Distributed Algorithm for Tree-Structured Data Aggregation Service Placement in Smart Grid

Zongqing Lu, *Student Member, IEEE*, and Yonggang Wen, *Member, IEEE*

***Abstract*—Smart grid, regarded as the next generation of power grid, uses two-way flows of electricity and information to create a widely distributed automated energy delivery network. One distinguishing aspect of smart grid communication networks is the large-scale deployment of smart meters and sensors. Thus, a large amount of data and information will be generated from metering, sensing, monitoring, etc. Data aggregation (integration or fusion) aims at the merging of data from disparate sources with differing conceptual, contextual, and typographical representations. In order to satisfy the requirement of different information services, the placement of data aggregation services in smart grid communication networks is a critical design issue. In this paper, we propose a minimum-cost-forwarding-based asynchronous distributed algorithm to find the optimal placement for the data aggregation service tree with optimal cost of in-network processing. It is shown that minimum-cost forwarding can dramatically reduce message overheads of the asynchronous algorithm. It is also shown that our algorithm has less message overheads than the synchronous algorithm (Sync) by both mathematical analysis and simulation-based evaluation. For a regular grid network and a complete binary service tree, the messages sent at each node are $O(\sqrt{N}M)$ for our proposed algorithm and $O(\sqrt{N}M\log_2 M)$ for the Sync, where N is the number of network nodes and M is the number of data objects in the service tree.**

***Index Terms*—Distributed algorithm, minimum-cost forwarding, service tree placement, smart grid.**

I. Introduction

SMART GRID, considered as the next generation of power grid, uses two-way flows of electricity and information to create a widely distributed automated energy delivery network. In this network, the amount of electricity generated can be adjusted according to the real-time demand of consumers. This not only ensures that consumer demands are satisfied but also avoids excess electricity generation. The latter can help increase the profit of the power operators and protect the environment. An intense research and design effort is under way to define this future energy grid [1]–[3].

Smart grid can deliver power in more efficient ways and respond to wide-ranging conditions and events. It is designed to handle any event that occurs anywhere in the grid, such as power generation, transmission, distribution, and consumption. More specifically, smart grid can be regarded as an electric system that uses information in two-way, [4] and cyber-secure communication technologies [5], and computational intelligence in an integrated fashion across electricity generation, transmission, substations, distribution, and consumption to achieve a system that is clean, safe, secure, reliable, resilient, efficient, and sustainable.

The evolution of smart grid relies on not only the advancement of power equipment technology but also the improvement of sophisticated computer monitoring, information aggregation, information analysis, optimization, and control from exclusively central utility locations to the distribution and transmission grids. In addition, wireless communication is utilized to support this two-way information flow between the various entities in smart grid. Moreover, smart grid must support advanced information management including data modeling, information aggregation, information analysis, etc. [1].

In smart grid, one distinguishing aspect of smart grid communication networks is the large-scale deployment of sensors and smart meters [6]. Thus, a large amount of data and information will be generated from metering, sensing, monitoring, etc. Information aggregation (integration or fusion) aims at the merging of data from disparate sources with differing conceptual, contextual, and typographical representations [1], [7]. As a large amount of data is generated in smart grid communication networks, data aggregation must take place in the designated place to satisfy the different requirements of information flow, for example, information from meters should be real-time delivered and processed for power demanding management and pricing, and the quality of service and energy optimization should be provided for monitoring using sensor networks. Thus, the placement of data aggregation services in smart grid communication networks is a critical design problem.

These data aggregation services in smart grid can be structured as a tree, for example, first, the metering information in one district should be aggregated at the district aggregation center, as shown in Fig. 1, then the information of different aggregation centers should be fused in city aggregation centers, and so on. Therefore, these aggregations can be structured hierarchically as tree services.

In this paper, we propose a distributed algorithm to find the placement of data aggregation trees in smart grid communication networks. We choose the nodes with the lowest costs for performing tree-structured services as data aggregation placements. Since our algorithm is designed for a general model, the metric can be various and can depend on different requirements of services, for instance, energy cost and latency. Our research work explores a minimum-cost-forwarding-based asynchronous distributed solution for finding the optimal placement for a service tree. Our approach achieves two goals: 1) finding the optimal placement of the service tree with the

Manuscript received April 11, 2012; revised September 4, 2012; accepted January 23, 2013.

The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: luzo0002@ntu.edu.sg; ygwen@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2013.2260917

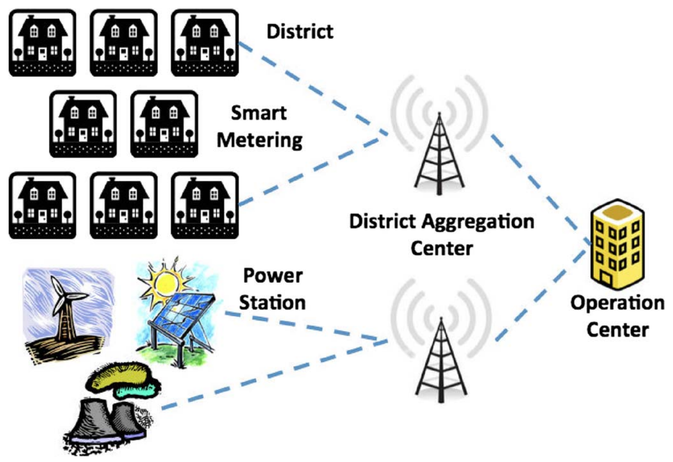


Fig. 1. Information aggregation in smart grid communication network.

optimized cost of in-network processing by doing so in an asynchronous mode; and 2) reducing the message overheads of the searching process of optimal placement.

The contributions of this work are summarized as follows:

1) an asynchronous distributed solution that can achieve optimal placement of the aggregation service tree by exchanging information among network nodes;
2) a minimum-cost-forwarding-based asynchronous algorithm (MCFA) that reduces the message overheads of searching the optimal placement;
3) mathematical analysis and simulation-based evaluation which show that MCFA achieves a significant reduction in the message overheads of searching the optimal placement compared with the synchronous algorithm (Sync).

The rest of this paper is structured as follows. Section II summarizes related work. Section III formulates the optimization problem for data aggregation service tree placement with optimal cost of in-network processing. Section IV describes the straightforward asynchronous algorithm (Async). Then, the design details and the analysis of MCFA are shown in Section V. Section VI presents the simulation-based evaluation of our proposed algorithms. At last, we conclude the work in Section VII.

II. Related Work

Existing algorithms on data aggregation (fusion or service) placement or mapping in a network can be classified into two categories: centralized algorithm and distributed algorithm. In centralized algorithms, each node is required to have the complete topology information of the entire network, limiting its scalability in large networks. Those in [8]–[11] are in this category. Distributed algorithms can relax this constraint; however, they either cannot find the optimal placement, result in huge message overheads, or have another requirement from the network. Those in [12]–[17] are distributed algorithms.

