An Adaptive State-Aware Routing Algorithm for Data Aggregation in Wireless Sensor Networks

Yalin Nie[1], Sanyang Liu[2], Zhibin Chen[3], and Xiaogang Qi[2]
[1]School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi 710071, China
[2]School of Science, Xidian University, Xi'an, Shaanxi 710071, China;
[3]School of Microelectronics, Xidian University, Xi'an, Shaanxi 710071, China
Email: nieyalin111@163.com; {liusanyang, zhibinchen2013, qixiaogang2013}@126.com

***Abstract*—**Due to different data correlation among events in driven Wireless Sensor Networks (WSNs), over-overlapping paths from events in order to maximize the data aggregation will weaken the monitoring ability of WSNs instead of improving the network performance. There should be a tradeoff between data aggregation maximization and energy balance. Excessive pursuit of high data aggregation regardless of the actual state of network could bring about premature death of some backbone nodes, leading to unstable network structure. Based on the problems, in this paper, a novel adaptive state-aware routing algorithm for data aggregation is proposed. The algorithm maximizes the possible data aggregation by building and updating a Hop-Tree, takes the local state of nodes to build and maintain Hop-Tree to gain better adaptation to heterogeneous wireless sensor networks, depends on *T*ime-*T*o-*L*ive (*TTL*) mechanism to limit the Hop-Tree update range to avoid over-overlapping of paths according to the correlation of events, and designs a forced path building strategy to balance the data load on the backbone nodes of Hop-Tree to further balance the energy consumption. Theoretical analysis and simulation results show that our algorithm can maximize the possible data aggregation while balance the energy consumption among nodes and enhance the monitoring ability of WSNs significantly.

***Index Terms*—**wireless sensor networks, hop-tree, cluster, data aggregation, energy balance

I. Introduction

Wireless Sensor Networks (WSNs) consist of sensor nodes deployed over a geographical area for cooperatively monitoring physical phenomena. Generally, sensor nodes are powered by batteries with limited energy budget which is impossible or inconvenient to recharge because of the unpractical environment or the huge numbers of sensor nodes. Energy is very scarce for WSNs. Many researches show that the energy spent on data transmission dominates the whole energy consumption for sensor nodes.

Normally, the density of WSNs is high and there is a large amount of redundant data transmitted in network being collected by nodes which are spatially close. In order to reduce the energy consumption, in-networking data aggregation is often executed for reducing the amount of data transmission by eliminating the inherent redundancy in raw data collected by source sensor nodes. Essentially, data aggregation depends on where and when data meet together, that is, raw data collected by source nodes should be convergent in space and time.

Routing plays an important role in data aggregation process. How to route data in order to achieve effective data aggregation is an important topic in WSNs. In event-driven WSNs, a fixed routing structure is not suitable for data aggregation because it does not take advantage of the correlation between data to reduce data redundancy, resulting in large data load. It is important to dynamically build routes overlapping as much as possible according to the events for efficient data aggregation. Under normal circumstances, the closer the nodes are apart, the more correlated data they collect and the higher the degree of data aggregation is, and the farther, the less and the lower. So, when the events are far apart, over-overlapping of routes will not gain benefit from data aggregation because of the low data correlation which prevents high degree aggregation, but also results in uneven energy consumption which leads to unstable network structure. A good routing protocol for data aggregation should overlap routes according to data correlation and local state and balance the energy consumption for the whole network in order to enhance the monitoring ability of WSNs. Here we propose a Novel Adaptive State-Aware Routing Algorithm for data aggregation (ASARA). Our algorithm constructs an adaptive routing tree with shortest paths based on events and nodes' local states, which can maximize data aggregation according to data correlation, well balance node energy consumption and route data in a reliable way.

The rest of the paper is organized as follows: Section 2 introduces some related work. Section 3 details ASARA algorithm. Section 4 shows the algorithm analysis and experiments. Finally, Section 5 presents our conclusions and future work.

II. Related Work

For better data aggregation, cluster and tree structure are often used. The Low-Energy Adaptive Clustering Hierarchy (LEACH) [1] algorithm is the first work based

[1] Manuscript received March 22, 2013; revised May 11, 2013.
This work was supported by Youth Fund of Luoyang Institute of Science & Technology under Grant No. 2008QZ11.
Corresponding author email: liusanyang@126.com.
doi:10.12720/jcm.8.5.296-306

on cluster, in which cluster-heads act as aggregation nodes and communicate directly with the Sink. Hybrid Energy-Efficient Distributed Clustering (HEED) [2] algorithm periodically selects cluster heads according to a hybrid of their residual energy and a secondary parameter (e.g. node degree), balancing load among cluster heads to extend the lifetime of WSN. CPCP [3] is a coverage-aware clustering algorithm designed for applications with the coverage preservation requirements. Jiguo Yu *et al*. proposed EADC algorithm [4] to cope with clustering for nonuniform distributed WSN. Shortest Path Tree routing (SPT) [5] is one of the most typical routing strategies based on tree in WSNs. It makes each node transmit its collected data along the shortest path from itself to the Sink, with data aggregation occurring at any intersection of paths.

All of the algorithms above do not consider the data correlation while clustering or building tree. DACP [6] makes full use of predicted data to help cluster heads decide whether send aggregated data or not, improving the efficiency of data aggregation. Aiming event-driven scenarios, the Information Fusion-based Role Assignment (InFRA) algorithm [7] clusters source nodes detecting the same event and maximizes the overlap of shortest paths from clusters to the Sink. Woo-Sung Jung *et al*. [8] proposed a clustering technique based on the status of the network to improve the data aggregation and energy efficiency. Leandro A. Villas *et al*. proposed the Dynamic and Scalable Tree (DST) [9] algorithm, which reduces the number of working nodes according to the correlation requirement of applications and maximizes the overlap of routes regardless of the order of event occurrence.

Under the assumption of perfect aggregation for both of intra- and inter- event data, the routing optimization problem for best data aggregation is equivalent to the Steiner Tree Problem. DRINA [10] is a routing algorithm for data aggregation with the aim to find an approximate Steiner tree, with a reduced number of messages for setting up a routing tree, maximized number of overlapping routes, high aggregation rate, and reliable data aggregation and transmission. When the assumption is not held, such problem is no longer equivalent to finding the Steiner Tree. It involves many aspects, such as the correlation of data, load balance, duration of event, and so on. So, there are some shortcomings of DRINA as follows:

a) The earlier the path is built, the heavier the load of the nodes on the path is. This will cause premature death of those nodes.
b) It might cause some data to be transmitted along longer paths, which would increase the total energy consumption.
c) It does not consider the correlation of events, and assumes that the data from different event areas could be aggregated perfectly (that is *n* packets could be aggregated into one packet), which is unrealistic in many cases. For example, if two events are far apart, making their paths overlap cannot gain benefit from data aggregation because the data from the two events cannot be aggregated or can be aggregated just a little, but leading to more imbalanced energy consumption on the overlapping part of the paths.

