmatched" on each level $j < i$.
   - Return T'.
5) Else, add the edges on the shortest path from u to t (SP_{ut}) to T' as unfusion edges. Set $g(u) = \sum_{e \in SP_{ut}} c_e$. Set u as "unmatched" on each level $j \leq i$. Return T'.

4.2 Algorithm for Existing Source Departure

For a node leaving the existing route tree, our key idea in reconstructing the routing structure is to connect all nodes fused at the departing node to new points in the existing tree with minimal additional cost while minimizing the influence to other exiting nodes. Therefore, only the direct children of the leaving source will change their paths, and all other routes will remain untouched.

Formally, given the existing data routing tree T for source set S and the network graph $G = (V, E)$, our objective is to construct a new routing structure T' for a new source set $S' = S - \{u\}$, where u is the departing source node. Our designed algorithm is detailed below.

ONLINE ALGORITHM FOR SOURCE DEPARTURE:

1) Remove the edge from u to its parent in T to construct T'.
2) If v is the fusion point for departing node u, Set v as "unmatched" at u's level. Reduce the weight of all nodes on existing path (v, t) by $w(u)(1-\rho_{uv})$.
3) If u itself is the fusion point of other sources, for each child r, perform source arrival algorithm with existing level assignment to add r and its children to T'. Reduce $g(\cdot)$ of all source nodes in the subtree of node r as same amount as node r.
4) Return T'.

5 ALGORITHM ANALYSIS

In this section, we first detail the theoretical reasons of the above design and in particular those of (3) and (4), and then we analyze the complexity and performance of the online algorithm.

5.1 Random Level Assignment

In the offline AFST, if every node matches its pairing node for fusion in every stage, there will be $\log(k+1)$ levels and $\frac{k+1}{2^i}$ nodes in level i, $1 \leq i \leq \log(k+1)$. In the online algorithm, in order to maintain this level structure for balancing fusion cost among source nodes, we randomly assign a level i to each new arrival source with probability $\frac{1}{2^i}$. A new source with level i can only match with an existing source node whose level is higher than i and has no fusion partner (unmatched) at level i. This way, the same level structure is preserved as in the offline algorithm.

5.2 Calculation of Minimum Additional Cost Access

As AFST contains a lower part with full aggregation and an upper part with no aggregation, when a new node u arrives, it shall either join the full aggregation lower part or directly send its data to the sink via shortest path and, hence,

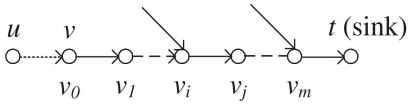


Fig. 4. Example of path from candidate to sink.

become a member of the upper part, all dependent on whether fusion is beneficial. To make this decision, we include all nodes with levels higher than u's level i and unmatched at level i as the potential fusion candidate set F. Obviously, sink t also belongs to F. For each potential fusion node $v \in F$, if v is a source node, we calculate the total network additional cost when node u selects v as its access point to the network, which is composed of the data aggregation cost at v, the transmission cost from u to v, and transmission/fusion costs for the increased data along the path from v to the sink t. On the contrary, if v happens to be the sink, the total network additional cost will only include the transmission cost from u to t. To minimize the total energy consumption, we will select the candidate that will incur the lowest additional cost. Therefore, the key challenge is how to determine the additional cost for each candidate node.

Assume that the path from candidate v to sink t includes m intermediate nodes: $v(v_0)$, $v_1, v_2, \ldots, v_m$, t, as shown in Fig. 4. If u connects to v for data fusion and use the same path for deliver additional data, we can infer that the cost for transmitting u's data to node v and performing corresponding fusion, denoted by $C(P_{uv})$, is

$$C(P_{uv}) = q_{uv}(w(u) + \tilde{w}(v)) + c_{uv}w(u), \tag{5}$$

and the cost on transmitting and fusing u's additional data (weight) on the path from v to sink t, denoted by $\triangle C(P_{vt})$ can be written as

$$\begin{aligned}\triangle C(P_{vt}) = {} & g_{v_0v_1}\triangle w(u)|_{v_0} + g_{v_1v_2}\triangle w(u)|_{v_1} \\ & + \cdots + g_{v_mt}\triangle w(u)|_{v_m},\end{aligned} \tag{6}$$

where $\triangle w(u)|_{v_i}$ denotes the additional weight on node v_i due to the addition of node u to the path, and $g_{v_iv_{i+1}}$ includes the unit transmission cost $c(e)$ and potential unit fusion cost $q(e)$ on edge $e = (v_i, v_{i+1})$.

For simplification, we use $\triangle w(u)|_{v_0} = w(u)(1 - \rho_{uv})$ to approximate $\triangle w(u)|_{v_i}$ for all i and consequently (6) can be simplified to

$$\triangle C(P_{vt}) = w(u)(1 - \rho_{uv})g(v), \tag{7}$$

where $g(v) = \sum_{e \in P_{vt}} g_e$ denotes the unit cost for path (v, t) in the existing tree, and $w(u)(1 - \rho_{uv})$ approximates the additional weight on path (v, t). Combining (5) and (7), we obtain (4) used in the online algorithm described in Section 4.

Indeed, the above approximation will introduce deviation from the original value. However, this deviation is rather small and acceptable. Notice that additional weights introduced by node u on path (v, t), $\{\triangle w(u)|_{v_i}\}$, monotonously decrease due to consecutive data aggregation operations along the path. Moreover, if v is the best fusion point for node u among all candidates, v must possess higher fusion benefit with node u than other candidates. Therefore, the decreasing rate of $\{\triangle w(u)|_{v_i}\}$ shall also be reducing which renders the deviation to be small. Furthermore, our experiments also show that even by employing this approximation, the online algorithm still closely follow the offline version.