Bonfils and Bonnet [12] proposed a decentralized algorithm for aggregator placement, which progressively refines the placements of aggregators by neighbor exploration and placement adaptation. The approach that an aggregator is gradually moved toward optimal placement is called in-network relaxation or aggregator migration. References [13] and [14] are other works in the same category.As in these algorithms the aggregator migration is only based on local information (information from neighbors), they suffer from oscillating change, which might force the placement of an aggregator to a different direction before reaching the optimal placement. They are also prone to local minima, and they cannot guarantee the optimality of aggregator placement based on local information only.

In [15], a one-median point is considered as the optimal placement in the network, and a distributed search algorithm is proposed to find the optimal aggregator placement. Surprisingly, this algorithm is designed to handle only one aggregator placement.

Abrams and Liu [17] proposed a greedy algorithm, which places each aggregator on the node with minimized input data cost. Obviously, the greedy placement is not the optimal placement, and it can be much worse when the greedy placement is backward to the sink. Furthermore, the distributed implementation of the greedy algorithm is only about placement adaptation, and the authors do not elaborate how to find the initial placement for each aggregator in a distributed manner.

In [16], a distributed algorithm is proposed to achieve the optimal placement for a tree-structured query graph with minimized total cost of storage, computation, and data transmission, by exchanging information for hosting query aggregators among network nodes. However, the proposed algorithm requires that the network should have full time synchronization, and all the nodes should know when other nodes finish information updating and finish broadcasting updated information. Such Syncs are difficult and costly to be implemented into large-scale networks like smart grid. Furthermore, it has large message overheads for searching the optimal tree placement $O(\sqrt{N}M\log_2 M)$, where N is the number of network nodes and M is the number of data objects in the service tree.

III. Problem Formulation

This work aims to find the optimal placement for a service tree in a smart grid communication network with the optimal cost of in-network processing. We consider the network of information flow in smart grid as a undirected graph, where vertices represent network nodes and edges represent communication links, and services as a tree consisting of services and data objects. In this paper, we use aggregator or service alternatively to denote data aggregation service. We define the following.

1) Given a network as a graph $G_N = (\zeta, \pi)$, ζ denotes the set of vertices representing network nodes. π denotes the set of communication links between nodes. For each node $p, q \in \zeta$ in the communication radius, we denote the edge as $(p, q) \in \pi$.
2) Given a service tree $G_T = (\eta, \gamma, \delta)$, η denotes the set of services. γ denotes the set of communication dependences connecting the services. δ denotes the set of data objects. For $k \in \delta$, we denote d_k as the size of data object k. As the service tree is oriented, each service has one or more children and, at most, one parent. Data objects can be divided into three categories: source data object (for example, generated by smart meters or sensors; data objects 1, 2, 3, and 4), immediate data object

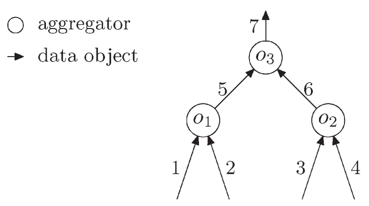


Fig. 2. Service tree.

(generated by service and transmitted to the next service; data objects 5 and 6), and final data object (generated by the aggregator and transmitted to the information center; data object 7), as shown in Fig. 2. H_T is the depth of the service tree, which is defined as the maximum number of operations needed for the source data to become the final data, for example, $H_T = 2$ in Fig. 2. We denote the set of data objects used to generate data object k by Δ_k ($\Delta_5 = \{1, 2\}$ in Fig. 2); Δ_k represents the children of k.

3) P denotes the service placement, such as $P(k, p) = 1$ if data object k is generated at node p (the service generating k is placed on node p) and $P(k, p) = 0$ if otherwise.
4) R denotes how to route the data from the nodes generating the data to the node requiring the data, for example, all the data of Δ_k need to transmit to the node that generates data object k.

Data Transmission Cost $f_T(P, R, G_N, G_T)$—Unit Data Transmission Cost Under R: We denote $C_T(p, q)$ as a chosen cost metric between nodes p and q. As discussed previously, since we aim to provide a distributed solution for a general model, we do not specify the cost metric. Examples of the cost metric include delay, transmission energy, hop counts, and euclidean distance between two nodes.

Computation Cost $f_c(P, R; G_N, G_T)$—Unit Data Computation Cost Under P: We denote $C_c(k, p)$ as the computation cost for generating unit data k at node p. For source data k, $C_c(k, p) = 0$ if p is the source node generating k; otherwise, $C_c(k, p) = \infty$. The cost metric can also be various like data transmission cost.

Hence, the total cost of in-network processing is

$$f_{\text{cost}}(P, R; N, T) = (f_T + f_c)(P, R; G_N, G_T). \tag{1}$$

The problem, minimizing the cost of performing service tree in-network processing, is to determine the process scheme (P^*, R^*) that solves (2) for a given network and service tree. It is a joint problem of routing and placement

$$(P^*, R^*) = \arg \min_{(P,R)} f_{\text{cost}}(P, R; G_N, G_T). \tag{2}$$

IV. Straightforward Asynchronous Distributed Algorithm

In this section, we give the straightforward asynchronous distributed algorithm to solve the optimization problem (2) inspired by Ying *et al.* [16].

Our proposed algorithm requires every network node to maintain the following information for each data object of the service tree:

1) $C(k, p)$: the current lowest cost of acquiring data object k at node p;
2) $P(k, p)$: the indication of whether data object k is generated at node p, i.e., $P(k, p) = 1$ if so or $P(k, p) = 0$, otherwise;
3) $O(k, p)$: the indication of whether data object k is acquirable at node p, i.e., $O(k, p) = 1$ if so or $O(k, p) = 0$ if otherwise (if $C(k, p)$ is ∞, $O(k, p) = 0$);
4) $R(k, p)$: the indication of from which node does node p acquired data object k with the current lowest cost and whether k is generated by p.

There are two approaches for computing the acquiring cost at network node. If data object k is transmitted from neighbor q

$$C(k, p) = C(k, q) + d_k C_T(p, q). \tag{3}$$

[1]If data object k is generated by p

$$C(k, p) = \sum_{m \in \Delta_k} C(m, p) + C_c(k, p). \tag{4}$$

By exchanging the cost for acquiring each data object of the service tree among network nodes, updating the cost using (3) and (4) if it can be improved, and broadcasting updated information to its neighbors, every network node will get the least cost for acquiring each data object of the service tree, eventually.