In this work, we improve DRINA and propose a novel Adaptive State-Aware Routing Algorithm for data aggregation (ASARA). ASARA utilizes the local states of nodes to build and update a Hop-Tree for data routing and aggregation, balancing energy consumption. It builds shortest paths from the events area to the Hop-Tree or the Sink based on the correlation between the events and the energy state of the Hop-Tree to form new backbones of the Hop-Tree. It can maximize the overlapping of routes to achieve better data aggregation when events are in close proximity, shorten the data transmission paths while the correlation of events is low, and further balance the energy consumption of the Hop-Tree to enhance the network monitoring capabilities. Simulation results confirm the effectiveness of ASARA.

III. Adaptive State-Aware Routing Algorithm for Data Aggregation in WSNs (ASARA)

The main goal of ASARA is to build an adaptive routing tree (Hop-Tree) for in-networking data aggregation according to local states and event correlation which can maximize the data aggregation while balance energy consumption, to enhance the monitoring capability of WSNs. It consists of 5 parts: Hop-Tree Building, Cluster Formation, Path Building, Hop-Tree Update, and Routes Maintaining.

Here we use the terms Coordinator, Collaborator and Sink, with the same meanings as they are in [10], to indicate the roles of nodes and define the State of a node as follows:

$$State(u) = \alpha \frac{E_R}{E_{\max}} + \beta \frac{M_R}{M_{\max}} + \gamma \frac{1}{L+1}, \qquad \alpha + \beta + \gamma = 1$$

where E_R is the residual energy of node u, $E_{\max}$ is the maximum energy of nodes, M_R is the residual memory of u, $M_{\max}$ is the maximum memory of nodes, and L is the link packet loss rate of u.

A. Hop-Tree Building

In this work, a Hop-Tree rooted at Sink is formed after deploying sensor nodes, with the shortest paths (in hops) that connect all source nodes to the Sink while maximizing the possible data aggregation and balancing energy consumption. Each node has two parameters: HTT (Hop-To-Tree) and HTS (Hop-To-Sink), recording the minimum number of hops from the node to the Hop-Tree and to the Sink respectively. The value of HTT for a node might be changed when the Hop-Tree updates because of some new events, while the value of HTS for each node does not change. At the beginning, the HTT and the HTS are the same, then different as events occur

one after another. The HTT of the Sink and any node on the backbone of the Hop-Tree is 0. The farther the node is away from the backbone of the Hop-Tree, the larger the HTT is. The HTT of any node changes in 2 cases: (a) it becomes a backbone node of the Hop-Tree, or (b) it receives a *H*op *C*onfiguration *M*essage (*HCM*) which announces better distance.

HCM is a four-tuple as < Type, ID, HTT, State>, where Type specifies *HCM* message, ID is the identifier of a node that started or retransmitted the *HCM* message, HTT is the distance by which an *HCM* message has passed, and State records the State of the sender.

Hop-Tree building algorithm is shown in Table I. At the beginning, the Sink set its HTT 0, and other nodes infinity. This phase is started by the Sink flooding an *HCM* with HTT 1. On receiving an *HCM* message, each node compares its HTT with the HTT in the *HCM* message. If there is a shorter way to the Sink, the node will update the relevant information and retransmit the *HCM*, as shown in Lines 3-6. If there are several nodes which are next hop candidates, a node with better State will win (Lines 7-8). Otherwise, the node will take no action but discard the received *HCM*. This process runs repeatedly until a Hop-Tree rooted at the Sink is built.

Before the first event taking place, there is no established path (that is the Sink constructs the backbone of the Hop-Tree), the HTT has the same meaning with the HTS. So the HTS of each node is set equal to its initial HTT. After event occurrence, HTT which is the smallest hops to the available backbone of the Hop-Tree becomes different from HTS.

TABLE I. HOP-TREE BUILDING ALGORITHM

1. The Sink floods an *HCM* message with HTT=1;
2. For each node *u* that received an *HCM* message
3. If HTT(*u*)> HTT(*HCM*)
4. NH(*u*)=ID(*HCM*); % NH (u) stores the next hop of node u
5. HTT (*u*)=HTT (*HCM*); ID(*HCM*)=ID(*u*); HTT (*HCM*)= HTT (*u*)+1; State(*HCM*)=State(u);
6. *u* transmits the *HCM* message to its neighbors;
7. Else If (HTT (*u*)==HTT (*HCM*) & State(NH(*u*))< State(*HCM*))
8. NH(*u*)=ID(*HCM*);

B. Cluster Formation

When an event occurs, a cluster based on the nodes which detect it (we may call them event nodes) will be formed. The key process of the cluster formation phase shown in Table II is the election of the leader node (called Coordinator) for the cluster, and the information delivery in this phase is by means of *C*luster *C*onfiguration *M*essage (*CCM*). *CCM* is also a four-tuple: < Type, ID, HTT, State>, where ID is the identifier of the node that started the MCC message, HTT and State fields store the HTT and State value of the node with the identifier ID separately.

At the beginning, all event nodes are eligible candidates. Any node that has detected the event sets its role Coordinator, constructs a *CCM* message and sends it out, as shown in Lines 1-3. The node with the shortest path to the Hop-Tree will become the Coordinator, shown in Lines 5-7. If there are several nodes have the smallest HTT value, the one with the best state will be the final victor, detailed Lines 8-10. Other nodes in the event area will downgrade to the Collaborators.

TABLE II. CLUSTER FORMATION ALGORITHM

1. For each node *u* that detected the event
2. Role(*u*)=Coordinator;
3. *u* creates an *CCM* and broadcasts it;
4. For each node *u* that received a *CCM* message of the same event
5. If HTT (*CCM*)<HTT(*u*)
6. Role(*u*)=Collaborator;
7. *u* retransmits the *CCM*;
8. Else If HTT(*CCM*)==HTT(*u*) & State (*CCM*)>State (*u*)
9. Role(*u*)=Collaborator;
10. *u* retransmits the *CCM*;

C. Path Building

Each Coordinator, the elected group leader, starts establishing a path which will be the new part of the backbone of the Hop-Tree with the help of *P*aths *B*uilding *M*essage (*PBM*), *F*orced *P*aths *B*uilding *M*essage (*FPBM*) and *R*ebuilding *M*essage (*RM*), all of which are 2-tuple: <Type, Path> where Path records the nodes in the path from the Coordinator to the backbone of the Hop-Tree or the Sink depending on the Type. Path Building Algorithm is detailed in Table III.

