Front. Comput. Sci., 2016, 10(5): 936–950
DOI 10.1007/s11704-016-4525-7

RESEARCH ARTICLE

Energy efficient approximate self-adaptive data collection in wireless sensor networks

Bin WANG[1], Xiaochun YANG (✉)[1], Guoren WANG[1], Ge YU[1], Wanyu ZANG[2], Meng YU[3]

1 School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
2 Department of Accounting, Computing and Finance, Texas A&M University at San Antonio, San Antonio TX 78363, USA
3 Computer Science Department, The University of Texas at San Antonio, San Antonio TX 78249, USA

Abstract To extend the lifetime of wireless sensor networks, reducing and balancing energy consumptions are main concerns in data collection due to the power constrains of the sensor nodes. Unfortunately, the existing data collection schemes mainly focus on energy saving but overlook balancing the energy consumption of the sensor nodes. In addition, most of them assume that each sensor has a global knowledge about the network topology. However, in many real applications, such a global knowledge is not desired due to the dynamic features of the wireless sensor network. In this paper, we propose an approximate self-adaptive data collection technique (ASA), to approximately collect data in a distributed wireless sensor network. ASA investigates the spatial correlations between sensors to provide an energy-efficient and balanced route to the sink, while each sensor does not know any global knowledge on the network. We also show that ASA is robust to failures. Our experimental results demonstrate that ASA can provide significant communication (and hence energy) savings and equal energy consumption of the sensor nodes.

Keywords wireless sensor networks, data collection, energy efficient, self-adaptive

Received November 18, 2014; accepted January 15, 2016

E-mail: yangxc@mail.neu.edu.cn

1 Introduction

One of the most important applications in wireless sensor networks (WSNs) is data collection, where the sensor nodes collect the sensing data and then forward them to the sink [1, 2]. Sensor nodes have limited supply of power, and thus such operation may cause energy-hungry and further shorten the *lifetime* of sensor network since wireless transmission costs more energy compared with other operations, i.e., computing [3]. There are different definitions on the network lifetime, and here we use the lifetime definition of Ref. [4], in which it is defined as the duration from initial start of the network to the moment when a certain percent of sensors is disconnected from the sink.

To extend the network lifetime, reducing while balancing the power consumption among all sensor nodes are great challenges in designing data collection schemes. Unfortunately, the existing research on data collection mainly focuses on low energy consumption and overlooks the importance of balancing the power consumption of the sensor nodes. Besides, most of the existing data collection approaches [5] assume that each sensor in the network has the global knowledge on the network topology, which is not feasible due to the dynamic features of WSNs.

Furthermore, an effective data collection method needs to consider the following technical challenges as well. 1) How to find an efficient routing (forwarding) path to the sink for

a sensor without the global knowledge on the network topology? 2) Sensing data often has spatial correlations, that is, sensors in a certain region read the same or quiet similar data in each time step. How to utilize this unique feature in the routing path search? 3) Sensing data also has temporal correlations, i.e., sensors may read quiet similar value in consecutive time steps. Ideally, we hope to find a "minimal" forwarding solution to convey sensing data to the sink. Here, "minimal" has two meanings: a shortest data routing path to transmit all clusters readings, and a path that forwards least updates to the sink. The difficulty is that the forwarding solution cannot be determined or fixed in advance, since values and clusters may change for different time steps. 4) How to resilient to the failure caused by the sensor nodes and data transmission?

To address above mentioned challenges, we propose an *approximate self-adaptive* (ASA) method, which can approximately gather the sensing data and prolong the network lifetime without any global knowledge of the network. That is, given a fixed sink and an arbitrary distributed sensor network, by using query "SELECT * FREQ f WITHIN $\pm\epsilon$", ASA makes the $\rho(\%)$ sensors connected to the sink as long as possible. In order to do it, ASA first divides the WSNs into multiple disjointed clusters based on the spatial correlations between the sensors. It then searches a "minimal" routing path to the sink by transmitting the cluster readings instead of the sensor readings. To balance the energy consumption of the sensor nodes, the sensing data will be forwarded to the sink through different nodes periodically. The main contributions of this paper are as follows.

- We propose a novel ASA approach to cluster sensors into different groups according to their sensing values without global knowledge of network topology in Section 3. If a set of sensors detect similar values with $\pm\epsilon$, then ASA classifies them into one cluster, which we call a *value cluster* (VC). Different from the existing clustering approaches, ASA uses at most four messages to construct small clusters when initializing the network.
- We design an effective map-based forwarding technique, which guides sensors to self-adaptively find a "minimal" routing path to the sink. Furthermore, ASA only forwards updates to the sink and changes routing path periodically to balance the energy consumption of the sensor nodes. We also propose a compact routing map to compress the navigation message along the routing path, and discuss how to adjust the compact routing map according to the remaining message size in a forwarding package in Section 4.
- For each time step, our approach expands and maintains its clusters gradually through its routing nodes (see Section 5). This clustering strategy is more practical for sensor applications, since ASA needs a few message communications for each time step and get optimal clusters in convergent fashion.
- We describe different failures occurred in the data collection, and analyze why ASA is resilient to the failures in Section 6.
- Last but not the least, we evaluate ASA and demonstrate its advantages on energy-balanced and energy-efficient data collection through extensive experiments on synthetic data sets.

2 Data collection problem

Sensor network A sensor network consists of a set of fixed-location sensors, each of which has a unique ID. We use an $m \times n$ grid to describe sensor locations. For each cell in the grid, it contains at most one sensor. Without loss of generality, we assume that the sink has unlimited energy, which locates in one edge of the grid and knows all sensors' ID and locations, and is responsible for performing computation. Two sensors are said to be neighbors if they are within the transmission range of each other.

For the data collection task, the sink initially broadcasts the locations of all sensors and a query "SELECT * FREQ f WITHIN $\pm\epsilon$" to the network. Different from the existing work, we focus on dynamic features of WSNs that only the sink knows all sensor locations [6] and each sensor maintains a location map about the sensor location but does not know the topology of the network.

The primary consumption of energy in sensor nodes is for radio communication, whereas computation cost is relatively much smaller than communication cost [7]. That is, the total amount of energy spent in sending a message with x bytes of content is given by $\sigma_s + \delta_s x$, where σ_s and δ_s represent the per-message and per-byte sending costs, respectively. Therefore, we try to maximize the lifetime of a network by minimizing the following factors, which are 1) the number of forward messages, 2) the size of messages sent through the network, and 3) the ratio of the used inner layer sensors to the used outer layer sensors (to handle the case that inner layer sensors "die" first).

Problem definition Given a sensor network that continuously senses values at each time step, and a sink that requires

an ϵ-loss approximation of the sensing data at all times, design a data collection protocol that makes $\rho\%$ sensors connected to the sink as long as possible.

In our proposed approach, there are three interlaced steps to accomplish approximate data collection at each time step: clustering sensors, forwarding the cluster values, and maintaining clusters dynamically along the time.

3 Value-based self-clustering model and initialization

As illustrated in Section 1, spatially neighbored sensing data are generally quite similar, which inspires us to cluster sensors into disjoint regions/clusters based on the sensing data value. Therefore, there is no need to broadcast every sensing data to the sink node and only a few sensors are chosen as representatives for each cluster to report the sensing values. Different from the existing approaches, we focus on dynamic feature of a WSN, where clusters can be constructed without any global knowledge. In this section, we devise energy efficient techniques with which sensors can automatically construct disjoint clusters using up to four communication messages.

Figure 1(a) shows the snapshot of network communication among ten sensors and their sensing values at time step t. For clear presentation, no grid lines are drawn in the figure. Note that, grid does not describle the communication topology, but describes location of sensors. For example, two sensors within communication distance does not mean that they are located in adjacent cells, and vice versa, since there may exist an obstacle preventing them from communicating with each other. Each node in the figure represents a sensor. Edges between nodes represent neighbors, i.e., one hop communication distance. For example, sensors s_1 and s_2 are neighbors. The inner layer sensors s_4, s_5, and s_9 are neighbors to the sink. Figure 1(b) shows an example of clustering result when $\epsilon = 0.3$. We use shade colors to describe clusters. Notice that, there are many different ways to divide the sensor network into disjoint value clusters (DVCs). Ideally, a network is desired to be partitioned into fewer clusters, so that fewer sensors are employed to sense and transmit data.