5.3 Algorithm Complexity

To understand the complexity of the online algorithm, we first investigate the complexity of offline AFST algorithm described in Section 3. In each stage, it first takes $O(|E|)$ time to compute the metric $M(e)$ on every edge. Using Dijkstra's Algorithm, we can compute the shortest path from one node to others in $O((|V| + |E|)\log|V|)$. Therefore, we can compute the shortest path under metric $M(e)$ for all node-pairs in stage i in $O(|S_i|(|V| + |E|)\log|V|)$, where $S_i = \frac{k+1}{2^i}$ is the number of nodes in stage i. At the same time, the perfect matching operation can be performed in $O(|S_i|^2 \log|S_i|)$ time [25]. It then follows that stage i takes at most $O(|S_i|(|V| + |E|)\log|V| + |S_i|^2 \log|S_i|)$ time. Since the size of S_i is reduced at least half in each stage, by summarizing all stages, we conclude the total running time to be $O(|S|(|V| + |E|)\log|V| + |S|^2 \log|S|)$. For the special case when all nodes are source nodes, i.e., when $|S| = |V| = n$, the total running time is $O(n^2 \log n + |E|n \log n)$.

Now, we turn to the complexity of online AFST algorithm. Using Dijkstra's Algorithm, we can compute the shortest path from the new node to all other nodes in $O((|S_i| + |E_i|)\log|S_i|)$, where S_i and E_i are the number of existing nodes and edges, respectively. The selection process takes $O(S_i)$ time. The data updating step can be performed in $O(\log S_i)$ time as the length of any path is no more than $\log S_i$. It follows that each newcomer takes at most $O((|S_i| + |E_i|)\log|S_i| + \log S_i + S_i)$ time. As a result, the online algorithm can be finished in polynomial time to derive the new data gathering tree.

The total running time for adding all n nodes is $\sum_{|S_i|=1}^{n}((|S_i|+|E|)\log|S_i| + \log S_i + S_i) = O(n(|E| + n)\log n)$, which is actually the same for constructing the data gathering tree via the offline algorithm. However, the online algorithm provides the additional flexibility of incorporating node into the network as they arrive.

5.4 Competitive Ratio

Due to the complex nature of the problem and algorithm, the competitive ratio is hard to obtain theoretically. Instead, we will show that the resultant routing tree has bounded cost in the worst case.

In AFST, the shortest path is the selection if there is no fusion benefit. In offline AFST, this decision is made by taking all source nodes into account. However, in online AFST, this decision is made depending only on the current tree structure and the incoming node. In addition, the original tree structure is preserved in the online algorithm to avoid additional maintain overhead specially for distributed computation. As a result, adversarial arriving order will lead the online AFST to select the shortest paths even though there exists fusion benefit if all source nodes are known in advance. It is then one worst case when the offline result is one full fusion tree but the online result is purely SPT.

This worst case performance of the online algorithm can be illustrated by the example depicted in Fig. 5. In Fig. 5a, there are four sensor nodes, p, s, u, v, located linearly and

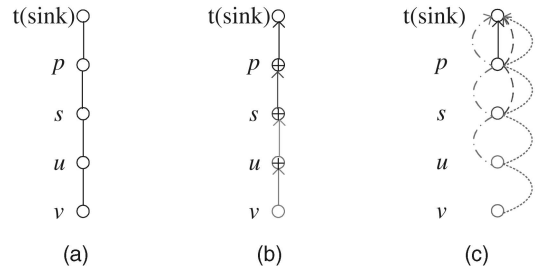


Fig. 5. Example of adversarial arrival. (a) Sensor network. (b) Offline result. (c) Online result.

equidistantly with the sink at the end of the line. Assume that a node can only communicate with its neighbors and the unit transmission cost is c_0 for such communications. We also assume that the unit fusion cost is a constant q_0 for simplification.

If the correlation coefficient ρ between neighbors is large enough, the offline AFST will generate a full aggregation tree as shown in Fig. 5b. According to the offline AFST algorithm, the condition of matching node v with node u is that performing data aggregation at u will lead to less energy consumption than transmitting the raw data from both nodes separately, i.e.,

$$c_0 w(v) + q_0(w(u) + w(v)) + 3c_0 w(v)(1 - \rho_{uv}) < 4c_0 w(v). \quad (8)$$

Assume that the original data amount of u and v is the same, i.e., $w(u) = w(v)$. This condition can be rewritten as $\rho_{uv} > \frac{2q_0}{3c_0}$. For illustration, we set $\rho_{uv} = \frac{3}{4}$ and $q_0 = c_0$ in order to satisfy inequity (8).

In the online algorithm, four sensors can join the network in the adversarial order of v, u, s, p. In this case, when the farthest node is activated first, all the other nodes only act as relays. Specifically, when node v is activated, it reports its data to the sink through the shortest path, the cost is $4c_0 w(v)$. When node u is activated, since the fusion condition for fusing u's data at v in online AFST, denoted by

$$c_0 w(u) + q_0(w(u) + w(v)) + 4c_0 w(u)(1 - \rho_{uv}) < 3c_0 w(u), \quad (9)$$

cannot be satisfied for $\rho_{uv} = \frac{3}{4}$ and $q_0 = c_0$, node u will employ direct relaying to report its data to the sink. The left side of inequity (9) summarizes the cost if fusion is performed while the right side denotes the cost if direct relay is adopted.