Algorithm 1: Straightforward Algorithm

```
Event: Node $p$ receives ADV from node $q$ for data object $k$
1 begin
2     $C'(k,p) = C(k,q) + d_k C_T(p,q)$
3     if $C'(k,p) < C(k,p)$ then
4         replace $C(k,p)$ with $C'(k,p)$
5         $O(k,p) = 1$
6         $R(k,p) = q$
7         broadcast ADV containing $C(k,p)$ for $k$
8         for each ancestor $k'$ of $k$; // from bottom up
9         do
10            if $\Delta_{k'}$ are acquirable then
11                $C'(k',p) = \sum_{m \in \Delta_{k'}} C(m,p) + C_c(k',p)$
12                if $C'(k',p) < C'(k',p)$ then
13                    replace $C(k',p)$ with $C'(k',p)$
14                    $O(k',p) = 1$
15                    $P(k',p) = 1$
16                    broadcast ADV containing $C(k',p)$
                      for $k'$
17                end
18            else
19                return
20            end
21        end
22    end
23 end
```

[1]Equations (3) and (4) are highly customizable according to different cost metrics and service requirements. In this paper, we use these two popular terms as an example to illustrate the cost computations. Here, the transmission cost of a data object is proportional to the size of the data object.

After all network nodes initialize the cost for all the data objects, each data source node initiates the searching process by broadcasting an advertisement (ADV) message to its neighbors. The ADV contains the following information:

$$\langle \ data\_object, acquiring\_cost \ \rangle$$

where $acquiring\_cost$ is the cost of acquiring $data\_object$ at the sender of ADV. ADV for data object k is firstly broadcast by the source node generating k with an initial acquiring cost of zero. Algorithm 1 shows the straightforward solution to find the optimal placement of the service tree. When network node p receives ADV from its neighbor q for data object k, it first computes acquiring cost using (3), updates the acquiring cost, and broadcasts ADV right after getting less acquiring cost (steps 4–7) if computed cost is less than the recorded one or discards the ADV, otherwise. Then, it updates acquiring cost for the ancestors of k in a bottom-up manner using (4) and broadcasts ADV if the acquiring cost has been changed (steps 10–20).

Searching the optimal placement for the service tree terminates when there is no information exchanged among network nodes. Then, the network can execute a trace-back algorithm to solve in-network processing with minimum cost using $P(k, p)$ and $R(k, p)$ kept by nodes as follows.

1) The trace-back process is initialized at the sink node (information center) by sending a trace message about final data object k^* to $R(k^*, \text{sink})$.
2) If node p receives a trace message about k from node q, node p sends data object k to q when k is available. In addition, if $P(k, p) = 1$, node p sends a trace message about k'' to $R(k'', p)$ for each $k'' \in \Delta_k$ and generates k when Δ_k is available; if $P(k, p) = 0$, node p sends a trace message about k to $R(k, p)$.

Unlike the Sync that requires all the nodes to be fully synchronous, the asynchronous algorithm only requires nodes to broadcast updated information right after they received a lower acquiring cost. Hence, asynchronous mode may incur more message overheads than the synchronous one. In the next section, we will discuss how to reduce the message overhead of the asynchronous algorithm.

V. MCFA

The reason that the asynchronous algorithm has more message overhead is that the nodes broadcast immediately after obtaining a lower acquiring cost, no matter whether the cost is optimal or not. If we can delay the broadcast at the node to the time after it has received ADV with the optimal cost, the node may broadcast an ADV message only once, carrying its optimal cost. Thus, how long the node defers its broadcast becomes critical. Our minimum-cost forwarding algorithm sets the total delay time to be proportional to the transmission cost from the sender to the receiver

$$T_{\text{delay}} = \lambda C_T(p, q). \tag{5}$$

We first use Fig. 3 as an example to illustrate how the minimum-cost forwarding algorithm does work to reduce the ADV broadcast for the source data object.

1) At time t, node a broadcasts ADV that includes the acquiring cost $C(k, a) = C_a$ for data object k, and d_k is assumed as one for simplifying the calculation. After nodes b, c, and d receive ADV from node a, they set the acquiring cost for data object k to $C_a + 4$, $C_a + 2$, and $C_a + 5$, respectively (assuming that the initial cost of nodes b, c, and d is ∞). Then, each of them sets a timer for rebroadcasting ADV. The expiration period is proportional to the transmission cost between the sender and receiver. For nodes b, c, and d, the expiration periods are 4λ, 2λ, and 5λ, respectively. If the straightforward algorithm was used, nodes b, c, and d would broadcast an ADV message right after they received ADV from node a since they got less acquiring cost than ∞ (as shown from step 4 to 7 in Algorithm 1).
2) At time $t + 2\lambda$, the timer of node c expires. Node c finalizes the acquiring cost for data object k $(C(k, c) = C_a + 2)$ and broadcasts an ADV message including $C(k, c)$. When node d receives this ADV, as $C_a + 5 > C(k, c) + 2.5 = C_a + 4.5$, node c updates the cost to $C_a + 4.5$ and resets the timer to 2.5λ (note that the previous timer does not expire by the time $t + 2\lambda$ and, if the straightforward algorithm was implemented, node a would broadcast the second ADV message at this time). For node a and b, as $C_a < C_c + 2$ and $C_a + 4 < C(k, c) + 3 = C_a + 5$, they simply discard this ADV message.
3) At time $t + 4\lambda$, the timer of node b expires. Node b finalizes $C(k, b) = C_a + 4$ and broadcasts an ADV message containing $C(k, b)$. All other nodes will discard this ADV message because they have already received a lower acquiring cost.
4) At time $t + 4.5\lambda$, the timer of node d expires. Node d finalizes $C(k, d) = C_a + 4.5$ and broadcasts an ADV message with its minimum acquiring cost.

For the minimum-cost forwarding algorithm, we can observe from Fig. 3 that each node only broadcasts the ADV message for data object k once with the optimal cost and reduces the nonoptimal ADV message broadcasts. However, for the straightforward algorithm, the nodes will always advertise an ADV message when it gets less acquiring cost, and it is totally six in the scenario in Fig. 3 for the worst case.

The minimum-cost forwarding algorithm has the following two properties for the source data object, as shown in [18].

1) Each node only broadcasts the optimal acquiring cost of the data object to its neighbors and discards all redundant or nonoptimal ADV messages.
2) Nodes can get the minimum acquiring cost of the data object by only one ADV message broadcast at each node.