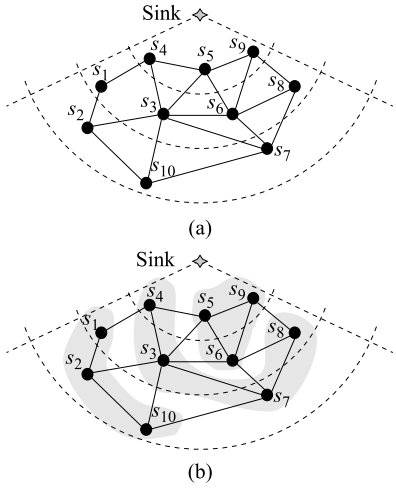


Fig. 1 Disjoint value clusters when $\epsilon = 0.3$ (Sensors and their values at time t are $\langle s_1, 25.0\rangle$, $\langle s_2, 24.9\rangle$, $\langle s_3, 24.8\rangle$, $\langle s_4, 24.7\rangle$, $\langle s_5, 25.1\rangle$, $\langle s_6, 25.0\rangle$, $\langle s_7, 24.5\rangle$, $\langle s_8, 24.7\rangle$, $\langle s_9, 25.1\rangle$, $\langle s_{10}, 25.2\rangle$). (a) Snapshot of sensing values; (b) three disjoint VCs

The process of constructing clusters starts from an initial wireless sensor network, where the sink broadcasts a start message to all sensors to construct DVCs. We define VCs as follows.

Definition 1 (VC) A set of sensors $S = \{s_1, s_2, \ldots, s_m\}$ belong to a value cluster C at time step t, if and only if C is connected, and for any two sensors $s_i, s_j \in S$, $|v_t(s_i) - v_t(s_j)| \leqslant \epsilon$, where $v_t(s)$ denotes sensing value of s at time step t.

Definition 2 (DVC model) A VC model is denoted as $C_v = C_1, C_2, \ldots, C_k$, where k is the number of VCs, and for any two C_i and C_j, they are disjoint.

In order to let sensors automatically construct VCs, in our framework, some sensors are chosen to actively invite their neighbors to join in their cluster. We call such chosen sensors *seeds*. Seeds are desired to distribute evenly in the network, so that all sensors can be involved in the clustering procedure simultaneously. That is, for any sensor s in the network, either s is a seed, or s is a neighbor to a seed. The choice of seeds is critical to the performance of the DVC model. For example, in Fig. 1(a), sensors s_2, s_4, and s_6 are good choices as seeds, since any other sensor in the network is a neighbor of one of them.

If the sink is aware of the communication topology of the network, then choosing seeds is a typical problem of constructing a dominating set over the topology graph. We can use a greedy approach to choose seeds. That is, we 1) start with the network and an empty set S; 2) pick a sensor with most neighbors which is not already in the set S and add it to S; 3) repeat step 2 until all sensors are in S or adjacent to the sensors in S. This greedy approach can guarantee $O(n)$ complexity, where n is the number of sensors in the network. However, as discussed in Section 1, in a dynamic wireless sensor network, it is hard for the sink to know the topology of the network, i.e., globally determining seeds is not practi-

cal.

Therefore, we devise a self-clustering approach to tackle this problem. Each sensor in the network is equipped with a random number generator with range $[0, 1]$. When the sink notifies that the network begins to work, each sensor generates a random value. If a sensor whose generated value is greater than 0.5, then it is chosen as a seed. Random value generator can guarantee a set of evenly distributed seeds in the network. There may exist special cases, e.g., sensors in a small region are all chosen as seeds or no seed exists in a certain region. After receiving the start message sent from the sink, each seed broadcasts its sensor reading and receives neighbors responses.

The detailed techniques are covered in Algorithms 1 and 2. The algorithm SeedDVC(s, t, t_I) shown in Algorithm 1 illustrates how a seed constructs a set of DVCs. Given a tolerant time interval t_I, if seed s gets responses within t_I, s will construct its value cluster $VC(s)$ (lines 4–7). Otherwise in the case that s cannot get any response from its neighbors within t_I, if it gets invitation message from another seed s', then s sends response message to s' and waits for the response from s' to see if it can join in the cluster of s' (lines 10–12). If s does not get any response or invitation, it will find a neighor s'' to join in (lines 13–15).

Algorithm 1 SeedDVC(S, t, t_I)

Input: A set of seeds S, time step t, tolerant time interval t_I

```
1  ;
2  VC(s) = s;
3  Each seed s in S broadcasts its value v_t(s) to invite its neighbors to
   join in its region;
4  Seed s gets messages from its neighbors s'_1, s'_2, ..., s'_h within t_I;
5  if there is at least one response message then
6      for each responsed neighor s'_i do
7          if |v_t(s) − v_t(s'_i)| ⩽ ϵ then
8              VC(s) = VC(s) ∪ s'_i;
9      Let V_I(s) = [v_l, v_u], where v_l = min_{s'∈VC(s)}{v_t(s), v_t(s'_i)}, and v_u =
       max_{s'∈VC(s)}{v_t(s), v_t(s'_i)};
10     s notifies V_I(s) and its member IDs to neighors in VC(s);
11 else if an invite message is sent from another seed s' then
12     s changes itself to a non-seed sensor;
13     s sends response to s' according to Algorithm 2;
14 else if no neighbors send messages within t_I then
15     s changes itself to a non-seed sensor;
16     s chooses a seed s'' to join in, such that s has closest value distance
       to s'';
```

Algorithm 2 describes a non-seed behaviors by using at most four messages. For example, in Fig. 1(a), after using Algorithms 1 and 2, ASA builds up four DVCs, which are $C_1 = \{s_1, s_2, s_{10}\}$, $C_2 = \{s_3, s_4\}$, $C_3 = \{s_7, s_8\}$, and $C_4 = \{s_5, s_6, s_9\}$. If s_2, s_4, and s_6 are chosen as seeds, ASA only needs two messages to build up these VCs; however, if s_1, s_2, and s_{10} are chosen as seeds, then ASA needs four messages to achieve the four VCs.

Algorithm 2 NonSeedDVC(s', t_0, t_I)

Input: A set of non-seeds S', starting time step t_0, tolerant time interval t_I;

```
1  Each s' ∈ S' monitors messages from its neighbors;
2  if there is no messages from its neighbors at time step t ⩽ t_0 + t_I
   then
3      s' changes itself to a seed;
4      s' uses Algorithm 1 to construct its VC VC(s');
5  else if s' receives invitations from its neighbors S at time step
   t ⩽ t_0 + t_I then
6      Choose a seed s ∈ S such that for any seed s'' ∈ S, |v_t(s')−v_t(s)| ⩽
       |v_t(s') − v_t(s'')| and |v_t(s') − v_t(s)| ⩽ ϵ;
7      if there exists such a seed s' ∈ S then
8          s' sends v_t(s') to s;
9          if s' receives message V_I(s) from s at a time slot ⩽ t_0 + 2t_I then
10             s' stores V_I(s);
11         s' broadcasts v(s') to its other neighbor seeds;
12         NonSeedDVC(s', t, t_I);
```

4 Self-adaptive data routing

Once DVCs are constructed, every sensor in the WSN knows the sensing value range $V_I(s)$ in its cluster. So any sensor can play as a passer to route data to the sink.

Recall that sensors in the network are unware of the topology of the network. So a crucial problem that we need to address is how to smartly find a "minimal" route to send sensing values to the sink by concatenating all DVCs.