For the same reason, s and p will also employ direct relay without fusion to report their data to the sink when they are activated. As a result, the online algorithm will construct an all-relay structure as shown in Fig. 5c as the data gathering route. This is indeed a Shortest Path Tree.

This simple example shows that the worst case scenario can only happen when each source node arrives before all its potential matching pair nodes, which is rather rare in real deployment. Moreover, even in this case, Fig. 3 in Section 3 shows the performance penalty is not large. There, the result of offline AFST is a full aggregation tree when $\rho \to 1$. Therefore, the ratio between SPT and AFST at $\rho = 1$ demonstrates the upper bound of the online competitive ratio in the extreme case.

In fact, we remark that the extremely worst case rarely happens since the tree construction of both algorithms is influenced by the correlation coefficient and the unit fusion cost. First, if the correlation coefficient is large or the unit fusion cost is very small, the offline AFST indeed tends to generate a full aggregation tree. Similarly, in the online AFST, a newcomer can aggregate its data with other existing nodes even if its perfect pair-node is not available, as the fusion benefit always exists. Therefore, the online algorithm will not result in a pure SPT. Second, if the correlation coefficient is small or the unit fusion cost is large, the online algorithm may tend to lead newly arriving nodes toward shortest paths. However, during the meantime, the offline algorithm will also employ shortest path as its routing strategy as shown in Fig. 2b, as the small fusion benefit cannot justify the fusion process itself. Third, if the correlation coefficient and unit fusion cost are moderate, the resultant trees from both algorithms will consist of two parts: the lower part of full fusion and the upper part with only direct relay. Lastly, in a real application scenario, the order of arrival and departure is random which lead the probability of the worst case scenario to be minimal. Therefore, the overall expected deviation shall be much smaller than the worst case. Our simulation result, as described in Section 7, will further solidify our arguments.

6 Distributed Computation

In this section, we describe how the proposed online algorithm can be implemented in a distributed manner in a real network. In particular, we detail how an incoming node can collaborate with its neighbors to perform the joining procedure to the routing tree and how neighboring nodes can effectively reconstruct the routing tree when a node leaves. Notice that in the above design, we have intentionally limited route reconstructions to be local only. At the same time, in the physical network environment, a new incoming node can only connect to neighbors within its communication range and likely to perform fusion with its neighbors of proximity due to higher correlation. These are the key enablers of distributed implementation of the online algorithm: when a new node arrives, it only needs to build up its candidate set from its source neighbors instead of all higher level nodes in the field.

In the distributed algorithm, each source node needs to maintain information about the local routing structure, which includes the information about its parent (fusion point), children (whose data is fused at this point), and itself. Parent information includes the parent's node ID. Children information consists of a list about all direct children's node ID and the additional weight each child brings in. Self information includes the level i, matched level list, current weight $w(\cdot)$, unit cost to sink $g(\cdot)$, and the neighbor list.

6.1 Initialization

When the sensor nodes are initially deployed, the sink executes the offline AFST to produce the initial data gathering tree and propagates this information to the network through the resultant tree. This process needs to be performed only once during the initialization phase.

6.2 Node Arrival

When a new source u arrives, it shall first determine the unit transmission cost $c(e)$ to its neighbors by using, for example, Received Signal Strength Indicator (RSSI) [26]. By querying neighbor's local routing data, u can also obtain a neighbor v's level, matched level list, unit cost to sink $g(v)$ and current weight $w(v)$, it can also derive the shortest path to the sink based on its neighbors information. Based on the location information, u can estimate the correlation coefficients ρ_{uv} between itself and a neighbor v. We assume that the unit fusion cost $q(e)$ is identical and the procedure for executing the online AFST locally is detailed below.

Node u randomly generates a level i for itself and determines fusion point candidates from its neighbors. Based on this candidate set, node u will execute the online algorithm to determine if fusion shall be performed with one candidate or direct relaying shall be adopted. If fusion is beneficial and node v is selected as the fusion point, u sends *source join* message to v with u's level and additional weight. v updates correspondingly its *matched level list, current weight, and children list* after receiving this message, and then forwards this message along its path to the sink. Each source node on the path will updates its *current weight* and the message sender's *additional weight* in its children list by adding u's additional weight.

In case u is a nonsource, when it arrives, it will simply find its shortest path to the sink and calculates its *unit cost to sink*. Consequently, other nodes can use u to adjust their fusion routes for more energy saving.

6.3 Node Departure

When a fusion point v finds that its child node u has left the network, it first updates its *children list* and *matched level list*. Subsequently, it shall reduce its *current weight* by deducting u's additional weight. At the same time, v shall send *child departure* message with u's additional weight along the path from v to sink t. Each node on this path shall update its *current weight* and the message sender's *additional weight* in its children list by deducting u's additional weight.

When a node s finds that its fusion point (parent) has left the network, it executes the *source arrival* procedure as detailed above to locate a new fusion point v. It then shall update its parent information and send *source join* message to its new parent, v, to update the data of all nodes on the path from v to t. Subsequently, node s sends a *change parent* message to all of its children with a unit cost amendment parameter that equals to $g_{new}(s) - g_{old}(s)$. Each child shall update its *unit cost to sink* by adding the unit cost amendment and forwards this message to all of its children till to the leaf of the routing branch.