For data object k generated by the aggregator (immediate data object and final data object), as the aggregator generating k can be placed on any network node (k can be generated at any node), the minimum acquiring cost of k at different nodes might associate with different aggregator placements (unlike the source data object that is only generated by a certain

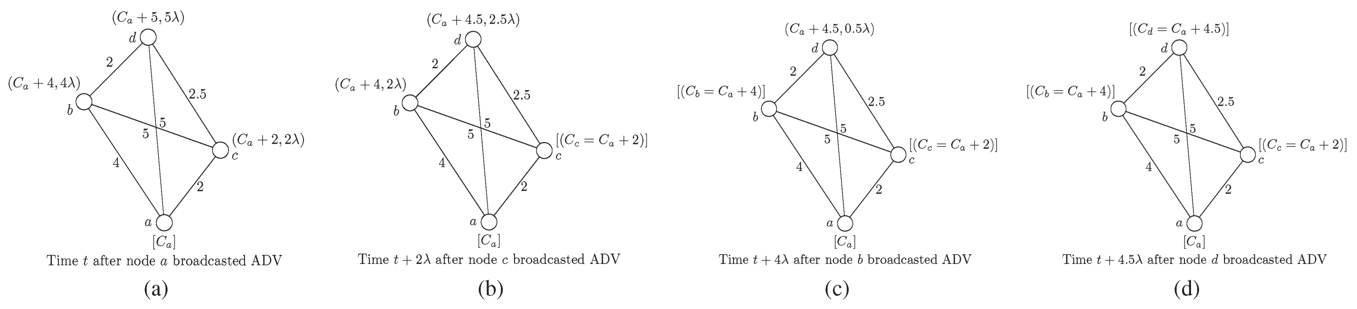


Fig. 3. Illustrative example of minimum-cost forwarding algorithm for source data object.

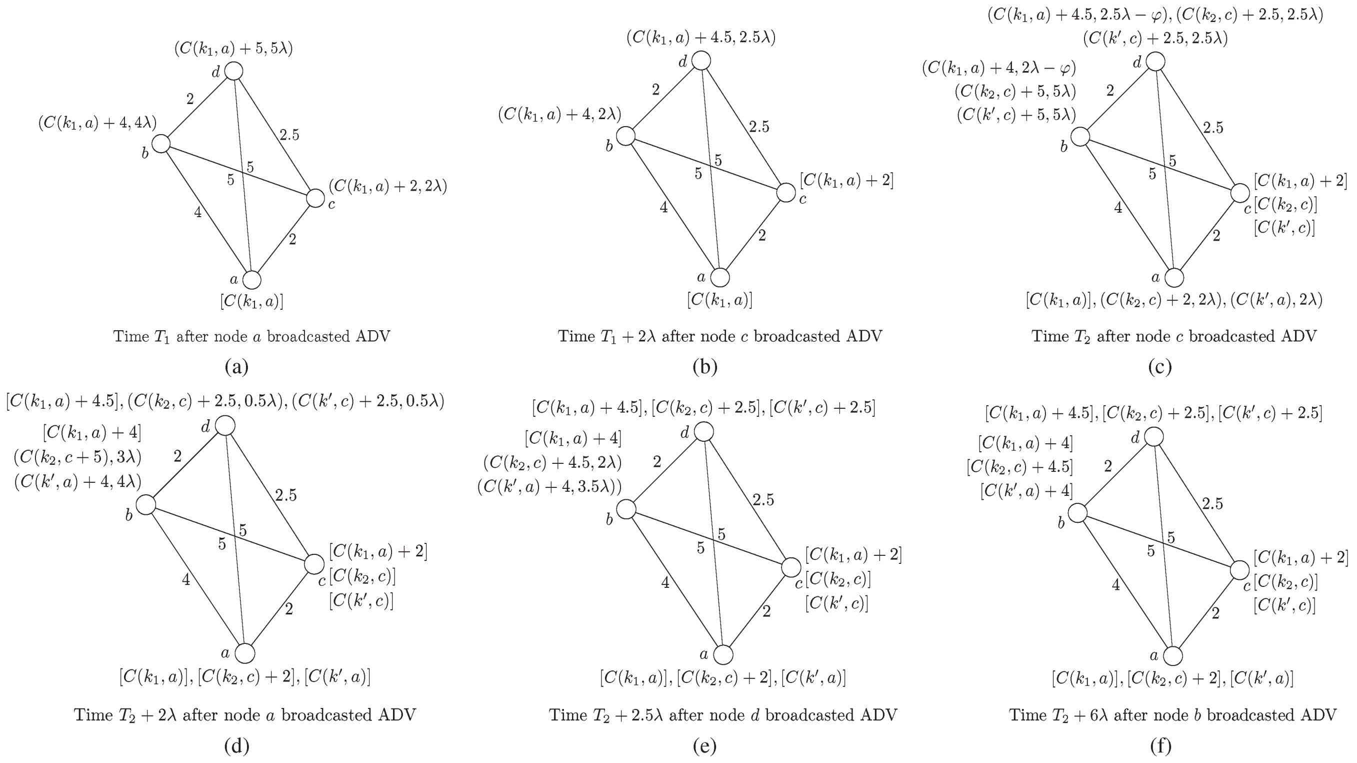

$(C(k_1,a)+5,5\lambda)$
$(C(k_1,a)+4,4\lambda)$
$(C(k_1,a)+2,2\lambda)$
$[C(k_1,a)]$
Time T_1 after node a broadcasted ADV
(a)
$(C(k_1,a)+4.5,2.5\lambda)$
$(C(k_1,a)+4,2\lambda)$
$[C(k_1,a)+2]$
$[C(k_1,a)]$
Time $T_1+2\lambda$ after node c broadcasted ADV
(b)
$(C(k_1,a)+4.5,2.5\lambda-\varphi),(C(k_2,c)+2.5,2.5\lambda)$
$(C(k',c)+2.5,2.5\lambda)$
$(C(k_1,a)+4,2\lambda-\varphi)$
$(C(k_2,c)+5,5\lambda)$
$(C(k',c)+5,5\lambda)$
$[C(k_1,a)+2]$
$[C(k_2,c)]$
$[C(k',c)]$
$[C(k_1,a)],(C(k_2,c)+2,2\lambda),(C(k',a),2\lambda)$
Time T_2 after node c broadcasted ADV
(c)
$[C(k_1,a)+4.5],(C(k_2,c)+2.5,0.5\lambda),(C(k',c)+2.5,0.5\lambda)$
$[C(k_1,a)+4]$
$(C(k_2,c+5),3\lambda)$
$(C(k',a)+4,4\lambda)$
$[C(k_1,a)+2]$
$[C(k_2,c)]$
$[C(k',c)]$
$[C(k_1,a)],[C(k_2,c)+2],[C(k',a)]$
Time $T_2+2\lambda$ after node a broadcasted ADV
(d)
$[C(k_1,a)+4.5],[C(k_2,c)+2.5],[C(k',c)+2.5]$
$[C(k_1,a)+4]$
$(C(k_2,c)+4.5,2\lambda)$
$(C(k',a)+4,3.5\lambda))$
$[C(k_1,a)+2]$
$[C(k_2,c)]$
$[C(k',c)]$
$[C(k_1,a)],[C(k_2,c)+2],[C(k',a)]$
Time $T_2+2.5\lambda$ after node d broadcasted ADV
(e)
$[C(k_1,a)+4.5],[C(k_2,c)+2.5],[C(k',c)+2.5]$
$[C(k_1,a)+4]$
$[C(k_2,c)+4.5]$
$[C(k',a)+4]$
$[C(k_1,a)+2]$
$[C(k_2,c)]$
$[C(k',c)]$
$[C(k_1,a)],[C(k_2,c)+2],[C(k',a)]$
Time $T_2+6\lambda$ after node b broadcasted ADV
(f)
a
b
c
d
2
2.5
5
5
4
2

Fig. 4. Illustrative example for minimum-cost forwarding algorithm for data object generated by aggregator.

node). However, the minimum-cost forwarding algorithm still can effectively reduce the number of ADV broadcasts.