Definition 3 (Valid data route) Given a set of DVCs $C_1, C_2, \ldots, C_k$ and sink s, a route $s_1 \to s_2 \to \cdots \to s_m \to s$ is valid, if and only if for each VC C_i, there exists at least one routing sensor $s_j \in C_i$ $(1 \leqslant j \leqslant m)$.

Based on above definition, the problem of *minimal valid data route analysis* can be represented as: given a set of disjoint value clusters $C_1, C_2, \ldots, C_k$, find a valid route $s_1 \to s_2 \to \cdots \to s_m \to s$ to forward values of all clusters to the sink s. The valid route is minimal if 1) s_1 is disconnected to s after removing s_j $(1 \leqslant j \leqslant m)$ from the route, and 2) only updates are transmitted to the sink in the remaining lifetime of the WSN.

In Section 1 we propose a novel data structure, called *routing map* to guide sensors to concatenate the constructed DVCs. In Section 2 we propose a compact routing map to

further reduce the transformation cost. In Section 3 we devise our techniques to balance energy consumption based on routing map.

4.1 Routing map

Recall sensors in the WSN have no global knowledge about the topology of the whole network. So it is a big challenge to ask sensors to self-adaptively construct a data route to the sink. A feasible way is to configure a very small routing map for each sensor, so that sensors can be guided to concatenate all DVCs in the network.

Figure 2(a) shows a sketch of a routing map. The routing map is an $m \times n$ matrix corresponding to the network grid, and each cell in the matrix is labelled using 0 or 1. If a VC has been routed, then all cells covered by the VC are marked, i.e., labelled 1. All un-routed cells are unmarked, i.e., labelled 0. Using a routing map, a sensor can know which regions need to be routed.

In the initialization phase, the sink broadcasts sensors' locations, i.e., the grid to the network. For each time step t, sink randomly selects a sensor s to start routing. s first marks 1 on sensors in its VC, and then it finds a shortest path to reach the areas that have not been marked, and have long distances to the sink if there are more than one shortest paths. The requirement 1) guarantees that the route crosses non-routed VCs, and 2) lets ASA heuristically route sensors with more hops to the sink in priority, which helps balance energy consumption of the sensors.

Besides appending $v_t(s)$ in the forwarding message, in order to navigate sensors to find a valid route, the last routing sensor adds its routing map to the forward message. When a routing sensor s' receives the forward message, it determines routing according to the attached routing map.

4.2 Compact routing map

Ideally we hope a routing map as small as possible so that it will not cause too much transmission cost. In this section we propose a *compact routing map* to compress the routing map.

A compact routing map consists of a compact matrix and a set of key points stored in an array A, such that the set of key points in A construct a contour line to distinguish routed VCs from non-routed VCs. Figure 2(b) shows three compact routing maps $\mathrm{CRM}^1_{8\times16}$, $\mathrm{CRM}^2_{8\times16}$, and $\mathrm{CRM}^3_{8\times16}$, against the full routing map $\mathrm{RM}_{8\times16}$ shown in Fig. 2(a). A compact routing map can guide sensors to find its routing according to following features.

- The left bottom region to the contour line marked by the key points in A is marked, i.e., 1, which expresses that VCs covered by the left bottom region have been routed.
- The right top region to the boundary including key points in A is unmarked, i.e., 0, which means that the VCs covered by the right top region need to be routed.
- The region in the compact matrix M is partially marked, i.e., part of 1 and part of 0.

The compact routing map contains the same information with the whole routing map. For example, the first compact routing map $\mathrm{CRM}^1_{8\times16} = A_1 + M_{2\times6}(\langle 5, 6\rangle)$ contains two parts, which are the array A_1 of key points and a small matrix $M_{2\times6}(\langle 5, 6\rangle)$. The four key points in A_1 separate the matrix of 8×16 cells into marked and unmarked parts. The matrix $M_{2\times6}(\langle 5, 6\rangle)$ contains marked and unmarked routing cells. Different compact routing maps might require different sizes of storage. For example, $\mathrm{CRM}^1_{8\times16}$ requires eight integers and a (2×6) bitmap, $\mathrm{CRM}^2_{8\times16}$ requires 6 integers and a (3×6) bitmap, and $\mathrm{CRM}^3_{8\times16}$ requires four integers and a (4×8) bitmap.

Ideally, we try to make the compact routing map as small as possible, that is, we use fewer key points and a small size matrix to describe the routing status. Equation (1) estimates the size of a compact routing map, where, α is 1 if we need a compact matrix, and otherwise, it is 0. $|M|$ is the size of the compact matrix, and n is the number of key points. Each key point costs 32 bits (i.e., 4 bytes) message (i.e., values in x-axis and y-axis cost 2 bytes, respectively).

$$size(CRM) = \alpha|M| + 32n. \tag{1}$$

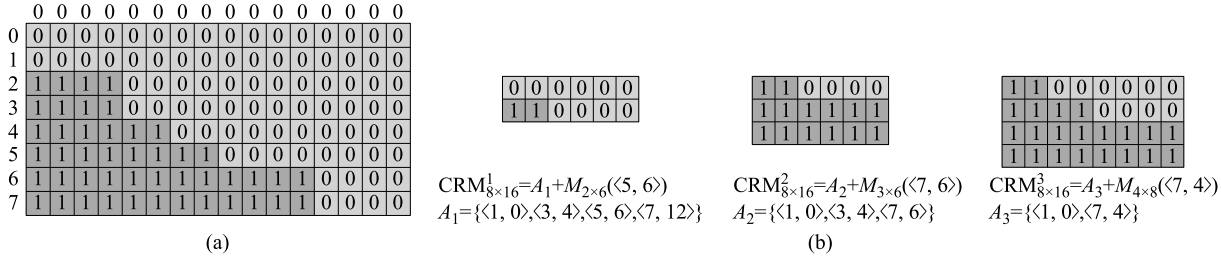


Fig. 2 Examples of different routing maps. (a) A full routing map $\mathrm{RM}_{8\times16}$; (b) three compact routing maps

Figure 2 shows three routing maps with compact information. For example, $size(\mathrm{CRM}^1_{8\times16}) = 140$, $size(\mathrm{CRM}^2_{8\times16}) = 114$, and $size(\mathrm{CRM}^3_{8\times16}) = 96$.

Minimal compact routing map problem Given a routing maps with n key points, find the compact routing map whose size is minimized.

Theorem 1 The problem of finding the minimal compact routing map is NP-hard.

This problem can be reduced to the classic set cover problem. For limit space reason, please ignore proof. The optimal solution that finds the minimal compact routing map can be approximated by a greedy algorithm. The greedy algorithm first selects a key point such that removing this point can construct a smallest compact matrix. This selection and deletion operation are repeated until no smaller compact matrix can be found. The pseudo-code of the greedy algorithm is shown in Algorithm 3.

Algorithm 3 Greedy algorithm

Input: A routing map and its n key points;
Output: An approximate minimal compact routing map

```
Let CRM_greedy.point = A and CRM_greedy.matrix = ∅;
for each key point i in A do
    get the corresponding matrix M;
    if size(CRM^i) < CRM_greedy then
        CRM_greedy.point = A − {i};
        CRM_greedy.matrix = M;
return CRM_greedy;
```

Algorithm 3 is an $\mathcal{H}(n)$-approximation algorithm, where $\mathcal{H}(n)$ is bounded by $1 + \log n$.

Relax compact routing map In addition to compressing routing map, we further cut the size of a compact routing map adaptively according to packet size constraint in a sensor network. The packet size in a sensor network has a limit L that is determined by the hardware or network protocols [8]. For example, the size limit of a packet is about 49 bytes on the Crossbow MICA2 motes [9]. Consequently, ASA adjusts the compact routing map under the permit transmission limit.