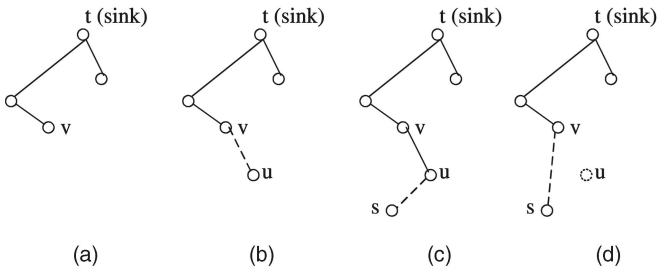


Fig. 6. Example of source arrival/departure.

When a nonsource node leaves, there is no effect to the network if it has no source child. If this is not the case, all the children of data source can observe this departure event and choose their new parents following the approaches described above.

Fig. 6 illustrates the process of source arrival and departure. Fig. 6a depicts the existing routing structure. Fig. 6b illustrates that incoming node u, after running the above described source arrival procedure, selects v as its fusion point and establish connection to it. Fig. 6c illustrates that another node s joins the data gathering tree. And Fig. 6d depicts the case that source u departs and node s reconnects to the tree via node v. Table 1 illustrates the changes of the two main local tree data variables, namely the *current weight* and child's *additional weight*.

6.4 Route Adjustment

When a node newly joins the network, it may serve as fusing node or relay node for existing sensor nodes. In order to explore the potential benefit introduced by this node, local routing adjustment can be performed. The basic idea here is to let the neighboring nodes to choose freely whether to use this new arrival as their new access point to the network, depending on if this adjustment is beneficial to the networks. This is detailed below.

When a node u first arrives at the network, it will join the network according to the procedure of node arrival as described above. It will then broadcast a *join finished* message to the neighbors with its *unit cost to sink*. All neighbors, except u's parent, consider u as a new candidate of access point and verify whether node u can introduce more energy saving based on the online algorithm described in Section 4.1. If a neighbor determines that selecting node u as its new parent is more beneficial, i.e., using u as the fusion point introduces less additional cost to the network than using its old parent, this neighbor will select node u as its new parent. Owing to this reason, assume that neighbor node v changes its parent from node p

TABLE 1
Change of Node's Weight for Source Arrival and Departure

	u joins in	s joins in	u leaves	s reconnects
current weight of v	$w(v) + w(u)(1 - \rho_{vu})$	$w(v) + w(u)(1 - \rho_{vu}) + w(s)(1 - \rho_{us})$	$w(v)$	$w(v) + w(s)(1 - \rho_{vs})$
additional weight of v's child	$u : w(u)(1 - \rho_{vu})$	$u : w(u)(1 - \rho_{vu}) + w(s)(1 - \rho_{us})$	0	$s : w(s)(1 - \rho_{vs})$
current weight of u	$w(u)$	$w(u) + w(s)(1 - \rho_{us})$		
additional weight of u's child	0	$s : w(s)(1 - \rho_{us})$		

to node u, node v shall send a *child departure* message to its old parent, p, with v's *current weight*, then node p and all nodes on the path from p to sink t can update their routing parameters accordingly. Meanwhile, node v should send a *source join* message to its new parent, u, according to the source arrival procedure to update the routing parameters of all nodes on the path from u to sink t. At the same time, node v should also send a *change parent* message to all its direct children to update the routing parameters of all nodes in the subtree rooted at v according to the source departure procedure.

As this routing adjustment may introduce potentially heavy routing maintenance overhead, we confine it only to be in the direct neighborhood of the new node. As a result, the adjustment will not be spread to the whole network and the routing structure change will be confined as well.

6.5 Simultaneous Arrivals and Departures

In wireless sensor networks, multiple failures can occur simultaneously and multiple nodes can also arrive simultaneously when performing network repairing or redeployment. Using only the basic procedures of source arrival and departure will keep the existing tree structure roughly unchanged. As a result, the routing tree constructed by the online algorithm may be significantly different from the one derived from offline AFST after multiple node arrivals and departures. Fortunately, this difference can be dramatically reduced by performing the procedure of the aforementioned routing adjustment. Doing this, the possibility of mutual selection of each other as best access points can be avoided. This can be illustrated using the example below.

Assume that two physically proximate nodes, u and v, arrive simultaneously. They will first perform the basic node arrival procedure to find their best access points to the existing networks, respectively. Note that it is possible that both u and v adopt simply shortest paths to the sink t without any aggregation if fusion benefit is nonexistent using the existing tree. Subsequently, both nodes will broadcast the *join finished* message. Assume that node u first successfully sends out its *join finished* message. When node v receives this message, it will stop broadcasting its own *join finished* message and performs the procedure of routing adjustment. Hence, node v has the chance to select u as its fusion point for cost reduction. As only one node can successfully broadcast the message at one time, this scheme can automatically sequence neighboring nodes arrivals and avoid the possibility of mutual selection of each other as network access points. In the case of multiple failures, remaining sensor nodes can execute the procedure of node departure to determine their new parents in the existing tree. As a result, the fusion tree can be repaired immediately.

We remark that the distributed algorithm closely approximates the performance of the online algorithm as the only difference between them is the potential candidate set, which is rather small based on our previous discussion and confined routing adjustment in the neighborhood only.