TABLE III. PATH BUILDING ALGORITHM

1. If HTT(*v*)==0 %Node *v* is a Coordinator;
2. There is no need to build path;
3. Else If Energy(NH(*v*))<Threshold_Energy & Node *v* can find a neighbor who satisfies: a) its residual energy is greater than Threshold_Energy, b) with less HTS.
4. *v* informs (or sends a *RM* along the Path in reverse direction to inform) the nodes within event field to enter the Forced Path Building phase.
5. Else
6. *v* appends itself to the Path and sends a *PBM* message to its next hop;
7. For each node *u* that received a *PBM* message
8. If HTT (*u*)==0
9. Path building successes,;
10. Else
11. *u* acts similarly to the behaviors shown in the Lines 3-6. %Forced Path Building phase
12. The node with the smallest HTS and best state in the event field will be chosen as the new Coordinator, denoted by u
13. NH(*u*)=*v*, *v* is a neighbor of u with the less HTS and best state;
14. *u* appends itself to the Path and sends a *FPBM* to its next hop;
15. For each node *w* received an *FPBM* message
16. If *w*==Sink
17. Forced path building successes.
18. Else
19. *w* acts similarly to the behaviors shown in the Lines 13-14 ;

If the Coordinator is on the backbone of the Hop-Tree, there is no need to build a new path to act as a new backbone of the Hop-Tree. Otherwise, the Coordinator might create a *PBM* message and sends the *PBM* to its

next hop or enter the forced path building phase according to the residual energy of the next hop, as shown in Lines 1-6. Any node who received a *PBM* will acts as the Coordinator, and if there is no need for forced path building, the process will be repeated until the *PBM* arrives at the Sink or a backbone node, depicted in Lines 7-11. When the forced path building phase is boosted, each node involved will find the neighbor with less HTS value and the best state as its next hop. The forced path building process fully depends on the delivering of *FPBM* and HTS value, and is ended when the Sink is found, as shown in Lines 12-19.

D. Hop-Tree Update

A Hop-Tree should be updated for gaining shortest paths connecting all source nodes while maximizing data aggregation and balancing energy distribution as events occur one by one. We update the HTT value of each node properly to achieve this goal.

Data aggregation is based on the correlation among data. The more correlated the data are, the better the result of data aggregation is (that is the less aggregated data and the more accurate reconstructed data from the aggregated data). Often, spatial correlation of data sensed by nodes depends on the spatial distance between events. The closer the two events are apart, the more correlation the sensed data are, and the farther apart, the less correlation. Usually, data from two far apart events could not be aggregated. So, if the Hop-Tree is updated to force the paths from the two distant events to overlap as early as possible and the data from the two corresponding Coordinators are aggregated forcedly, the data reconstruction will be very poor, or if the data are not aggregated, energy consumption for transmitting these data will increase because of a non-shortest path to the Sink and the load of the nodes on the overlapping path will be heavier resulting in uneven energy consumption.

So, ASARA takes a *TTL* (*T*ime-*T*o-*L*ive) indicator which records the transmission distance of *HCM* to limit the transmission scope of *HCM* in order to avoid the situation described above. This method is simple but has two benefits compared with DRINA: <1> The number of *HCMs* transmitted in the network is reduced, saving energy. <2> If two events far apart from each other occur, the shortest path will be built for the later event regardless of the affection caused by the previous event. Assume that when the distance between two events is greater than 3 hops the correlation between the two events is very low, from Fig. 1, it is very clear that the energy consumption and balance of DRINA is worse than that of ASARA. The initial state of the Hop-Tree is shown in Fig. 1(a). With DRINA, when an event happens, the Hop-Tree is updated across the whole network, shown in Fig. 1(b). If another event happens, a new path will be built overlapping the former path as early as possible according to the DRINA, causing larger energy consumption (data transmission is not along the shortest path implied by the red dotted line with arrow) and heavier load on the overlapping path, shown as Fig. 1(c). Fig. 1(d) shows the state of Hop-Tree after one event for ASARA, which is updated within 3 hops (controlled by *TTL*) started from the nodes on the path. When another event happens apart from the former more than 3hops, the path is built regardless of the affection of the former path, shown in Fig. 1(e).