For example, if a transmitted message needs $L - k$ bytes, only k bytes are left for transmitting a compact routing map. ASA transmits the compact matrix without key points if k is too small for any minimal compact routing map. In this way, a sensor who receives this compact matrix can still find regions that has not been routed although it cannot know all un-routed regions although the guide information is limited. A good case is if a transmitted message occupies $L+1$ bytes, a sensor s has to use two packets to do transmission, although the second packet contains only 1 byte. In this case, s can transmit a compact routing map up to $L-1$ size to maximally utilize message packet. In Section 6, we show such a relax approach results in more energy-efficient routing.

4.3 Balance of energy consumption

In order to balance the energy consumption of sensors, ASA builds up different routes periodically. During one period of time T, the route is expected to be similar so that only the updates need to be forwarded. We first describe how to forward messages when a new route is built up. Then, we discuss forwarding updates to the sink along the constructed route. Rules R_1 and R_2 describe how to forward messages along the route at the time when a new route is built up.

- R_1 For each routing sensor, it appends one sensing value and the member sensors ID of its VC to the forwarding message.
- R_2 If there are more than one sensors in a VC which has been routed, then only the last routing sensor needs to append message.

For different period of routing time, sensors find a different route to balance the energy consumption in the network. Each sensor in the network keeps its routing history and is aware of its remaining energy. Also, ASA utilizes Gossip protocol [10] to intelligently find the new valid route. For instance, at the first step of a new time period T, a new routing sensor s_a is chosen and then s_a selects a sensor s_b as its downstream routing node according to its routing history and (compact) routing map. It will choose a sensor from its neighbors such that the chosen downstream routing node never or does not route data much. Sensor s_b checks its routing history and remaining energy. In case that s_b serves a routing sensor in recently time period or has no enough energy to forward messages, it will not forward messages further. After s_a sends message to s_b, it can hear messages sent from s_b although those messages are not sent to s_a according to Gossip protocol. If s_a cannot hear message from s_b, then s_a knows that s_b does not forward its message. So, s_a chooses another sensor s'_b. In this way, a sensor can self-determine its role of routing according to its own status. There is one exception that s_a fails to monitor the broadcast message from s_b, and then s_a chooses another routing sensor and meanwhile s_b continues to forward messages. Therefore, at least two routes will arrive at the sink.

Notice that, the selection of new route sensor is guided by the routing history and (compact) routing map, which can avoid a long route as well as too much energy consumption.

5 Route-based VCs expansion and maintenance

Ideally, a larger size of VCs is expected to be found according to the sensing values, so that ASA uses fewer routing sensors to forward messages. In addition, maintaining a large VC along the time is a critical problem. In this section, we describe VCs expansion and maintenance based on the valid route. We do not consider failure in this section. We discuss the failure processing in Section 6.

5.1 VCs expansion

Reexamine Fig. 1(a), the optimal DVCs under $\epsilon = 0.3$ are shown in Fig. 1(b), where each shaded area expresses a VC. Instead of spending huge number of messages to get the optimal DVCs at one time step, ASA gradually expands VCs to an optimal solution.

The main idea of expanding VCs is to utilize forwarding values along data routing to concatenate VCs $C_1, C_2, \ldots, C_h$ into a larger VC C, where minimizes the value interval of C:

$$\text{minimize} \quad V_I(C) = [v_l(C), v_u(C)]$$

$$\text{subject to:} \quad |v_u(C) - v_l(C)| \leqslant \epsilon,$$

where $v_l(C) = \min_i\{v_l(C_i)\}, v_u(C) = \max_i\{v_u(C_i)\}$.

Now we discuss how to expand VCs based on a valid data route. As defined in Definition 3, each VC contains at least one routing sensor in a valid data route. Assume s_1 and s_2 are adjacent routing sensors ($s_1 \to s_2$) that belong to VCs C_1 and C_2, respectively. Sensor s_2 receives the value of s_1 and checks if the distance between $v_t(s_1)$ and the value interval of cluster C_2 is less than ϵ (see the distance function in Eq. (2)). If so, s_2 in C_2 sends an invitation message containing sensor IDs and its value interval to s_1. Sensor s_1 checks whether the two VCs satisfy merge condition ($|\max\{v_u(C_1), v_u(C_2)\} - \min\{v_l(C_1), v_l(C_2)\}| \leqslant \epsilon$). If so, it broadcasts merge message among sensors in the two VCs. In this way, sensors in the two VCs know the new value interval and all sensor IDs of the merged VC.

$$dist(v, V_I(C)) = \begin{cases} v_u(C) - v, & \text{if } v < v_l(C); \\ v - v_l(C), & \text{if } v > v_u(C); \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

For example, in Fig. 1, there are four constructed VCs when initializing the network. Suppose the data route is $s_{10} \to s_7 \to s_3 \to s_5 \to sink$, then s_3 knows $v_t(s_7)$ and sensors IDs what s_7 represents. The distance between $v_t(s_7)$ and the value interval $V_I(s_3)$ is 0.3 ($\leqslant \epsilon$), and therefore, s_3 sends merge invitation to s_7. They satisfy the merge condition and can be merged to a big one (see Fig. 1(b)).

Lazy member refresh in a VC In the above discussion, when two VCs are merged into one VC, the routing sensor broadcasts sensor IDs of one VC to another VC. If the merged two VCs are large, then broadcasting all sensor IDs also cost large energy. In order to save more energy cost, ASA adopts a lazy member refresh technique. That is, when the routing sensor s_1 in the upstream VC C_1 receives the invitation of its downstream VC C_2, and finds the two VCs satisfy the merge condition, it does not broadcast the new sensors IDs until it becomes a non-routing sensor.

For a set of VCs that satisfy the merge condition, obviously, the merge order among these VCs depends on the adopted data route. However, using our routing strategy, VCs expansion is convergent since data route changes every period of time to achieve an energy-balanced network.

5.2 Dynamic VCs maintenance

Sensing values in a WSN may change frequently. Therefore, the self-adaptive data collection based on clusters needs to be maintained dynamically.

One of the advantage of our approach is that routing sensors also play roles for monitoring updates in their own VCs. In order to monitor updated sensing values using few transformed messages, when a valid routing path is built up, each routing sensor broadcasts messages in its own VC and builds up a steiner tree using tiny agregation (TAG) technique [8]. Figure 3(a) shows the steiner trees rooted at routing sensors s and p, respectively. These steiner trees are used to report updates of sensor readings.

Notice that, in order to save network communication, small update will not report to the sink. Each cluster C has a value interval $V_{I_{t-1}}(C)$ at time step $t-1$, therefore, a sensor reports its update at time step t only when its sensing value v_t does not belong to $V_{I_{t-1}}(C)$, i.e., $v_t(s) \notin V_{I_{t-1}}(C)$, and then sensor s informs $v_t(s)$ to its corresponding routing sensor. The neighbors of s can hear such information. Generally, neighbor sensors detect a similar value to $v_t(s)$. If the neighbors of s cannot detect such updates, then we assume that $v_t(s)$ is a noise (or outlier). If there is at least one neighbor s' detects update and $|v_t(s) - v_t(s')| \leqslant \epsilon$, then s uses Algorithm 1 to build up a new VC.

Figures 3(a) and 3(b) show two update cases, respectively:

1) a new VC appears inside an old VC (e.g., regions II and

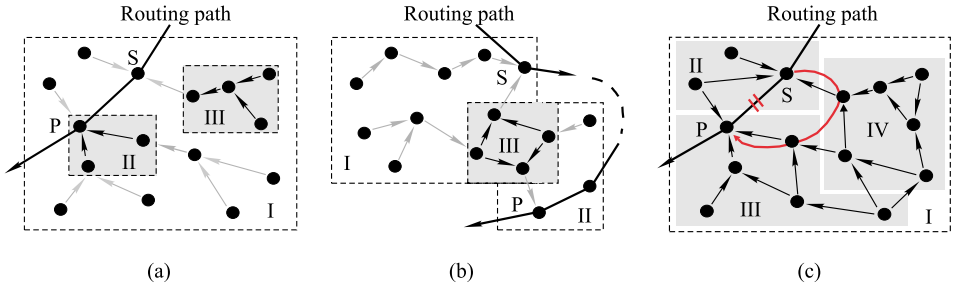


Fig. 3 VCs maintenance. (a) Inside update; (b) boundary update; (c) routing adjustment

III in Fig. 3(a) are new VCs), and

2) a new VC appears in the boundary of more than one old VCs (e.g., region III is a new VC that locates in the boundary of regions I and II in Fig. 3(b)).