7 Experimental Study

In this section, we present an extensive set of simulations to evaluate the performance of the proposed online algorithm comparing with the offline AFST algorithm. We study the impact of network connectivity, correlation coefficient, unit fusion cost, and the number of arrivals on both algorithms. Our key finding of the experiments is that the online algorithm can adapt itself to a wide range of scenarios while maintaining small performance difference from the offline version. Indeed, the difference between online and offline algorithms is consistently within 15 percent. In some extreme cases, the difference is less than 2 percent. Furthermore, we find that the difference exhibits logarithmic property to the arrival number, indicating that the competitive ratio of the online algorithm is logarithmic to the number of arrived sources.

7.1 Simulation Environment

In our setup, up to 100 sensor nodes are uniformly distributed in a region of a 50×50 m square. All sensors act as both sources and routers. In the experiments of offline algorithm, all source nodes are deployed at the beginning and we perform the offline algorithm once to obtain the data gathering tree. On the contrary, for the online algorithm, a fraction of nodes are deployed first and the offline AFST algorithm is performed to obtain the initial tree structure. The remaining nodes are then deployed one by one in random order which will trigger the online algorithm to expand the data gathering tree. Experiments of different numbers of sensors and different sizes of field are also performed. The results are similar and omitted here.

We assume the maximal communication radius of a sensor is r_c, and the unit transmission cost on each edge, $c(e) = \beta d^{\gamma} + \varepsilon$ when $d < r_c$, where d denotes the distance of edge e. By varying r_c, we can control the network connectivity and, hence, the network density. We set $\gamma = 2$ and $\beta = 100\ \mathrm{pJ/bit/m^2}$ to calculate the energy consumption on the transmit amplifier, and set $\varepsilon = 100$ pJ/bit to denote the energy consumption per bit on the transmitter circuit and receiver circuit according to [18]. A set of other values are also studied, which lead to similar results and are omitted here. We note here that the transmission cost can implicitly model noise and transmission errors as well: higher noise will lead to more retransmissions and, hence, higher unit transmission cost.

The correlation model employed here is an approximated spatial model where the correlation coefficient decreases with the distance between two nodes provided that they are within the correlation range, r_s. If two nodes are more than r_s distance apart, simply the correlation coefficient $\rho = 0$. Otherwise, it is given by $\rho = 1 - d/r_s$, where d denotes the distance between the nodes. By varying the correlation range r_s, we can control the average correlation coefficient of the network, and further control the average data reduction after data fusion. For the fusion cost, we assume that q is constant and use ω to denote the average fusion cost per bit at each node.

In the following sections, we will study the performance of the online algorithm under various system setups, including network connectivity, correlation coefficient, fusion cost, and number of new sources.

7.2 Impact of Network Connectivity

Since r_c denotes the communication range of a node, by varying r_c, we can control the connectivity of the network

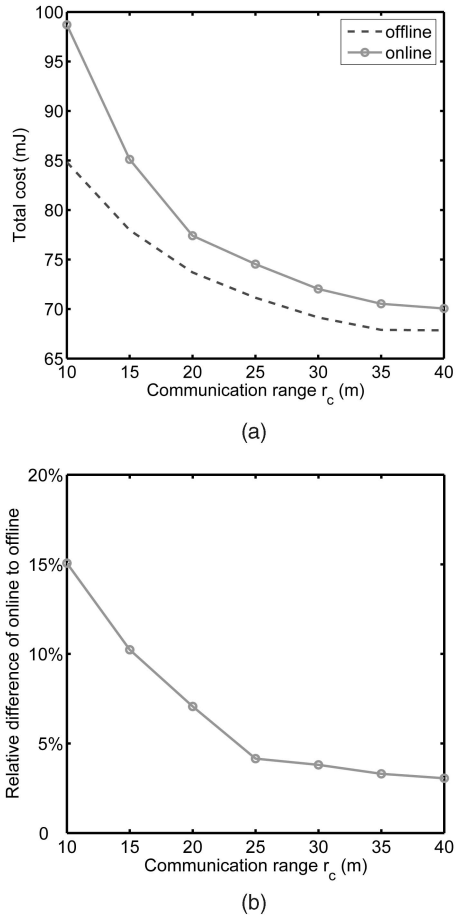


Fig. 7. Impact of communication range to energy consumption. (a) Total cost. (b) Difference range.

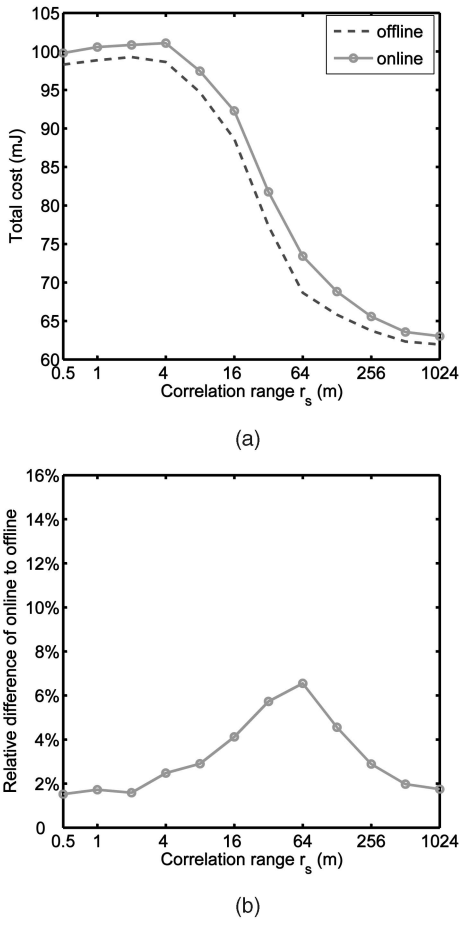


Fig. 8. Impact of correlation range to energy consumption. (a) Total cost. (b) Difference range.

and, hence, equivalently the density of the network. The larger r_c is, the more strongly the network is connected. Here, we set ω, the average unit fusion cost, to be 50 nJ/bit, which is a typical value as demonstrated by our experimental study for fusion cost described in [23]. And we set r_s, the correlation range, to be 50 m to simulate a network with moderate data reduction.