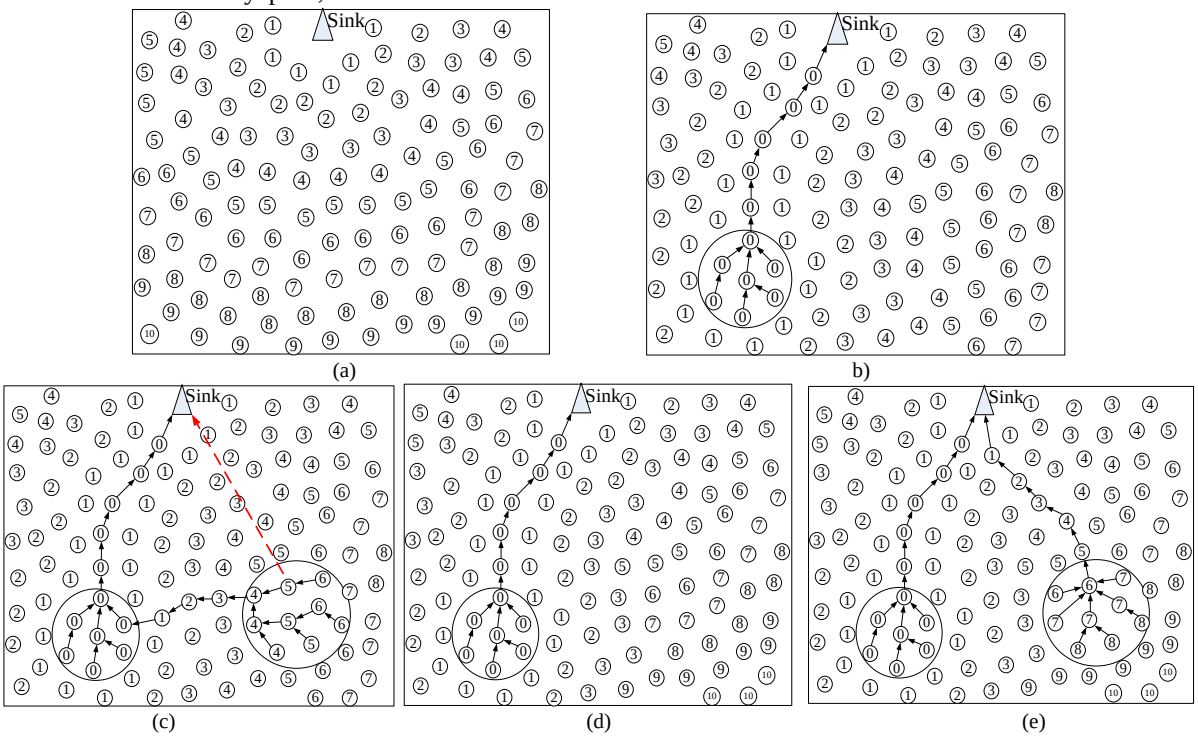


Figure 1. (a) the initial state of the Hop-Tree, (b) and (c) the state of Hop-Tree after an event for DRINA and ASARA respectively, (b) and (d) the building of the new path based on the former Hop-Tree when another event happens for DRINA and ASARA respectively.

The Hop-Tree update algorithm is depicted in Table IV. After the new shortest path from the new Coordinator being built, it should become a part of the Hop-Tree backbone. Consequently, each node in the path set the HTT value 0 (Line 2). From another perspective, the HTT 0 means a backbone node in the Hop-Tree. Then, each of those nodes constructs an *HCM* message and floods it out (Line 3), and any node that received an *HCM* message follows the same mechanisms of Hop-Tree Formation algorithm, to form a new Hop-Tree with more potential for building a better Hop-Tree on aggregation when another new event happens. Pay attention that the *HCM* flooding is limited by the *TTL* indicator. To reduce overhead, the HTT field of *HCM* is used to work as *TTL*.

TABLE IV. HOP-TREE UPDATE ALGORITHM

1. For each node *u* in the new path from the Coordinator to the Sink
2. HTT(*u*)=0;
3. *u* creates an *HCM* and floods it out;
4. For each node *u* that received an *HCM* message & HTT(*HCM*)<Range
%We take advantage of the HTT field of *HCM*, and let it acts as the role of *TTL* to limit the transmission range of *HCM*, which is set according to applications.
5. *u* acts the same with the Hop-Tree Building Algorithm-Lines 3 to 8

E. Routes Maintaining

The Hop-Tree is unique, and any failure of nodes will cause disruption in data transmission. The route repair mechanism in DRINA algorithm is a kind of piggybacked, ACK-based mechanism, and it may cause such problems as a longer delay due to several faulty nodes in the path and routing loops. ASARA takes a pretreatment for route repair to settle those problems. By means of periodic MAC communication, nodes can determine whether a neighbor dies or not. Any node that finds its next hop died chooses a neighbor which has lower HTT level and better state as its next hop. If a node could not find a neighbor as its next hop, the node will set its HTT infinity (called abnormal node) and inform its neighbors to change their next hop if necessary. When such case appears, it shows the local state around the node is poor.

IV. THEORETICAL ANALYSIS AND SIMULATION EVALUATION

A. Theoretical Analysis

ASARA is an improved algorithm based on DRINA. So, we analyse ASARA here compared with DRINA.

1) ASARA is more suitable than DRINA for heterogeneous WSNs.

In the Hop-Tree building and update phases, when there are several candidates for a node choosing next hop, ASARA makes the node with best local state win in the competition while DRINA only chooses the node whose *HCM* first reaches not considering the state of nodes. Naturally, the nodes with better states, such as more residual energy and memory and better link quality, to form the routing structure would result in a reasonable network topology for heterogeneous WSNs. So, ASARA is more suitable than DRINA for heterogeneous WSNs.

2) ASARA leads to less transmission energy than DRINA when the correlation of events is low.

For DRINA, longer paths are often formed because of over-overlapping paths from events which are far apart from each other and with low correlation, and it wastes much energy to transmit data along non-shortest paths. ASARA utilizes *TTL* to determine the Hop-Tree update scope so as to decrease the opportunity for paths from the events with low correlation overlapping. If the events are apart by more than *TTL* hops, the Hop-Tree update caused by the former event will not affect the Hop-Tree structure of those areas which are out of *TTL* hops from the event, and the latter event will build the shortest path not over-overlapping the former path, leading to data transmission energy saving for non-curving paths.

As an illustration, we show how transmission energy is saved by ASARA concretely. After the network deployment, suppose that there happen two events which are apart from Sink both d_1 hops, denoted by E_1 and E_2, with E_1 occurring before E_2 and being separated by d_2 hops, where $TTL < d_1$ and $TTL < d_2 < d_1$. For E_2, DRINA chooses the nearest node to the backbone of the Hop-Tree, which is the shortest path built for E_1 to the Sink here, as the cluster head and builds a shortest path to the backbone. So, the distance of a packet transmitted from the cluster head of E_2 to the Sink should be $d_1 + d_2$. For the same scenario, ASARA chooses the nearest node to the Sink because $TTL < d_2$ and builds a shortest path to the Sink. Here, the distance of a packet transmitted from the cluster head of E_2 to the Sink should be d_1. Since $TTL < d_2$, the correlation between E_1 and E_2 is low. Without loss of generality we suppose the inter-cluster data aggregation ratio is 90% for E_1 and E_2 and the amount of data transmitted is N_{data} bytes for each event per transmission. Consequently, the energy consumption of data transmission from the cluster heads of E_1 and E_2 to the Sink for DRINA and ASARA is $C_{DRINA} = 0.9 \times 2N_{data}d_1 + N_{data}d_2$ and $C_{ASARA} = 2N_{data}d_1$ respectively. If $d_2/d_1 > 0.2$, $C_{DRINA} > C_{ASARA}$. The load of each node on the path from E_1 to the Sink for DRINA and ASARA is $d_1 \times 2N_{data} \times 0.9 = 1.8N_{data}d_1$ and $N_{data}d_1$ respectively. It can deduce that ASARA will relieve load imbalance which can easily causing network structure unstable when the correlation among events is low while DRINA will aggravate the load imbalance. For these reasons, ASARA leads to less transmission energy than DRINA when the correlation of events is low.

3) ASARA can achieve the same effectiveness as DRINA for data aggregation when the events are much correlated.

When the events are much correlated, they should be in close geographical proximity, within some hops. There is a relationship between correlation and hops for events according to the specific application, and it is reasonable that correlation of two events is measured by the number of hops they are apart. If two events are high correlated, they will be apart within say t hops. Set *TTL* as t, ASARA will make the paths of the events which are apart within t hops overlap as early as possible like DRINA.