For case 1, an update message including sensing value and the related sensor IDs is reported to its corresponding routing node along a steiner tree. Routing nodes broadcast update IDs in the old VC. Note that, any new VC only involve one hop neighbor sensor, therefore, the number of update sensors is small, which results in small size of broadcast message in the old VC. For case 2, sensors located in region III report updates to their own rooted routing sensor, e.g., sensors S and P, respectively. Then all sensors in the network are aware of the updates.

After VCs update or merge, ASA adjusts the old data route to make itself valid for the new VCs. If update happens during a routing time of period, then ASA only incrementally updates route between the old VC and the new VC.

We discuss cases where there is only one update VC in the network. Without loss of generality, it can be extended to the case of multiple update VCs. For the update case 1, an update VC is inside an old VC. When a routing node is involved in a new VC (e.g., P in region II in Fig. 3(a)), then ASA keeps using the old data route to forward update value through P. When a new VC does not contain a routing node (e.g., region III in Fig. 3(a)), then S forwards a compact routing map to a root node in region III along the steiner tree and aims at P as the routing target node. Furthermore, S notifies P a "cancel" message to break the route path $S \rightarrow P$.

For the update case 2, an update VC is located in the boundary of more than one VCs (see Fig. 3(b)). In order to avoid S and P duplicately adjusting their downstream route, the new VC should specify a routing node. Since the new update VC is little using our strategy, sensors inside the update VC can exchange message to choose a routing node with a shortest path to the update VC. Similar to the solution to case 1, a chosen routing sensor (e.g., S) sends its compact routing map to the update VC and lets it self-find a route to the downstream route sensor of S. The old routing path is broken after the adjustment.

Figure 3(c) shows a special case where sensors in an old VC all detect updates, and three new small VCs are built up. ASA uses the same routing approach to add the red path in between S and P and break the old path $S \rightarrow P$. It is interesting to find that if the three VCs have the same incremental updates, then ideally, the old VC should not be broken into three parts. Using ASA, each sensor builds up a new small VC, and then using an update routing path, we can concatenate and merge these three VCs quickly after forwarding updates.

6 Failure-resilient data collection

The problem of high failure rates for the processes of both sensing and forwarding is a major concern in wireless sensor networks. Zhang et al. [11] presented an innovative scheduling algorithm, a fault-tolerant scheduling for data collection. The main contribution of this work is that it observes that time is a factor of improving fault-tolerance. It divides time into discrete time slots and adjusts the pre-schedule and adaptive schedule to improve the fault-tolerance performance. Notice that, this approach is based on two implicit assumptions.

1) It knows exactly when the node failure could happen.
2) There exists a sink or node that stores a copy of the sensing value of the failure node.

Different from this work, ASA enhances fault-tolerance ability according to the correlation among sensing values. According to our observations over real experiments, there are three major failures in data collection: 1) *node failure*: a sensor is power off; 2) *sensing failure*: a sensor may detect a wrong value if its remaining energy is below than a critical level; and 3) *forwarding failure*: a message gets lost when it is forwarded along a certain routing path.

Node failure Our approach is robust to node failure. In our DVC model, a sensor forwards its message only after it detects an update reading. When a sensor s is power off, it acts as no update detected. In this case, if the neighbors of s do not report any updates, then it assumes that readings in the region of s do not change. If the neighbors of s report updates, s is either inside or on the boundary of the changed region. For the first case, a new VC covering s is constructed, since s is the only sensor that has a different value (i.e., keeping the old value) from those of its neighbors. As discussed in Section 5, in this case, s is regarded as an outlier. For the second case, a sensor on the boundary of the VC does not affect the sensing value much. When s is chosen to forward message, the previous sensor s_a cannot hear any message from s using Gossip protocol, then s_a will choose another sensor to forward its message as discussed in Section 1 if sensors within an area fail, no sensing value can be collected in this area. In this case, our approach still can collect and forward other sensing values to the sink.

Sensing failure Sensing failure is a serious problem. However, to our best knowledge, it has not been discussed by any previous work. When a sensor encounters a sensing failure, it still acts as a normal sensor to detect and forward messages. In this case, it has enough energy to correctly forward incorrect sensing values to other nodes. If a single sensing failure lasts for a few time, like a node failure, it can be detected as an outlier. When adjacent sensors all suffer from sensing failures, the case becomes challenge. These sensors may construct a new VC and further forward the wrong sensing value to the sink. Since our approach tries to balance the energy consumptions, the probability of a partial adjacent sensing failure is very small. Even in the case, the sink can periodically check whether the energies of more than one close sensors' energies are below the critical level. According to the affected region, the sink raises an alarm to require to add or change sensors in the monitoring area.

Forwarding failure Forwarding failure is a critical problem to our approach, since we use single path to forward values of all update VCs to the sink. If one sensor along the routing path fails, all previous collected messages will be lost. In order to address this issue, we use gossip protocol to make our approach robust. Consider a routing path $s_1 \rightarrow s_2 \rightarrow s_3$ and assume that s_2 fails forwarding message to s_3. There are three possible reasons that cause such a failure: 1) s_3 fails to receive the message; 2) the radio signal between s_2 and s_3 is breached; or 3) s_2 fails to broadcast its message. For the first two cases, if s_2 cannot hear a message from s_3 after s_2 forwarding message, s_2 will choose another sensor and resend message. For the third case, s_1 cannot hear messages sent from s_2, then s_1 will resent its message to another sensor.

The side-effect of this strategy is that s_3 correctly forwards message to the next sensor, but s_2 may not detect the message. Thus, s_2 makes wrong judgement that forwarding failure happens in between $s_2 \rightarrow s_3$. The immediate result is that two routing paths are constructed, one is from s_3 to its next sensor s_3' and the other is from s_2 to another sensor s_2'. Thus, energy is wasted. For a period of time T, sensors use the same route to forward messages. After the two routes are built up, s_2 chooses a path $s_2 \rightarrow s_2'$ to forward message, whereas, s_3 assumes that the value of previous sensors (e.g., s_2) does not change, since it does not get update message from s_2. After sink receives messages from these two routes, it records all updates and chooses one route to forward updates in the remaining of T.

7 Experimental results

We evaluate the performance of our proposed approach by simulating light monitoring experience in our lab.

7.1 Data sets

We simulate a data collection scenario of a light monitoring application based on the real sensing data collected from a network with 15 sensors in our laboratory. Figure 4 shows the topology of the sensor network and part of the real sensing data.

In order to simulate a network with large number of sensors, we develop a simulator to construct a network based on our lab data set. We build up a 30×40 grid, in which each cell contains one sensor node. We set 10K communication edges and varied communication edges between different sensors to imitate dynamic network topology. We randomly choose several cells to take the values of any 15 real sensing data in our lab test as their readings and partitioned the grid from 40 to 60 value regions. For each chosen cell in a value region, we change its values a little bit as the sensing value of a closer sensor to it. In this way, sensors in the network can read their own sensing data according to our real data set. Our simulator can vary the number of node failures and forwarding failures. Notice that sensing failures have been contained in the sensing values, since sensing values were generated based on real data test.

We let each sensor read values every five seconds and let routing time period be 1 000 seconds. According to the setting in Refs. [5, 12], where they considered the number of

forwarding and receiving packets for evaluating energy consumption, we let each sensor communicate at most 20K packets before it exhausted, where a sensor reading required 2 bytes, and each packet contained 49 bytes based on our application development experiences on the Crossbow MICA2 motes [9]. The initial battery capacity for each sensor could forward 20K packets, receive 33K packets, or do 50K sensing operations, i.e., the energy consumption is calculated as $E = x + 0.6y + 0.4z$, where x is the number of forwarding packets, y is the number of receiving packets, and z is the number of sensor readings. When $E > 20$K, we say a sensor "died". We omit each sensor's energy cost for computing.