Fig. 7 summarizes the performance of both online and offline algorithms. Notice that the energy consumption of both algorithms decreases along with increased network connectivity. Notably, the difference between them decreases from 15 percent to 3 percent, which shows that the online algorithm actually benefits more from increased network connectivity and network density. The reason is that a new arrival can have more access candidates if the communication range is larger, and hence, the new arrival has larger chance of locating more suitable access point. Specifically, when $r_c < 25$ m, the difference between the two algorithms decreases very fast, from 15 percent to 4 percent, which means that the increase of network connectivity and network density can give the online algorithm more chance to locate the best access point. The result shows that in a dense sensor network, the online algorithm is an excellent solution approximating the offline version.

7.3 Impact of Correlation Coefficient

By varying the correlation range r_s, we can control the average correlation coefficient of the network, and further control the average data reduction after data fusion. For example, a very small r_s essentially eliminates the correlation among sensors $(\rho \to 0)$, while an extremely large r_s makes the sensed data completely redundant $(\rho \to 1)$. Fig. 8 illustrates the result when we increase the correlation range r_s from 0.5 to 1,000 m which corresponds to varying ρ from 0 to 1.

When $\rho \to 0$, both algorithms will connect each source to the sink through shortest path without fusion. While $\rho \to 1$, both algorithms will produce full aggregation trees. But in the online AFST, due to the randomly incoming order of new nodes and preservation of existing gathering tree structure, the path for an arrival u in online AFST may not be the same as that in the offline algorithm. Therefore, the expected performance of the online algorithm shall be worse than the offline version. Fortunately, from Fig. 8b, we can see that the relative differences are relatively small, which are within 2 percent in both cases.

In general $(0 < \rho < 1)$, the offline AFST can let each node either get its perfect pair-node, or connect to the sink through shortest path, only depending on the evaluation of fusion benefit. On the contrary, in the online algorithm, an arrival may not find its perfect pair-node since source nodes join in the network randomly. Indeed, an arrival may

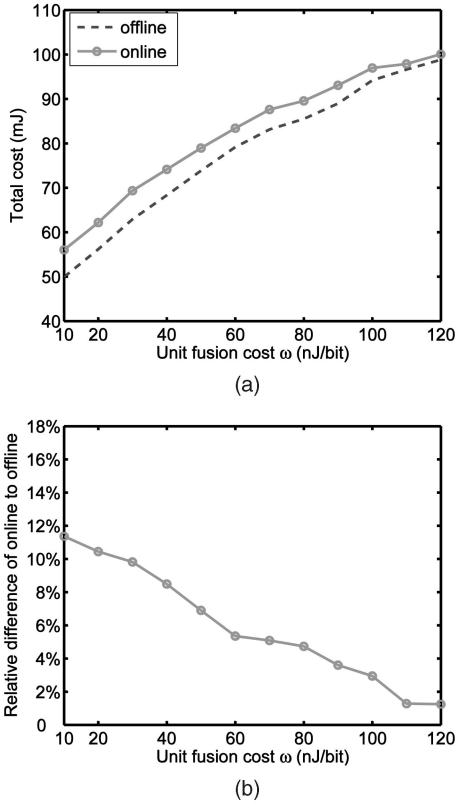


Fig. 9. Impact of unit fusion cost to energy consumption. (a) Total cost. (b) Difference range.

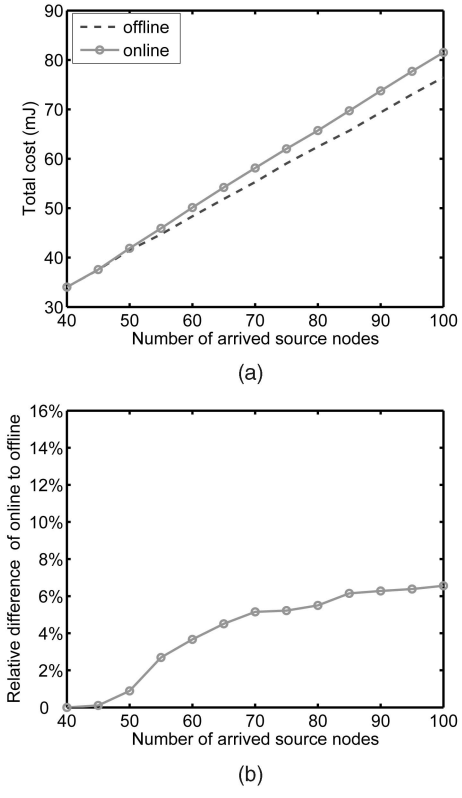


Fig. 10. Impact of the number of arrived sources. (a) Total cost. (b) Difference range.

choose connecting to the sink through shortest path if all its possible matching pair have not arrived yet. Therefore, the difference is larger as compared with both extreme cases: $\rho \to 0$ and $\rho \to 1$. However, during the whole range, the difference is consistently within 7 percent.