As shown in Fig. 4, assuming that data object k_1 is generated by node a at time T_1, data object k_2 is generated by node c at time T_2, and $\Delta'_k = \{k_1, k_2\}$. For convenient calculation, we also assume that $2\lambda < T_2 - T_1 < 3\lambda$ and d_{k_1}, d_{k_2}, and $d_{k'}$ are one.

1) Before T_2, when k_2 is generated, the cost for k_1 and the timer kept by the nodes are the same as in Fig. 3(a) and (b). As shown in Fig. 4(c), as k_1 is already available at node c when k_2 is generated at T_2, node c finalizes the costs for data object k_2 and k' and broadcasts ADV messages of $C(k_2,c)$ and $C(k',c) = C(k_1,a) + C(k_2,c) + 2$. Other nodes update the cost for each data object and set the ADV timer for each data object updated with less cost after receiving an ADV message from node c ($\varphi = T_2 - T_1 - 2\lambda$). The node also compares the acquiring cost of data object k' generated locally with the acquiring cost of data object k' transmitted from node c, as shown from steps 10 to 20 in Algorithm 1. For nodes b and d, as $C(k_1,a) + C(k_2,c) + 9 > C(k',c) + 5$ and $C(k_1,a) + C(k_2,c) + 7 > C(k',c) + 2.5$, it is costlier to generate k' locally. However, for node a, it is less expensive to generate k' at node a, since $C(k_1,a) + C(k_2,c) + 2 < C(k',c) + 2$ (where $C(k',a) = C(k',c)$).
2) At time $T_2 + 2\lambda$, node a finalizes the cost for k_2 and k' and broadcasts an ADV message. Node b updates the cost for k' to $C(k',a) + 4$ and resets the timer for k', as shown in Fig. 4(d) (from T_2 to $T_2 + 2\lambda$, nodes b and d finalize the cost for k_1, respectively, and as they have no effect on the cost kept by the nodes, they are omitted here).
3) At time $T_2 + 2.5\lambda$, node d finalizes the cost for k_2 and k' and broadcasts an ADV message. Node b updates the cost for k_2 to $C(k_2,c) + 4.5$ and resets the timer for k_2, as shown in Fig. 4(e).
4) Finally, node b finalizes the costs for k_2 and k' at times $T_2 + 4.5\lambda$ and $T_2 + 6\lambda$.

From the aforementioned discussion and as shown in Fig. 4, the nodes will get the minimum acquiring cost for each data object by one ADV message broadcast at each node for each data object (totally 12 broadcasts for the example in Fig. 4). For completeness, the pseudocode of MCFA is shown in Algorithm 2.

However, the minimum-cost forwarding algorithm cannot guarantee the two properties mentioned previously for a data object generated by the aggregator. We use Fig. 5 as an illustration. Assume that the transmission, propagation, and processing delays are negligible, the time period from the sending of ADV at the sender to the sending of ADV at the receiver for the same data object is proportional to the transmission cost between the sender and receiver. As shown in Fig. 5(a), the transmission costs are both 1050 from node a to c and from node b to c. Assume that node a and b send out an ADV message for k_1 and k_2 at time T; then, node c will broadcast an ADV message with $C(k', c) = 2100$ for generating k' by itself at time $T + 1050\lambda$. However, there may be a lower cost way to obtain data object k' at node c, as shown in Fig. 5(b). In Fig. 5(b), the acquiring cost for k' at node c is $C(k', c) = 2080$. However, node c receives an ADV message with this cost at time $T + 1070\lambda$ that is later than $T + 1050\lambda$. When node c receives an ADV message with the less cost $C(k', c) = 2080$ at $T + 1070\lambda$, it has already broadcast ADV with the cost $C(k', c) = 2100$ at $T + 1050\lambda$. Node c still needs to broadcast this ADV message because it has less cost. Thus, one broadcast for one data object cannot be guaranteed for a data object generated by the aggregator.

Algorithm 2: MCFA

```
  Event: Node $p$ receives ADV from node $q$ for data object $k$
1 begin
2     $C'(k,p) = C(k,q) + d_k C_T(p,q)$
3     if $C'(k,p) < C(k,p)$ then
4         replace $C(k,p)$ with $C'(k,p)$
5         $O(k,p) = 1$
6         $R(k,p) = q$
7         reset timer of $k$ to expire after $\lambda \cdot C_T(p,q)$
8         for each ancestor $k'$ of $k$; // bottom-up
9         do
10            if $\Delta_{k'}$ are acquirable then
11                $C'(k',p) = \sum_{m \in \Delta_{k'}} C(m,p) + C_c(k',p)$
12                if $C'(k',p) < C'(k',p)$ then
13                    replace $C(k',p)$ with $C'(k',p)$
14                    $O(k',p) = 1$
15                    $P(k',p) = 1$
16                    reset timer of $k$ to expire after $\lambda \cdot C_T(p,q)$
17                end
18            else
19                return
20            end
21        end
22    end
23 end
  Event: Node $p$'s timer for data object $k$ expires
24 begin
25    broadcast ADV containing $C(k,p)$
26 end
```

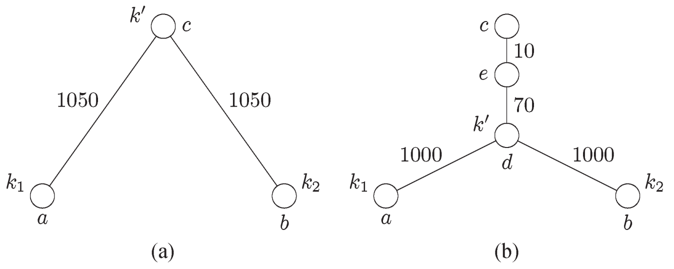


Fig. 5. Illustration of minimum-cost forwarding algorithm which cannot guarantee to get minimum acquiring cost for data object generated by aggregator by one broadcast.

It is easy to see that the nodes process messages one by one; then, the asynchronous algorithm converges after a finite number of message exchanges. Next, we will show that Algorithm 2 solves the optimization problem within time $\lambda C_{\max}(H_T + 2)$ and the number of messages sent per node is $MC_{\max}/C_{\arg}$, where M is the number of data objects of the service tree, $C_{\max}$ is the maximum transmission cost of the minimum-cost path, and $C_{\arg}$ is the average transmission cost between the node and its immediate neighboring nodes. Specially, $C_{\max} = \max_{pq} C_{pq}$, where C_{pq} denotes the transmission cost of the minimum-cost path from node p to node q, and $C_{\arg} = \arg_p C_p^{\text{neighbor}}$, where C_p^{neighbor} denotes the minimum transmission cost between node p and its neighboring nodes.