Suppose event E_1 occurs followed by E_2 with high correlation separated by d hops after the sensor network configuration, where $d < TTL$. When E_1 happens, ASARA will choose the nearest node to the Sink as the cluster head and build a shortest path R_{ASARA1} to the Sink as DRINA does whose shortest path is denoted by R_{DRINA1}, with the length of R_{ASARA1} equal to R_{DRINA1}. After that, ASARA updates the HTS of each node within *TTL* hops from the shortest path while DRINA updates all nodes' HTS in the whole network. When E_2 happens, DRINA chooses the nearest node to the path R_{DRINA1} and builds a shortest path R_{DRINA2} to R_{DRINA1}. Because $d < TTL$, ASARA acts as DRINA, choosing the nearest node to the path R_{ASARA1} and building a shortest path R_{ASARA2} to R_{ASARA1}. It is easy to know that the length of R_{ASARA2} is equal to R_{DRINA2} and the degree of overlap is the same for the two algorithms in this situation.

In a word, by setting an appropriate *TTL* according to the applications, that is *TTL* represents the correlation bound, ASARA can work as effective as DRINA when the correlation of events is high. But how to find an appropriate *TTL* is a further research for us, not discussed here.

4) ASARA can further balance data load by the forced path building process than DRINA.

If the data aggregation is not perfect aggregation, which means the output packet is one no matter how many packets to be aggregated, the overlapping parts of paths will have heavier load compared with the rest parts of paths, so the nodes in them consume energy faster and then die prematurely. The premature death of nodes brings about unstable network structure, exacerbating network energy consumption. There is a tradeoff between the energy balance and data aggregation. In the path building phase, if the energy of nodes in the former path is under the threshold, ASARA will build a shortest path from the event to the Sink by force, not trending to overlapping the former path.

After network deployment, suppose there are events $\{E_i \mid t_i$ is the occurence time of i-th event, $i = 1, \mathrm{L}\ , N_E\}$, satisfying if $i < j$, $t_i < t_j$ and $\{E_i \mid 2 \le i \le N_E\}$ occur in the adjacent area of E_1, and the amount of data transmitted is N_{data} bytes for each event per transmission. The shortest path from the cluster head of E_1 is denoted by R. DRINA builds shortest paths for $\{E_i \mid 2 \le i \le N_E\}$ in sequence, and the load of some nodes on R grows heavier as more events happen. For example, if k events happen, the load of some nodes will be rkN_{data}, where r is the inter-cluster data aggregation ratio. If $rk > 1$, those nodes consume more energy than rest nodes. This will cause premature death of some nodes and unstable network structure. For ASARA, suppose t_l is the earliest time when the state of residual energy of some nodes on R lower than the threshold is discovered. It is easy to see that ASARA acts the same with DRINA on paths building until E_l occurs. When E_l happens, ASARA makes the involved nodes enter into the forced path building phase, and the built path does not trend to overlapping R but to the Sink directly based on better state nodes, relieving the load on nodes in path R, preventing these nodes from premature death and prolonging the duration of network structure staying stable.

Overall, if the events are close together and there is enough energy in correlated backbones, ASARA will work as the same with DRINA, and if the events are far apart with very low correlation, ASARA will build shortest paths from latter events not over-overlapping with the former events to save and balance energy consumption, and if there is not enough energy in the correlated backbones though the events are close, ASARA will build new paths not overlapping the low energy backbones by forced path building, balancing the network load and enhancing the stability of network. In addition, due to the Hop-Tree and path building and update based on the local states of nodes, ASARA can further balance the network resources and fit the heterogeneous network.

B. Simulation Evaluation

This section presents the performance evaluation for ASARA. Since ASARA is an improvement on DRINA, we compare it with DRINA using the following performance metrics:

- Time of the first died node: It is the time until the first node dies.
- Event loss ratio: It is the ratio of the number of events which are not monitored by the Sink to the total number of events.
- Number of events: It is the number of events that monitored by the Sink finally.
- Number of lost events: It is the number of events which are not monitored by the Sink.

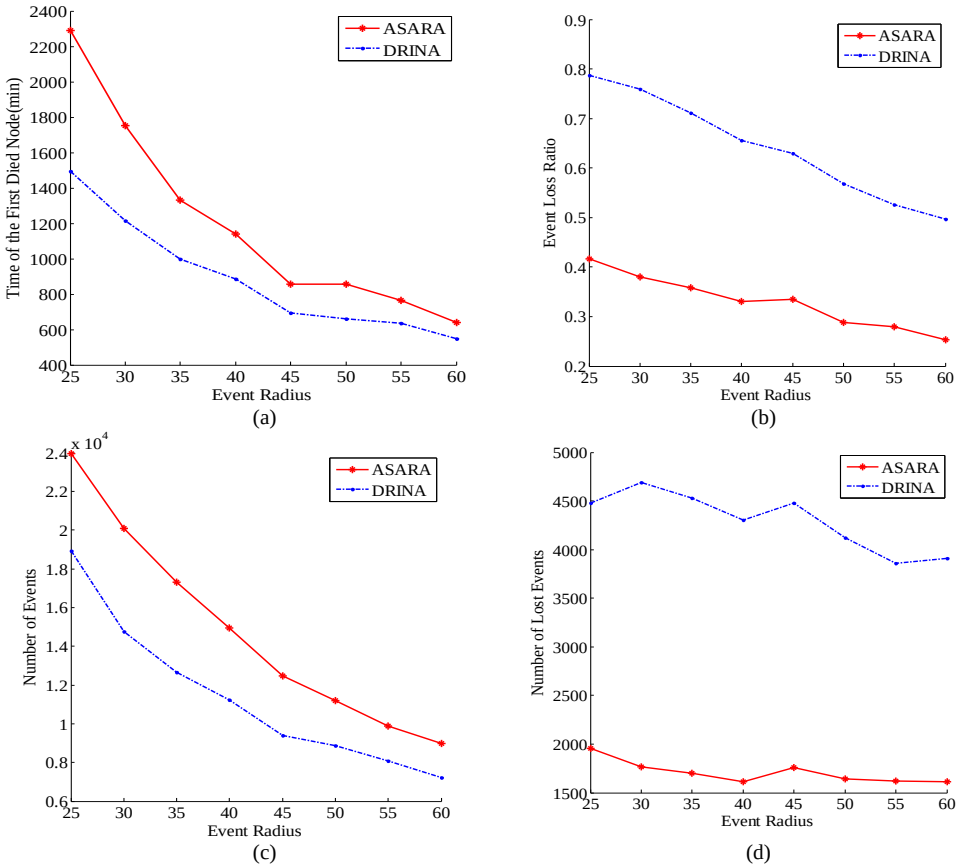


Figure 2. Event size (a) Time of the first died node (b) Final event loss ratio (c) Number of events monitored by the Sink (d) Number of lost events in 9000 minutes.