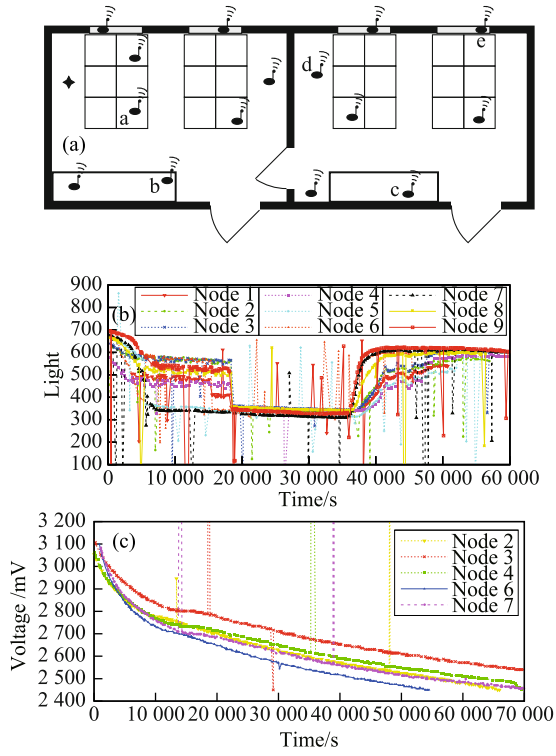


Fig. 4 Real sensing data collected from a network with 15 sensors. (a) Topology of a sensor network with 15 sensors; (b) sensing light values; (c) voltages of 5 different sensors

Different from the approaches [4,5] relying on topology of the underlying sensor network or the spatial-temporal clustering results, ASA focuses on dynamic network environment. We select snapshot [13] as the representative of state-of-the-art data collection algorithm in this set of experiments. Snapshot is the most related work to ours, which deals with both data and network dynamics.

7.2 Comparison of energy consumption on data collection

We first test energy consumption on data collection, including network lifetime, energy balance, and energy efficiency.

Network lifetime As defined in Section 1, the lifetime of a sensor network is defined to be related with $\rho\%$ connected sensors in the network. Let n be the number of sensors in the network. We say readings in the network is unreliable if the ratio of disconnected sensors exceeds a threshold $\rho\% \times n$. Figure 5(a) shows that when increasing ρ, the lifetime of the network drops.

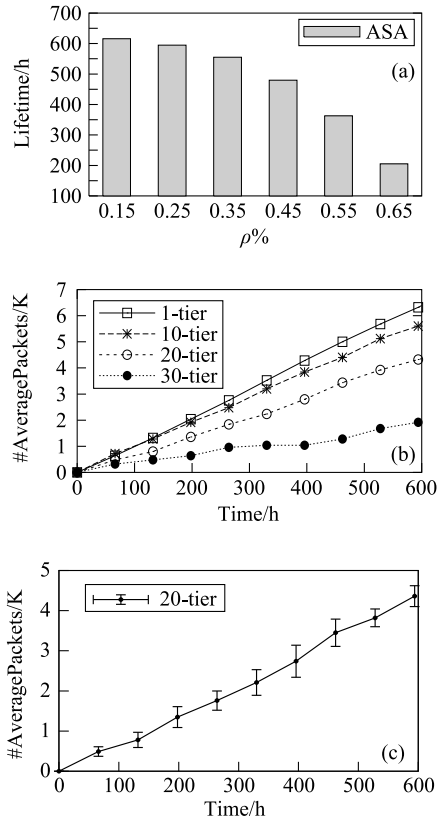


Fig. 5 Energy consumption. (a) Lifetime vs. varied $\rho\%$; (b) $\rho\% = 20\%$; (c) average packets with error bars ($\rho\% = 20\%$)

Energy balance Figure 5(b) shows the result of balancing energies when $\rho\% = 20\%$ and Fig. 5(c) shows results with error bars for sensors in 20 tiers. The results for other tiers are similar. We use TAG model [8] to classify sensors in different "tiers" and evaluate energy consumptions on different tiers. We choose nodes with different distances to the sink to do the test and consider three kinds of energy consumptions: forwarding messages, receiving messages, and sensing readings. The figure shows that ASA can effectively decrease the energy consumption ratio of inner "tier" sensors compare with outer layer sensors.

Energy efficiency (i.e., energy consumption comparison between snapshot [13] and ASA) We implement snapshot and compare its performance with ASA. Figure 6 shows comparison results between snapshot and ASA.

Figure 6(a) shows the number of exhausted nodes when in-

creasing time period. As expected, we find that the first node failure by using ASA is at the 60th hour, which is much later than snapshot. This is because ASA is an energy-balanced approach. There are 30 nodes in the first layer of the network. After these nodes fail, the outer layer nodes are disconnected to the sink. Figure 6(b) shows that the network produced reliable readings within 600 hours when $\rho\%$ is set to 20% for both two approaches. The network lifetime using ASA is much longer than the lifetime using snapshot.

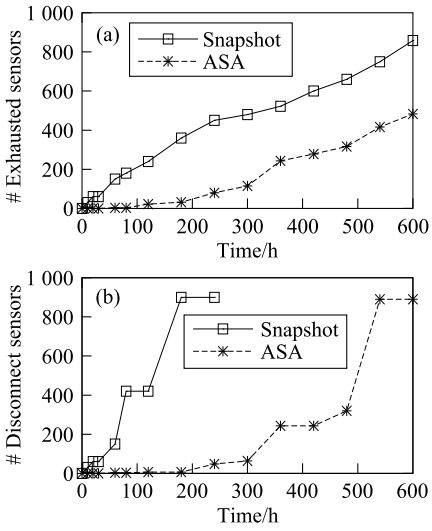


Fig. 6 Lifetime comparison ($\rho\%$ = 20%)

We also find that only a small number of outer layer sensors encounter an energy problem before inner tier nodes become exhausted. Therefore, the number of disconnected nodes in Fig. 6(b) increases much sharper than the number of exhausted nodes in Fig. 6(a).

7.3 Energy consumption on self-maintenance

Figure 7 shows the results of DVCs expansion and maintenance during the first 12 time steps in the network initialization. In order to evaluate our approach, for each time step, we collect all sensing readings in the network and calculate the minimal number of DVCs as an optimal solution. We call this approach *Global* in Fig. 7(a). We then generate DVCs using ASA without any global knowledge and compare their numbers with the minimal number of DVCs using optimal solution.

Figure 7(a) shows the comparison results. At the first time step, ASA builds up 140 DVCs, and then the number of DVCs becomes smaller and inclines to the real number of DVCs. It indicates that the DVC expansion technique is effective. It is interesting to find that when increasing the number of time steps, the DVCs numbers increase to a peak and then drop quickly. This is because that updating sensing readings can change the number of DVCs. Upon the updates, ASA reconstructs several small DVCs to break a previous one, so the number of DVCs increases. Then ASA adopts DVC expansion technique to merge those small DVCs quickly. Figure 7(b) shows the scalability of ASA when increasing time duration. We can see that the accumulative communication costs increase slightly and keep stable when time goes by. This is because that during the remaining time periods, the major energy consumption is for reading values.

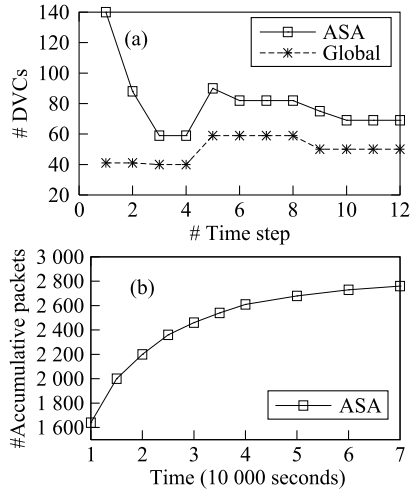


Fig. 7 DVCs expansion and maintenance. (a) Quality of DVCs; (b) maintenance costs

7.4 Effect of routing maps

We evaluate the sizes of different types of routing maps and their impact to the number of routing sensors (see Fig. 8).