Theorem 1: Algorithm 2 converges to the optimal solution within time $\lambda C_{\max}(H_T + 2)$ (if the process will be started by flooding the service tree into the network by one node) or $\lambda C_{max}(H_T + 1)$ (if source nodes of the service tree broadcast ADV messages for the data object generated at the same time) and with the number of messages sent per node $O(MC_{\max}/C_{\arg})$.

Proof: Although the transmission, propagation, and processing delays are nonzero and these factors may alter ordered broadcasting along the optimal path, if we set λ large enough, the impact can be minimal. Thus, these delays are not considered here. Assuming that the sink node broadcasts the service tree using minimum-cost forwarding at time T, all the source nodes will broadcast an ADV message for each data object generated within $T + \lambda C_{\max}$ right after they receive the service tree. Then, within $T + 2\lambda C_{\max}$, all the network node will get the minimum acquiring costs for data objects generated by source nodes. Iteratively, within $T + 3\lambda C_{\max}$, all the network node will get the minimum acquiring costs for data objects generated by the next service level. Finally, all the network nodes will get the minimum acquiring cost for the final data object within $T + \lambda C_{\max}(H_T + 2)$. Thus, Algorithm 2 converges to the optimal solution within time $\lambda C_{\max}(H_T + 2)$ from one node broadcast service tree into the network. If source nodes broadcast an ADV message for the data object generated at the same time, Algorithm 2 converges within $\lambda C_{\max}(H_T + 1)$.

Assuming that source nodes broadcast an ADV message for the data object generated at the same time, during the first time slot of $\lambda C_{\max}$, the number of messages sent per node is M_1 (each node broadcasts an ADV message once for each source data object), where M_1 is the number of source data objects. For the second time slot of $\lambda C_{\max}$, the number of messages sent per node is $M_2 C_{\max}/C_{\arg}$. As $\lambda C_{\max}$

is the time period, λC_{arg} is the average time for the node to broadcast one ADV message, and M_2 is the number of data objects generated by the next service level, we expect the number of messages each node sent out during $\lambda C_{\max}$ to be $O(M_2 C_{\max}/C_{\mathrm{arg}})$. Similarly, for the third time slot of $\lambda C_{\max}$, the number of messages sent per node is to be $O(M_3 C_{\max}/C_{\mathrm{arg}})$. Thus, for the convergence of the optimal solution, the total messages sent out per node are $M_1 + (M_2 + M_3 + \cdots + M_{H_T+1})C_{\max}/C_{\mathrm{arg}}$. We expect it to be $O((M_1 + M_2 + M_3 + \cdots + M_{H_T})C_{\max}/C_{\mathrm{arg}}) = O(MC_{\max}/C_{\mathrm{arg}})$. $\square$

The Sync proposed in [16] converges to the optimal solution with $L_{\max}(H_T + 1)$ iteration, and the number of messages sent out per node is $O(ML_{\max}H_T)$. $L_{\max}$ is the number of hops of the longest minimum-cost path, and $L_{\max} = \max_{pq} L_{pq}$, where L_{pq} denotes the number of hops of the minimum-cost path from node p to node q.

If we use hop count as the cost metric, $C_{\max}$ will be equal to $L_{\max}$, and C_{arg} will be equal to one. MCFA will converge to the optimal solution within time $\lambda L_{\max}(H_T + 1)$ and with the number of messages sent per node $O(ML_{\max})$. Thus, MCFA has less messages sent out per node $O(ML_{\max})$ than the Sync $O(ML_{\max}H_T)$. If the service tree is a complete binary tree, $H_T = \log_2(M + 1) - 1$. Moreover, if the network is a regular grid, then $L_{\max} = \sqrt{2N}$, where N is the number of nodes in the network. We expect the number of messages sent out per node to be $O(\sqrt{N}M)$ for MCFA and $O(\sqrt{N}M \log_2 M)$ for the Sync.

VI. Performance Evaluation

The simulations were conducted by using QualNet. We used a simulation setting with 2000 network nodes scattered randomly in 1500×1500 m^2. For each node, the data rate at the physical layer is 256 kb/s, the transmission range is 50 m, and Carrier Sense Multiple Access (CSMA) is used as the Media Access Control (MAC) layer protocol. We use the complete binary tree as the service tree, and the sizes of the service tree (the number of aggregators) are 1, 3, 7, and 15. In simulation, we denote the transmission cost between nodes p and q as d_{pq}, where d_{pq} is the distance between nodes p and q and $d_{pq} \leq 50$ m.

In simulation, we first investigate how the λ value affects the ADV message overhead. Then, we investigate the distribution of redundant ADV messages. Finally, we compare the MCFA, Sync [16], and Async in both ADV message overhead and setup time for searching the optimal solution.

A. Impact of Timer Coefficient

If λ is not large enough, the accumulative processing, transmission, and propagation delay factors along a path could alter the ordered broadcasts of nodes. Then, a node may broadcast an ADV message more than once for data objects generated by the source node. Furthermore, as the size of the service tree increases, it causes heavier traffic in the network and results in an increase of transmission delay. We use the average ADV message overhead for each source data object in the service tree as metric to evaluate the timer coefficient.

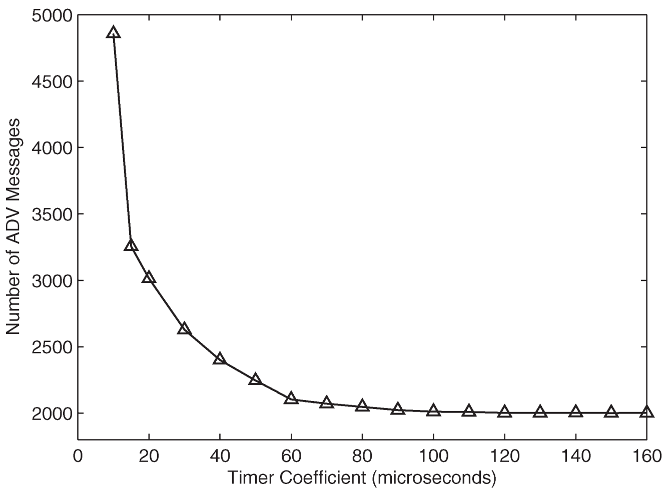


Fig. 6. ADV message broadcast for each source data object in service placement process according to timer coefficient λ. Service tree size is three.