TABLE V. MAIN SIMULATION PARAMETERS

Parameter	Value
Network size	200m×200m
Number of nodes	100-400
Communication radius	20-40m
Initial energy	1J
Data packet size	4000bits
Control packet size	200bits
Data generating rate	1packet/min
E_{elec} in energy model	50nJ/bit
E_{amp} in energy model	100pJ/bit/m^2
Inter-cluster aggregation ratio	75%
Radius of event area	25-60m, Uniform distribution
Event duration	50min-200min, Uniform distribution
TTL	4 hops

In this work, the energy consumption model is adopted from [1]. Suppose that the shape of event area is circle, and the position, time and duration of an event is random. Both of ASARA and DRINA aggregate data in event clusters perfectly. Data from different events which are apart within *TTL* hops can be aggregated by a ratio *C* (let the original amount of data is *N*, the data amount after the aggregation is *C*×*N*), otherwise, inter-cluster data cannot be aggregated. The main simulation parameters are listed in Table V.

1) Event size

To evaluate the impact of the event size, we simulate 300-node and 200m×200m network, fixing the communication radius at 35m and increasing the event radius from 25m to 60m. The results are shown in Fig. 2.

As a general result, ASARA outperforms DRINA. Fig. 2(a) shows that the time of the first died node, for both of ASARA and DRINA, declines with the increasing event radius. This is due to heavier load as event radius increasing. And the time gained by ASARA is longer by average 31% compared to the DRINA. The reason is that ASARA can better balance the energy consumption and other resources by using local state to build and update Hop-Tree and further balance data load on overlapping paths by forced path building mechanism. ASARA can tradeoff the data aggregation and energy balance by overlapping the paths from high correlated events as early as possible to maximize the possible data aggregation, limiting Hop-Tree update scope through *TTL* to avoid over-overlapping which could cause longer paths, and building shortest paths forcedly from events to the Sink to step aside the low energy backbones of the Hop-Tree to balance the data load, finally leading to the lower event loss ratio and more events monitored by the Sink. The final event loss ratio of ASARA is much lower than DRINA, about 48% in average, shown in Fig. 2(b), and the number of monitored events of ASRAR is average 39% more shown in Fig. 2(c). Fig. 2(d) describes the number of lost events in 9000 minutes for different event radius. In average, the number of lost events of ASARA is less than that of DRINA by 60%.

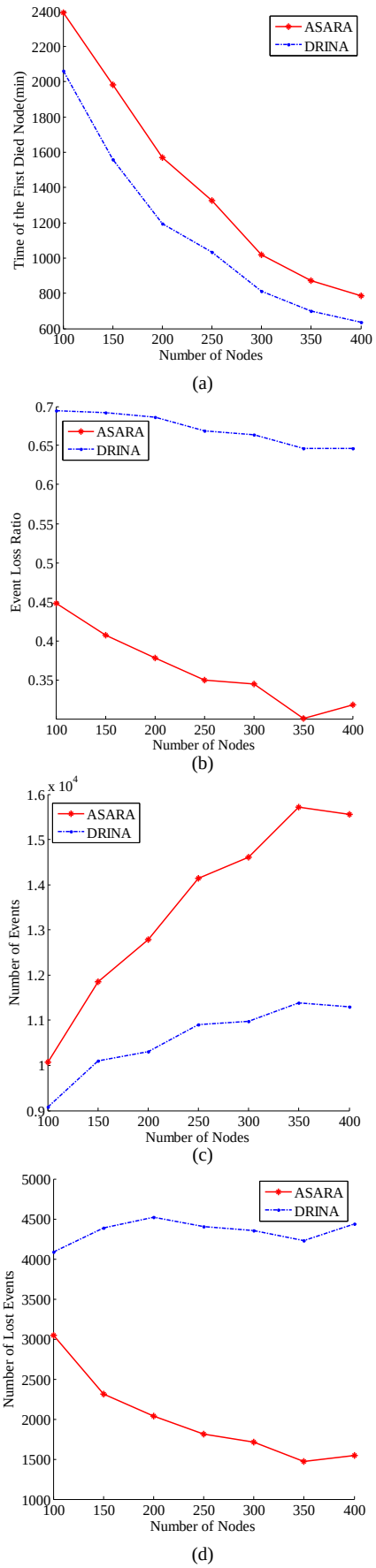


Figure 3. Density (a) Time of the first died node (b) Final event loss ratio (c) Number of events monitored by the Sink (d) Number of lost events in 9000 minutes

2) Density

Here, we evaluate the density impact by keeping the sensor field (200m×200m), communication range (35m) and event radius (40m) constant and varying the number of sensor nodes from 100 to 400 to vary density.

Through Fig. 3, when density varies, we find that ASARA still works better than DRINA. As the density varies, comparing ASARA with DRINA in average, the time of the first died node is 25% longer, the final event loss ratio is 45% lower, the number of monitored events is 27% more, and the number of lost events in 9000 minutes is 53% less, shown in Fig. 3(a)-(d). The reason is similar to the one described in the previous section. From Fig. 3(d), we can see that the number of lost events decreases obviously as density increases for ASARA. That is because the energy balance strategies taken by the ASARA can get more benefits from the increased density.

3) Communication radius

The impact of the communication radius is evaluated, by fixing network size at 200m×200m, the number of sensor nodes at 300 and the event radius at 40m. The results are shown in Fig. 4.

Since using the Hop-Tree to route data, the communication energy consumption grows as the communication radius increases, which directly results in the time of the first died node dropping, as shown in Fig. 4(a). Under different communication radius, ASARA also outperforms DRINA, with performance improvements in the time of the first died node, the final event loss ratio, the number of events monitored by the Sink and the number of lost events in 9000 minutes by 25%, 38%, 32% and 47% respectively. The number of events monitored by the Sink grows with increasing communication radius from 20 to 40. The number of events will fall when the communication radius increases to a certain value, which is not presented in our experiments.