7.4 Impact of Unit Fusion Cost

In this simulation, we fix the transmission and correlation ranges of the sensor nodes to $r_c = 20\,\mathrm{m}$ and $r_s = 50\,\mathrm{m}$, respectively, and study the impact of unit fusion cost to the performance of online AFST. Fig. 9 illustrates the results when ω, the unit fusion cost, increases from 10 to 120 nJ/bit.

Obviously, the difference between the online and offline algorithms decreases with the increase of unit fusion cost. It can be reasoned as follows. When ω is small, the source node tends to perform data fusion but may not find the perfect pair-node when $0 < \rho < 1$ as discussed in the previous section. Therefore, online AFST shows a 12 percent difference from offline version. When the unit fusion cost increases, the fusion benefit decreases. As a result, more and more source nodes tend to connect to the sink directly through shortest path. Similar to the case of $\rho \to 0$ in the previous section, the performance difference decreases to be within 2 percent, if little data fusion is performed.

Fusion cost may vary widely from network to network, from application to application. Our experiments show that online AFST preserved the property of offline AFST, namely that it can adapt to a wide range of fusion costs and, hence, be applicable to a variety of applications.

7.5 Impact of Number of New Arrivals

In this set of experiments, we study the difference between online and offline algorithms when the number of new arrivals slowly increases. The network is initialized with 40 source nodes, and the other 60 source nodes arrive randomly one by one later. During this process, we run offline algorithm under current number source nodes to obtain its corresponding performance. At the same time, we run online algorithm based on the existing data gathering tree and the newcomer to get the online result. Fig. 10a illustrates the energy consumption of both algorithms when the number of sources increases from 40 to 100. Fig. 10b shows their performance difference.

As we can see, the difference increases when more sources arrive since the online algorithm can only expand the existing tree but the offline algorithm can completely rebuild the tree using perfect matching. Although the difference increases, it is still within the range of 7 percent when all source nodes arrive. Furthermore, from Fig. 10b, we find that the difference curve is concavely increasing and can be approximated using logarithmic function. Therefore, the online algorithm is also applicable to large-scale sensor networks.

8 Related Work

En route aggregation can be generally classified into two categories: routing driven and aggregation driven. Routing-driven algorithms [3], [4], [5], [7], [8] emphasize source compression at each individual node and aggregation occurs opportunistically when routes intersect. On the contrary, routing paths in aggregation-driven algorithms

[9], [10], [11], [12] are heavily dependent on data correlation in order to fully benefit from information reduction resulting from data aggregation. In [9], a hierarchical matching algorithm is proposed resulting in an aggregation tree with a logarithmic approximation ratio to the optimal for all concave aggregation functions. In [10], a randomized algorithm MFST is proposed that jointly optimizes over both the fusion and transmission costs to minimize overall energy consumption. MFST is proved to achieve a routing tree that exhibits $\frac{5}{4}log(k+1)$ approximation ratio to the optimal solution, where k denotes the number of source nodes. The authors of [12] proposed an optimal algorithm MEGA for foreign-coding and an approximating algorithm LEGA for self-coding. In MEGA, each node sends raw data to its encoding point using directed minimum spanning tree (DMST), and encoded data is then transmitted to the sink through SPT. LEGA uses shallow light tree (SLT) [27] as the data gathering topology.

Indeed, the idea of embedding fusion decisions in routing has been implicitly explored in the literature. For example, LEACH [3] is a cluster-based protocol, in which sensors directly send data to cluster heads where data fusion is performed. Aggregated data is then delivered to the sink through multihop paths. LEGA and MEGA [12] implicitly assume that fusion stops after first aggregation as encoded data cannot be recoded again. However, the decision regarding fusions in these schemes are rather static and cannot adapt to network/data structure changes.

While AFST has been presented in our prior work [23] targeting at en route aggregation decision, it is an offline algorithm requiring global information and static networks.

9 Conclusion and Future Work

In this paper, we have proposed online AFST, an online algorithm for en route aggregation decision for gathering correlated data in sensor networks. Online AFST not only optimizes over both the transmission and fusion costs by adaptively adjusting its fusion decisions but also executes in an online fashion that allows nodes to dynamically join or leave the network. Due to its special design, online AFST can be readily implemented in a distributed fashion that requires only localized information, which is in deep contrast to the offline version. Furthermore, our analytical and experimental results show that this online algorithm deviates little from its offline ancestor in terms of network energy consumption.

Currently, we are implementing the proposed algorithm in real sensor networks and using image fusion as the targeted application.

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful comments which helped improve the technical quality of this paper. Hong Luo was a visiting scholar with Yonghe Liu at the University of Texas at Arlington while the work was partially performed. This work was partially supported by US National Science Foundation Grant CNS-0721951 and the Foundation for Western Returned Chinese Scholars of the Ministry of Education.

References

[1] C. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE,* vol. 91, no. 8, pp. 1247-1256, Aug. 2003.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* vol. 40, no. 8, pp. 102-114, Aug. 2002.

[3] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. 33rd Ann. Hawaii Int'l Conf. System Sciences (HICSS '00),* Jan. 2000.

[4] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of Data Aggregation in Wireless Sensor Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02),* pp. 575-578, July 2002.

[5] A. Scaglione and S.D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," *Proc. ACM MobiCom '02,* Sept. 2002.

[6] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," *Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN '04),* Apr. 2004.

[7] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02),* July 2002.