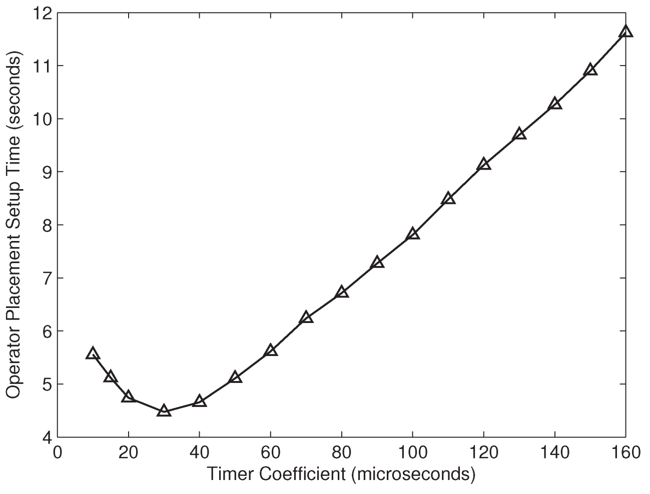


Fig. 7. Setup time of service placement according to timer coefficient λ. Service tree size is three.

In Fig. 6, λ varies from 10 to 160 μs with a step size of 10 μs. It can be seen that, when λ is 10 μs, the number of ADV broadcasts is more than 4500. This means that each network node broadcasts an ADV message twice on average for each source data object. As λ increases from 10 to 80 μs, the number of ADV messages drops dramatically. Then, the curve is relatively steady, and only a few nodes have more than one ADV broadcast at 80 μs. Further increase of λ almost eliminates multiple broadcasts for each source data object.

In Fig. 7, as λ increases from 10 to 30 μs, the setup time is decreased. When λ is small, it incurs large ADV message overhead for each data object of the service tree, as shown in Fig. 6. The key observation from simulation is that heavy traffic creates a network congestion (as the data rate of the physical layer is only 256 kb/s), which leads to a longer setup time period. The increase of λ results in less ADV overhead and reduces the network traffic, and network congestion disappears when λ is 30 μs. After that, the setup time is proportional to λ, and a larger λ leads to a linear increasing setup time.

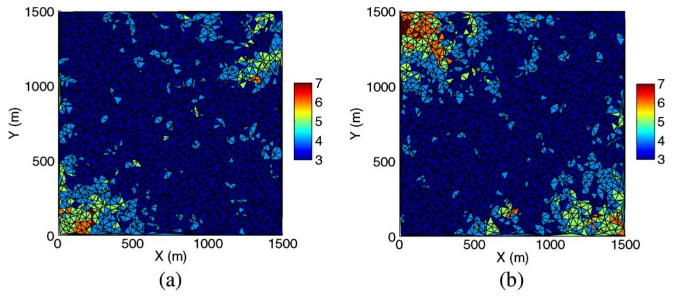


Fig. 8. Distribution of redundant ADV message at each node according to node location. The size of service tree is one. Two source nodes are respectively located at top-left corner and bottom-right corner in (a) and top-right and bottom-left corners in (b).

B. Distribution of ADV Redundant Messages

As discussed in Section V, MCFA cannot guarantee one ADV message broadcast for the data object generated by service. Now, we investigate the distribution of redundant ADV broadcast. In this case study, we use the tree with one aggregator. Thus, there are two source data objects and one final data object in the service tree. At least three ADV broadcasts are needed to get the optimal acquiring costs for all three data objects at each node. In simulation, we choose two nodes located at the diagonal corners of deployed field as source nodes. The distribution of ADV messages at each node is shown in Fig. 8. Most nodes have three ADV broadcasts for getting three optimal acquiring costs of the data object. However, few nodes have redundant broadcasts, i.e., seven ADV broadcasts at most (note that redundant ADV messages are broadcast for the final data object and all the nodes broadcast an ADV message once for getting the optimal acquiring cost for the senior data object), which are distributed near other two corners of deployed field, as shown in Fig. 8. This conforms to our analysis about ADV message redundancy for a data object generated by the aggregator in Section V.

C. Comparison Among Sync, Async, and MCFA

Now, we investigate the comparison among MCFA, Sync, and Async in ADV message overhead and setup time for the service tree. In this case study, we use the least λ, which can guarantee the optimality of service placement, and the minimized ADV message overhead, for MCFA. We also use the least time interval of ADV message transmission, which can guarantee the optimality of service placement, for Sync.

In Fig. 9, the ADV message overhead of all these three algorithms increases according to the increase of the size of the service tree. Among them, MCFA has the least ADV message overhead as we analyzed mathematically. As shown in Fig. 10, the setup time of all these three algorithms also increases with the increase of the service tree size. Async has the shortest setup time, and MCFA has the same setup time with Sync roughly. However, as discussed previously and observed in simulation, due to the network congestion incurred by heavy traffic, Async cannot guarantee the optimality of service tree placement, even though Async can terminate faster than the other two algorithms. Although Sync and MCFA have almost the same setup time, Sync requires the complete synchronization among all the network nodes, and all the nodes should know when other nodes finish information updating and when they finish sending out updated information. Hence, Sync is difficult and costly, due to time synchronization, to be implemented into real applications.

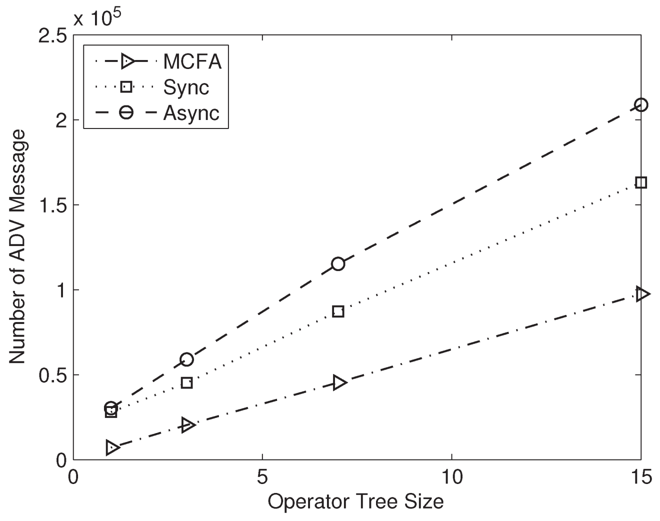


Fig. 9. ADV message broadcast in service placement process for each algorithm according to service tree size.

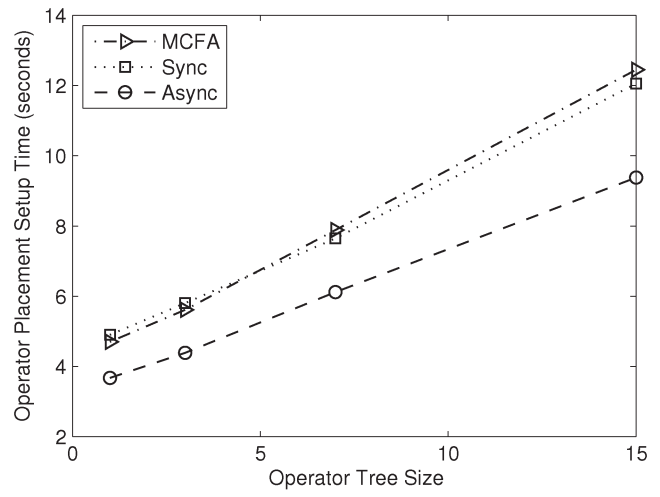


Fig. 10. Setup time of service placement for each algorithm according to service tree size.