Figure 8(a) shows the sizes of different types of routing maps when building up a new routing. We compare three types of routing maps, which are 1) routing map without compression (denoted RM), 2) compact map with minimal size (denoted min-CRM), and 3) relax compact routing map (denoted relax-CRM).

As Fig. 8(a) shows that the size of RM keeps stable, since RM describes all sensors in the network. Min-CRM and relax-CRM require fewer sizes to transmit routing navigation messages. Figure 8(b) shows the number of needed routing sensors at different period of times when ASA adopts min-CRM. Compared with the globally minimal number of routing sensors, ASA requires more sensors for routing at the beginning of the routing period. The difference between the number of routed sensors and minimal number of routing sensors will be smaller when increasing time period, when ASA self-adaptively finds good solution gradually.

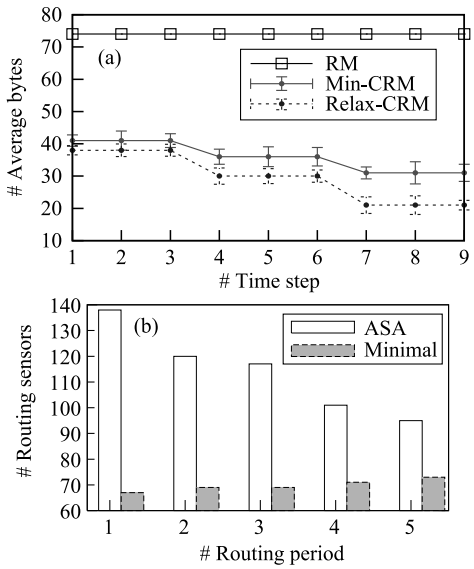


Fig. 8 Self-adjusted routing map size. (a) Number of routing maps; (b) number of routing sensors

7.5 Robust to failures

We show that ASA is robust to failures in this section. We used message loss rate to quantify forwarding failures. Figure 9(a) shows the number of routing sensors when varying the message loss rate. We let the message loss rate vary from 5% to 30%. We find that ASA requires much more routing sensors when increasing the message loss rate. It is not surprising since more forwarding failure sensors require longer paths to forward updates to the sink.

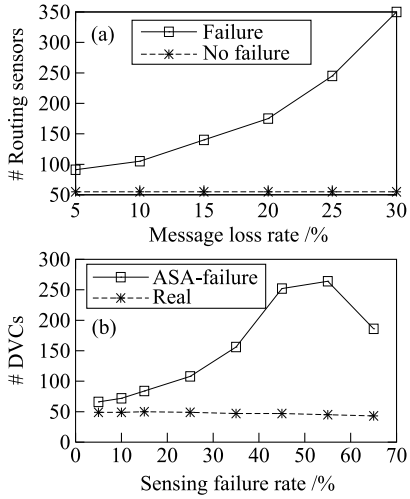


Fig. 9 Impact of failures. (a) Forwarding failure; (b) sensing failure

Figure 9(b) shows the impact of energy consumption when varying sensing failure rate, i.e., ratio of number of incorrect sensing nodes to the number of all sensors in the network. We set a fix number of VCs in advance and test the effect of sensing failure rate. When sensing failure rate is small, the number of DVCs is close to 50, the real number of VCs. However, when more sensing failures happen, sensors have large abilities to use wrong sensing values to construct small DVCs, which cause the increments of the number of DVCs. This number climbs to a peak and then drops sharply, which is consistent to our observations that most low-energy sensors will produce the same but wrong sensing values.

8 Related work

A number of research work have been published on energy efficient data collection over sensor networks. We can classify them into centralized control approaches [14–21] and decentralized control approaches [8, 13, 22–26].

Centralized control approaches Most of the existing prediction model-based approaches adopt centralized control strategy. Based on collected historic data, a sink (or PC base station) generates a prediction model [17–19] and disseminates this model to the network. Both sensor and sink maintain a prediction model of how data evolve and keep the model in synchronism. Barbie-Q: a tiny model query system (BBQ) [14] is a model-driven data acquisition framework that adopts a trained statistical model. BBQ can limit the number of required sensor readings to answer queries with high confidence. Potsch et al. [21] also adopted model-driven model to collect temperature sensor readings. Deshpande et al. [14] used a global model to capture dependencies based on a relatively stable network topology. Both *Ken* framework [16] and Kalman filter [15] fit the data into a model, in which sensing data can be forwarded to the sink only when they are beyond an approximation threshold. A disjoint-clique model is also proposed in Ref. [16] to partition sensor attributes to multiple localized clusters. CONCH [2] is a value-based data collection approach. The base station globally determines and disseminates a minimal CONCH plan into the whole network so that both temporal and spatial suppression can be combined for saving as much communication energy cost as possible.

Different from the above centralized control approaches, our work, ASA, allows an individual sensor self-adaptively to make local decisions for data suppression without global controller. It enables ASA to deal with the changes of sensing data, network topology, sensor deployment, and failures. Considering these dynamic cases, data distribution update may frequently arise and it is hard to find a perfect model adapted.

Decentralized control approaches TAG [8, 23] can be cat-

egorized as a decentralized control approach. Upon receiving a broadcast message from the sink, each individual sensor self-determines their parent nodes and builds up a routing tree rooted at the sink. The most related work to ours is snapshot [13], which also deals with both data and network dynamics. The primary difference between snapshot and ASA is the definition of clusters and forwarding path chosen. In snapshot, the sensor nodes of a cluster are within one hop count and only the chosen representative node can produce approximate data [27]. ASA, however, can partition the network into several multi-hop disjoint clusters (a.k.a. VCs). Any sensors in a value-based cluster can forward sensing readings. Lin et al. [24] used selective sampling technique to increase the network lifetime. A quiet similar cluster definition to our disjoint value clusters is proposed in Ref. [22]. Different from the approach in Ref. [22], ASA self-adaptively finds a "minimal" routing path to concatenate all clusters to the sink. Using this way, ASA saves more communication cost. Furthermore, ASA only forwards updates to the sink. In addition, self-adaptive routing protocols for sensor network are well-developed in network domain [28]. The above network-centric approaches mainly consider network topology, whereas, ASA is a data-centric approach, which considers sensing readings along the routing path to dynamically expand and maintain clusters for increasing the network lifetime.

Exploring spatial and temporal correlations EEDC [5] is a data collection framework based on a careful analysis of the surveillance data reported by the sensors. By exploring the spatial correlation of sensing data, it dynamically partitions the sensor nodes into clusters so that the sensors in the same cluster have similar surveillance time series. They can share the workload of data collection in the future since their future readings may likely be similar. Furthermore, during a short-time period, a sensor may report similar readings. Such a correlation in the data reported from the same sensor is called temporal correlation, which can be explored to further save energy. Vuran et al. [12] considered the nature of the physical phenomenon constitutes the temporal correlation between each consecutive observation of a sensor node. A theoretical framework is developed to model the spatial and temporal correlations in sensor networks and efficient communication protocols are developed.

The above two approaches are based on an implicit assumption that collection of data and the topology of a WSN are correlated, which might not be true in some cases, for example, the sensor network in a parking lot. Our data transformation route does not depend on the network topology. Instead, it can dynamically adapt according to the sensing values.

9 Conclusion

In this work, we present ASA approach, which is an energy-efficient and balanced data collection method based on adaptive map forwarding strategy. Moreover, ASA investigates the spatial correlations between sensor readings to provide an energy efficient route without knowing the global network topology of WSNs. The experimental results demonstrate that ASA provides significant energy savings and equal energy consumption among all sensor nodes, and is robust to failures.