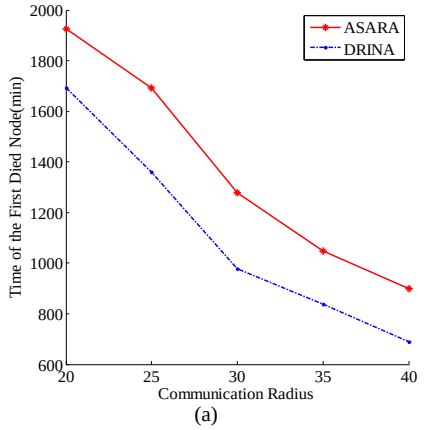
4) Network scalability

To evaluate the network scalability, we increase the number of nodes from 100 to 300, and resize the sensor field to keep a constant degree density of 14.4.

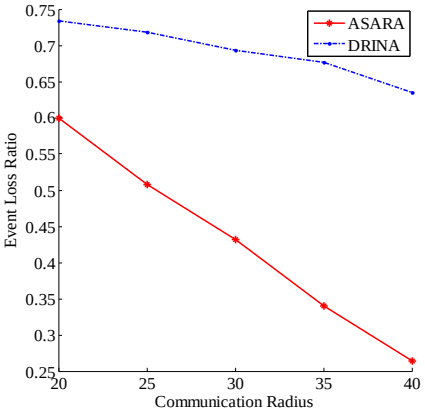
Fig. 5 shows that the ASARA is more scalable than DRINA. The reason is that the ASARA can save more energy and balance the load when the network becomes larger with stable node degree. ASARA performs better than DRINA, with improvements in the time of the first died node, the final event loss ratio, the number of events monitored by the Sink and the number of lost events in 9000 minutes by 26%, 37%, 18% and 40% respectively.

5) Event Loss Ratio At 20% nodes died

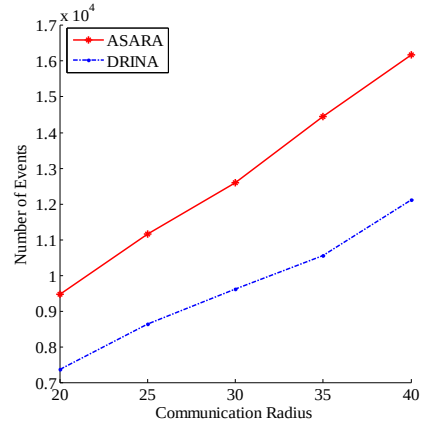
In order to further indicate the advantages of ASARA over DRINA, we also examine the event loss ratio when 20% nodes die, shown in Fig. 6(a)-(d), illustrating that ASARA has lower event loss ratio when 20% nodes die under different event radius, network density, communication radius and network scale.



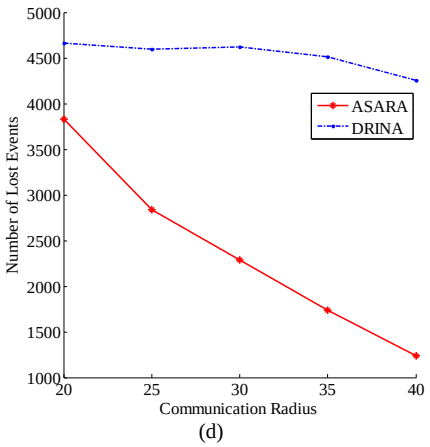
(a)



(b)

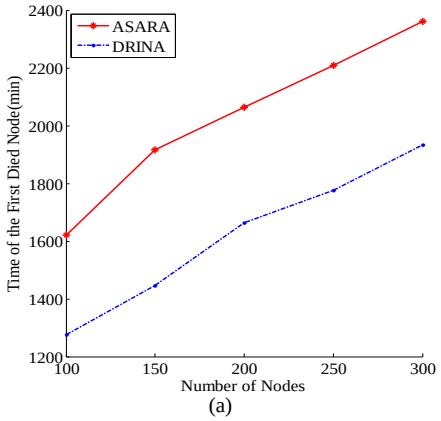


(c)

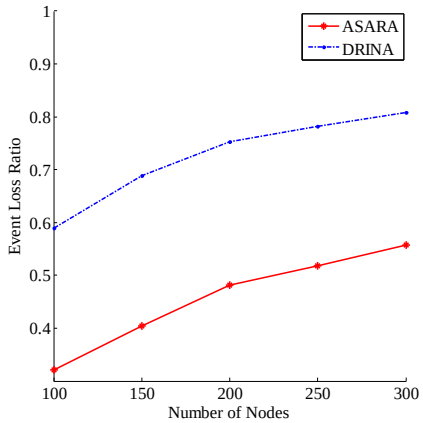


(d)

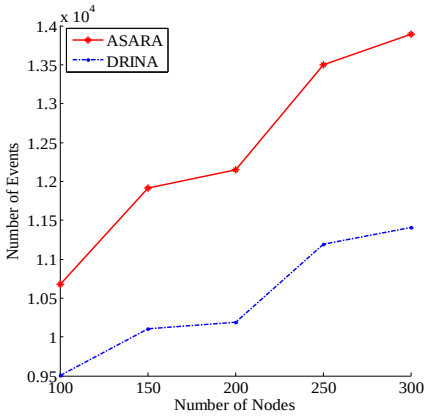
Figure 4. Communication radius (a) Time of the first died node (b) Final event loss ratio (c) Number of events monitored by the Sink (d) Number of lost events in 9000 minutes



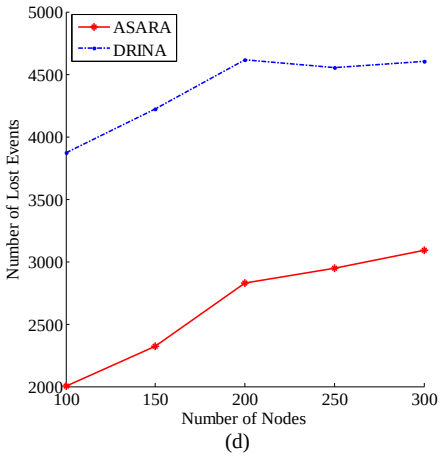
(a)



(b)



(c)



(d)

Figure 5. Network scalability (a) Time of the first died node (b) Final event loss ratio (c) Number of events monitored by the Sink (d) Number of lost events in 9000 minutes