VII. Conclusion

In smart grid communication networks, a large amount of data and information will be generated from metering, sensing, monitoring, etc. Data aggregation (integration or fusion) aims at the merging of data from disparate sources with differing conceptual, contextual, and typographical representations. In order to satisfy the requirement of different information services, the placement of data aggregation services in smart grid communication networks is a critical design issue. In this paper, we have considered the optimal placement of the service tree with optimal cost in transmission and computation for a general model. We have proposed Async first. However, it has a significant message overhead. Then, we have proposed a minimum-cost-forwarding-based asynchronous distributed algorithm. It is shown that minimum-cost forwarding can dramatically reduce message overheads of the asynchronous algorithm. We have shown that our algorithm has less message overheads than Sync by both mathematical analysis and simulation-based evaluation. For a regular grid network and a complete binary service tree, the messages sent at each node are $O(\sqrt{N}M)$ for our proposed algorithm and $O(\sqrt{N}M\log_2 M)$ for Sync, where N is the number of network nodes and M is the number of data objects in the service tree. We believe that our algorithm can be easily implemented into smart grid communication networks to satisfy the various requirements of information flow and services.

REFERENCES

[1] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid-the new and improved power grid: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 944–980, 2011.

[2] T. Chim, S. Yiu, L. Hui, and V. Li, "Pass: Privacy-preserving authentication scheme for smart grid network," in *Proc. IEEE Int. Conf. SmartGridComm*, 2011, pp. 196–201.

[3] M. Levorato and U. Mitra, "Optimal allocation of heterogeneous smart grid traffic to heterogeneous networks," in *Proc. IEEE Int. Conf. SmartGridComm*, 2011, pp. 132–137.

[4] Y. Zhang, R. Yu, M. Nekovee, Y. Liu, S. Xie, and S. Gjessing, "Cognitive machine-to-machine communications: Visions and potentials for the smart grid," *IEEE Netw.*, vol. 26, no. 3, pp. 6–13, May/Jun. 2012.

[5] D. He, C. Chen, J. Bu, S. Chan, Y. Zhang, and M. Guizani, "Secure service provision in smart grid communications," *IEEE Commun. Mag.*, vol. 50, no. 8, pp. 53–61, Aug. 2012.

[6] Y. Kim, V. Kolesnikov, H. Kim, and M. Thottan, "SSTP: A scalable and secure transport protocol for smart grid data collection," in *Proc. IEEE Int. Conf. SmartGridComm*, 2011, pp. 161–166.

[7] D. Li, Z. Aung, J. Williams, and A. Sanchez, "Efficient authentication scheme for data aggregation in smart grid with fault tolerance and fault diagnosis," in *Proc. IEEE Power Energy Soc. Conf. Innov. Smart Grid Technol.*, Washington, DC, USA, 2012, pp. 1–8.

[8] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst.*, 2005, pp. 250–258.

[9] N. Jain, R. Biswas, N. Nandiraju, and D. Agrawal, "Energy aware routing for spatio-temporal queries in sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2005, vol. 3, pp. 1860–1866.

[10] G. Chatzimilioudis, H. Hakkoymaz, N. Mamoulis, and D. Gunopulos, "Operator placement for snapshot multi-predicate queries in wireless sensor networks," in *Proc. 10th Int. Conf. MDM, Syst., Services Middleware*, 2009, pp. 21–30.

[11] A. Pathak and V. Prasanna, "Energy-efficient task mapping for data-driven sensor network macroprogramming," *IEEE Trans. Comput.*, vol. 59, no. 7, pp. 955–968, Jul. 2010.

[12] B. Bonfils and P. Bonnet, "Adaptive and decentralized operator placement for in-network query processing," in *Proc. 2nd Int. Conf. Inf. Process. Sensor Netw.*, 2003, pp. 47–62.

[13] K. Oikonomou, I. Stavrakakis, and A. Xydias, "Scalable service migration in general topologies," in *Proc. Int. Symp. WoWMoM Netw.*, 2008, pp. 1–6.

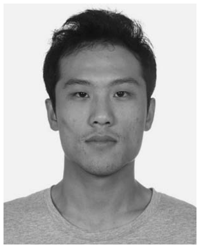
[14] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Proc. 22nd ICDE*, 2006, pp. 49–60.

[15] G. Chatzimilioudis, N. Mamoulis, and D. Gunopulos, "A distributed technique for dynamic operator placement in wireless sensor networks," in *Proc. 11th Int. Conf. Mobile Data Manage*, 2010, pp. 167–176.

[16] L. Ying, Z. Liu, D. Towsley, and C. Xia, "Distributed operator placement and data caching in large-scale sensor networks," in *Proc. IEEE 27th Conf. Comput. Commun. INFOCOM*, Apr. 2008, pp. 977–985.

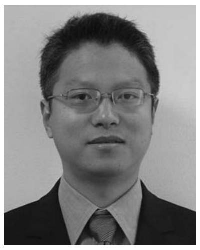
[17] Z. Abrams and J. Liu, "Greedy is good: On service tree placement for in-network stream processing," in *Proc. 26th IEEE ICDCS*, 2006, p. 72.

[18] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," in *Proc. 10th Int. Conf. Comput. Commun. Netw.*, 2001, pp. 304–309.



Zongqing Lu (S'12) received the B.S. and M.S. degrees from Southeast University, Nanjing, China. He is currently working toward the Ph.D. degree in the School of Computer Engineering, Nanyang Technological University, Singapore.

His research interests include wireless sensor networks, mobile ad hoc networks, social networks, delay-tolerant networks, mobile computing, and network privacy and security.



Yonggang Wen (S'99–M'08) was born in Nanchang, China, in 1977. He received the B.Eng. degree (with honor) in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 1999, the M.Phil. degree (with honor) in information engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 2001, and the Ph.D. degree in electrical engineering and computer science (with minor in Western literature) from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008.

He was a Senior Software Engineer for content networking products with Cisco. He was also a Research Intern with Bell Laboratories, Sycamore Networks, and Mitsubishi Electric Research Laboratory. He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. He has published more than 50 papers in top journals and prestigious conferences. His system research on cloud social television has been featured by international media (e.g., The Straits Times, The Business Times, Lianhe Zaobao, Channel NewsAsia, ZDNet, CNet, United Press International, Association for Computing Machinery (ACM) TechNews, The Times of India, and Yahoo News). His major field of study focuses on information and communication technologies (ICTs). His research interests include cloud computing, mobile computing, multimedia network, cyber security, and green ICT.

Dr. Wen is a member of Sigma Xi (the Scientific Research Society) and Society for Industrial and Applied Mathematics (SIAM).