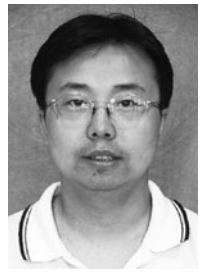
Acknowledgements The work was partially supported by the National Natural Science Foundation of China (NSFC) for Outstanding Young Scholars (61322208), the National Basic Research Program of China (973 Program) (2012CB316201), the NSF of China (Grant Nos. 61572122, and 61272178), the NSF of China for Key Program (61532021), NSF (CNS-1422355), and ARO (#66276-CS).

References

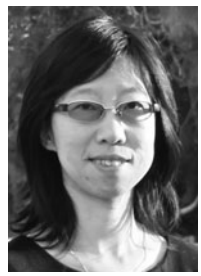
1. Tan H Ö, Körpeoğlu I. Power efficient data gathering and aggregation in wireless sensor networks. ACM SIGMOD Record, 2003, 32(4): 66–71
2. Silberstein A, Braynard R, Yang J. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. 2006, 157–168
3. Sharaf A, Beaver J, Labrinidis A, Chrysanthis K. Balancing energy efficiency and quality of aggreagation data in sensor networks. The VLDB Journal — The Internadional Journal on Very Large Data Bases, 2004, 13(4): 384–403
4. Xu Y, Heidemann J, Estrin D. Geography-informed energy conservation for Ad Hoc routing. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking. 2001, 70–84
5. Liu C, Wu K, Pei J. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. IEEE Transactions on Parallel and Distributed Systems, 2007, 18(7): 1010–1023
6. Moore D, Leonard J, Rus D, Teller S. Robust distributed network localization with noisy range measurements. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems. 2004, 50–61
7. Pottie G J, Kaiser W J. Wireless integrated network sensors. Communications of the ACM, 2000, 43(5): 51–58
8. Madden S, Franklin M J, Hellerstein J M, Hong W. TAG: a Tiny AGgregation service for Ad-Hoc sensor networks. In: Proceedings of the 5th Symposium on Operating System Design and Implementation. 2002, 313–325
9. Crossbow Technology, Inc. MPR-mote processor radio board user's manual. 2003
10. Kempe D, Kleinberg J, Demers A. Spatial gossip and resource location

protocols. Journal of the ACM, 2004, 51(6): 943–967

11. Zhang L, Ye Q, Cheng J, Jiang H B, Wang Y K, Zhou R, Zhao P. Fault-tolerant scheduling for data collection in wireless sensor networks. In: Proceedings of IEEE Global Communications Conference. 2012, 5345–5349
12. Vuran M C, Akan Ö B, Akyildiz I F. Spatio-temporal correlation: theory and applications for wireless sensor networks. Computer Networks, 2004, 45(3): 245–259
13. Kotidis Y. Snapshot queries: towards data-centric sensor networks. In: Proceedings of the 21st International Conference on Data Engineering. 2005, 131–142
14. Deshpande A, Guestrin C, Madden S R, Hellerstein J M, Hong W. Model-driven data acquisition in sensor network. In: Proceedings of the 30th International Conference on Very Large Data Bases. 2004, 588–599
15. Jain A, Chang E Y, Wang Y F. Adaptive stream resource management using Kalman filters. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. 2004, 11–22
16. Chu D, Deshpande A, Hellerstein J M, Hong W. Approximate data collection in sensor networks using probabilistic models. In: Proceedings of the 22nd International Conference on Data Engineering. 2006, 48
17. Silberstein A, Puggioni G, Gelfand A, Munagala K, Yang J. Making sense of suppressions and failures in sensor data: a Bayesian approach. In: Proceedings of the 33rd International Conference on Very Large Data Bases. 2007, 842–853
18. Yang X Y, Lim H B, Özsu T M, Tan K L. In-network execution of monitoring queries in sensor networks. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. 2007, 521–532
19. Ahmad Y, Nath S. COLR-Tree: communication-efficient spatio-temporal indexing for a sensor data Web portal. In: Proceedings of the 24th International Conference on Data Engineering. 2008, 784–793
20. Li J, Deshpande A, Khuller S. On computing compression trees for data collection in wireless sensor networks. In: Proceedings of the 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies. 2010, 2115–2123
21. Potsch T, Pei L, Kuladinithi K, Goerg C. Model-driven data acquisition for temperature sensor readings in wireless sensor networks. In: Proceedings of the 2014 IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing. 2014, 1–6
22. Meka A, Singh A K. Distributed spatial clustering in sensor networks. In: Proceedings of the 10th International Conference on Extending Database Technology. 2006, 980–1000
23. Bhattacharya A, Meka A, Singh A K. MIST: Distributed indexing and querying in sensor networks using statistical models. In: Proceedings of the 33rd International Conference on Very Large Data Bases. 2007, 854–865
24. Lin S, Arai B, Gunopulos D, Das G. Region sampling: continuous adaptive sampling on sensor networks. In: Proceedings of the 24th International Conference on Data Engineering. 2008, 794–803
25. Li Z J, Li M, Wang J L, Cao Z C. Exploiting ubiquitous data collection for mobile users in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 2013, 24(2): 312–326
26. Wang C, Ma H D. Data collection in wireless sensor networks by utilizing multiple mobile nodes. Ad Hoc & Sensor Wireless Networks, 2013, 18(1): 65–85
27. Wang C, Ma H D, He Y, Xiong S G. Adaptive approximate data collection for wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(6): 1004–1016
28. Buragohain C, Agrawal D, Suri S. Power aware routing for sensor databases. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies. 2005, 1747–1757



Bin Wang received the PhD degree in computer science from Northeastern University, China in 2008. He is currently an associate professor in school of Computer Science and Enginering at Northeastern University. His research interests include design and analysis of algorithms, databases, data quality, and distributed systems. He is a member of the CCF.



Xiaochun Yang received the PhD degree in computer science from Northeastern University, China in 2001. She has been a professor at Northeastern University since 2008. Her research interests include data quality, data privacy, and distributed data management. She has received a China Program Award for New Century Excellent Talents in Universities. She is a member of the ACM, the IEEE Computer Society, and a senior member of the CCF.

Guoren Wang received the PhD degree from Northeastern University, China in 1996. He is currently a professor in School of Computer Science and Enginering at Northeastern University. His research interests include XML data management, query processing and optimization, bioinformatics, high-dimensional indexing, parallel database systems, and P2P data management. He is a senior member of the CCF.

Ge Yu received the BE and ME degrees in computer science from Northeastern University, China in 1982 and 1986, respectively, and the PhD degree in computer science from Kyushu University, Japan in 1996. He has been a professor at Northeastern University since 1996. His research interests include database theory and technology, distributed and parallel systems, embedded software, and net-

work information security. He is a member of the IEEE, the ACM, and a fellow of the CCF.

Wanyu Zang joined Texas A&M University at San Antonio, USA as an assistant professor in 2015. She graduated with a PhD in computer science from the Nanjing University, China in 2001. Prior to joining Texas A&M University at San Antonio, She worked at Virginia Commonwealth University, USA and Western Illinois University, USA as an assistant professor and Pennsylvania State University as a postdoctoral researcher. Her research interests are in network security and cloud security, especially the security in multi-channel multi-interface network and virtual machine placement, which is supported by Army Research Office (ARO).

Meng Yu joined Department of Computer Science at University of Texas at San Antonio, USA in 2015 as an associate professor. He graduated with a PhD in computer science from the Nanjing University, China in 2001. Prior to joining Texas A&M University at San Antonio, he worked at Virginia Commonwealth University, USA as an associate professor, Western Illinois University, USA and Monmouth University, USA as an assistant professor, and Pennsylvania State University, USA as a postdoctoral researcher. His research interests include cloud computing security and system recovery. His research has been supported by multiple National Science Foundation (NSF) and Army Research Office (ARO) grants.