[8] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Trans. Networking,* vol. 11, no. 1, pp. 2-16, Feb. 2003.

[9] A. Goel and D. Estrin, "Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk," *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '03),* Jan. 2003.

[10] H. Luo, Y. Liu, and S.K. Das, "Routing Correlated Data with Fusion Cost in Wireless Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 11, Nov. 2006.

[11] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On Network Correlated Data Gathering," *Proc. IEEE INFOCOM '04,* Mar. 2004.

[12] P.V. Rickenbach and R. Wattenhofer, "Gathering Correlated Data in Sensor Networks," *Proc. ACM Joint Workshop Foundations of Mobile Computing (DIALM-POMC '04),* Oct. 2004.

[13] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-Latency Tradeoff for Data Gathering in Wireless Sensor Networks," *Proc. IEEE INFOCOM '04,* Mar. 2004.

[14] S. Lindsey and C.S. Raghavendra, "Pegasis: Power-Efficient Gathering in Sensor Information Systems," *Proc. IEEE Aerospace Conf.,* Mar. 2002.

[15] H. Gupta, V. Navda, S.R. Das, and V. Chowdhary, "Efficient Gathering of Correlated Data in Sensor Networks," *Proc. ACM MobiHoc '05,* May 2005.

[16] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An Efficient Clustering-Based Heuristic for Data Gathering and Aggregation in Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03),* Mar. 2003.

[17] C. Buragohain, D. Agrawal, and S. Suri, "Power Aware Routing for Sensor Databases," *Proc. IEEE INFOCOM '05,* Mar. 2005.

[18] A. Wang, W.B. Heinzelman, A. Sinha, and A.P. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated Microsensor Networks," *J. VLSI Signal Processing,* vol. 29, no. 3, pp. 223-237, Nov. 2001.

[19] D.W. Carman, P.S. Kruus, and B.J. Matt, "Constraints and Approaches for Distributed Sensor Network Security," Technical Report 00-010, NAI Labs, Sept. 2000.

[20] R.S. Wagner, R.G. Baraniuk, S. Du, D.B. Johnson, and A. Cohen, "An Architecture for Distributed Wavelet Analysis and Processing in Sensor Networks," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN '06),* pp. 243-250, Apr. 2006.

[21] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-Efficient Data Representation and Routing for Wireless Sensor Networks Based on a Distributed Wavelet Compression Algorithm," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN '06),* pp. 309-316, Apr. 2006.

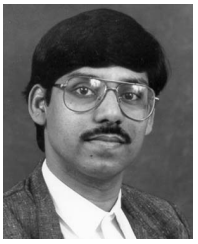
[22] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive Wireless Sensing," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN '06),* pp. 134-142, Apr. 2006.

[23] H. Luo, J. Luo, Y. Liu, and S.K. Das, "Adaptive Data Fusion for Energy Efficient Routing in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 55, no. 10, Oct. 2006.

[24] D. Lee, H. Kim, S. Tu, and M. Rahimi, "Energy-Optimized Image Communication on Resource-Constrained Sensor Platforms," *Proc. Sixth Int'l Conf. Information Processing in Sensor Networks (IPSN '07)*, pp. 216-225, Apr. 2007.
[25] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications Inc., pp. 247-266, 1998.
[26] T. Rappaport, *Wireless Communications: Principles and Practice*, second ed. Prentice Hall, 2001.
[27] A. Goel and K. Munagala, "Balancing Steiner Trees and Shortest Path Trees Online," *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '00)*, Jan. 2000.

Hong Luo received the BS, MS, and PhD degrees from Beijing University of Posts and Telecommunications, in 1990, 1993, and 2006, respectively. She is an associate professor in the School of Computer Science and Technology, Beijing University of Posts and Telecommunications. She is also a research member of the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia. Her research interests include wireless networking, sensor networks, and communication software. She is a member of the IEEE.

Yonghe Liu received the BS and MS degrees from Tsinghua University, in 1998 and 1999, respectively, and the PhD degree from Rice University, in 2004. He is an assistant professor in the Department of Computer Science and Engineering, University of Texas at Arlington. His research interests include wireless networking, sensor networks, security, and system integration. He is a member of the IEEE.



Sajal K. Das is a professor of computer science and engineering in the Department of Computer Science and Engineering, University of Texas at Arlington, (UTA) and also the founding director of the Center for Research in Wireless Mobility and Networking (CReWMaN), UTA. His current research interests include sensor networks, resource and mobility management in wireless networks, mobile and pervasive computing, wireless multimedia and QoS provisioning, mobile internet architectures and protocols, grid computing, and applied game theory. He has published more than 350 research papers in these areas and holds four US patents in wireless internet and mobile networks. He received the Best Paper Awards in ACM MobiCom '99, ICOIN '02, ACM MSwiM '00, and ACM/IEEE PADS '97. He is also a recipient of UTA's Outstanding Faculty Research Award in Computer Science (2001 and 2003), College of Engineering Research Excellence Award (2003), and University Award for Distinguished Record of Research (2005). He serves as the editor-in-chief of *Pervasive and Mobile Computing* journal, and an associate editor of the *IEEE Transactions on Mobile Computing*, *ACM/Springer Wireless Networks*, and the *IEEE Transactions on Parallel and Distributed Systems*. He has served as a general or a program chair and a TPC member of numerous IEEE and ACM conferences. He is the vice chair of the IEEE TCCC and TCPP Executive Committees. He is a member of the IEEE.