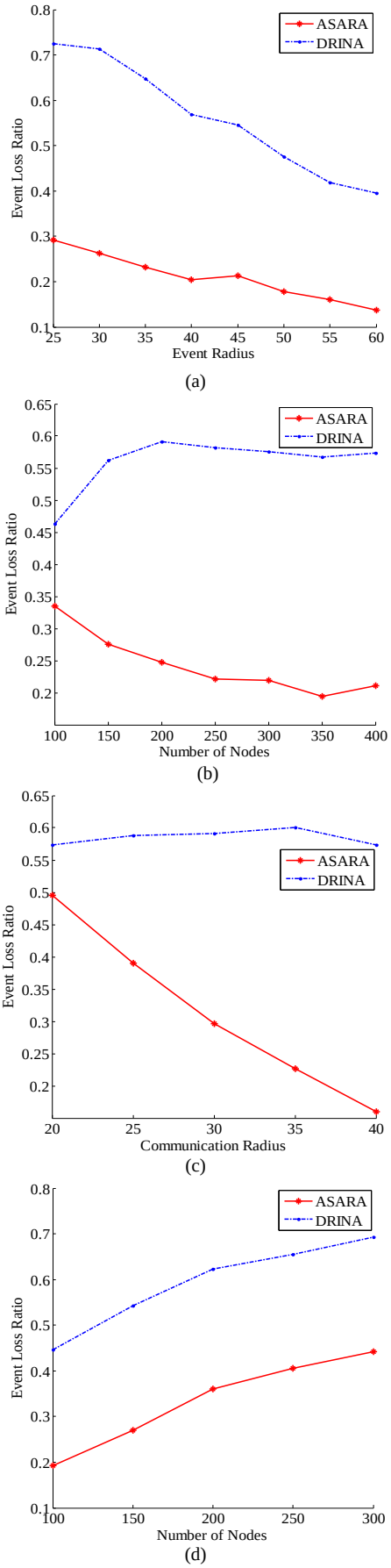


Figure 6. 20% nodes died (a) Event radius varies (b) Density varies(c) Communication radius varies (d) Network scale varies

V. Conclusion

In this paper, we study the routing problems for facilitating data aggregation in event-driven WSNs, and propose a novel Adaptive State-Aware Routing Algorithm for data aggregation (ASARA) to improve the DRINA algorithm designed by Leandro Villas *et al*. ASARA builds a Hop-Tree based on sensor local state calculated with residual energy, memory and link packet loss rate which is more appropriate to heterogeneous WSNs. It utilizes *TTL* to limit the update scope for the Hop-Tree, making the paths for the events with high correlation overlap as early as possible to maximize the degree of data aggregation and ones for the events with low correlation avoid over-overlapping and curving to save data transmission energy. If the energy of nodes on the path, which is intended to be overlapped by the latter path, is low, ASARA will build a new path forcedly to avoid overlapping with the paths in poor state, in order to balance the data load. Theoretical analysis and experiments show that ASARA can maximize the possible data aggregation while balancing the energy consumption, making WSNs monitor more events.

The paper does not consider how to gain a proper *TTL* according to specific application. Devising mechanisms for exploiting the data correlation to determine *TTL* dynamically is one of our future works.

Acknowledgment

This work was supported by a grant from Youth Fund of Luoyang Institute of Science & Technology (2008QZ11).

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, Institute of Electrical and Electronics Engineers Inc., vol. 1, no. 4, pp. 660-670, October 2002.

[2] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366-379, October 2004.

[3] S. Soro and W. Heinzelman. "Cluster head election techniques for coverage preservation in wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 5, pp. 955-972, July 2009.

[4] J. G. Yu, Y. Y. Qi, G. H. Wang, and X. Gu, "A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution," *International Journal of Electronics and Communications*, vol. 66, no. 1, pp. 54-61, January 2012.

[5] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. 22nd International Conference on Distributed Computing Systems*, Washington DC, USA, 2002, pp. 575–578.

[6] L. J. Meng, H. Z. Zhang, and Y. Zou, "A data aggregation transfer protocol based on clustering and data prediction in wireless sensor networks," in *Proc. 7th International Conference on Wireless Communications, Networking and Mobile Computing*, Wuhan, China, 2011, pp. 1-5.

[7] E. F. Nakamura, H. S. Ramos, L. A. Villas, H. A. B. F. Oliveira, *et al.*, "A reactive role assignment for data routing in event-based

wireless sensor networks," *Computer Networks*, vol. 53, no. 12, pp. 1980-1996, August 2009.

[8] W. S. Jung, K. W. Lim, Y. B. Ko, and S. J. Park. "Efficient clustering-based data aggregation techniques for wireless sensor networks," *Wireless Networks*, vol. 17, no. 5, July 2011.

[9] L. A. Villas, D. L. Guidoni, R. B. Araujo, A. Boukerche, and A. A. F. Loureiro, "A scalable and dynamic data aggregation aware routing protocol for wireless sensor networks," in *Proc. 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, New York, USA, 2010, pp. 110-117.

[10] L. Villas, A. Boukerche, H. R. Filho, *et al.*, "A. Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks," *IEEE Transactions on Computers*, vol. 62, no. 4, April 2013.

Yalin Nie was born in Yiyang, Hunan, China, August 1981. She received her B.S. in Computer Science and Technology in 2004 and M.S. in Computer application technology in 2007, both from School of Computer and Communication of Hunan University in Changsha, Hunan, China. Currently, she is working toward the Ph.D. in School of Computer Science and Technology of Xidian University in Xi'an, Shaanxi, China. She is also a teacher in Department of Computer and Information Engineering of Luoyang Institute of Science and Technology in Luoyang, Henan, China. Her research interest includes routing protocols, mobile computation and data aggregation in wireless sensor networks.

Prof. Sanyang Liu was born in Xi'an, Shaanxi, China, October 1951. He received his B.S. at Shaanxi Normal University in 1982, gained his M.S. at Xidian University in 1984, and received his Ph.D. from Xi'an Jiaotong Universty in 1989. At present, Prof. Liu is a PhD supervisor of Xidian University, the Dean of School of Sience of Xidian University and the Director of Institute of Industrial and Applied Mathematics. His research interest covers theory and application of optimization, network arithmetic.

Zhibin Chen was born in Zhangzhou, Fujian, China, December 1989. He received his B.S. from Xiamen University Tan Kah Kee College. Currently, he is working toward the M.S. in Microelectronics and Solid State Electronics at School of Microelectronics of Xidian University in Xi'an, Shaanxi, China. His research interest includes network modeling, algorithm analysis and design, and network optimization methods.

Prof. XiaoGang Qi was born in Baoji, Shaanxi, China, December 1973. He received his B.S. degree and M.S. degree in 1995 and 1996 from Institute of Equipment Command and Technology of Chinese People's Liberation Army, and gained his Ph.D. degree from School of Science of Xidian University in 2005. Now, Prof. Qi is an MS supervisor of Xidian University, the Director of the Society of Industrial and Applied Mathematics of Shaanxi, the leader and head coach of Mathematical Contest in Modeling of Xidian University. His research interest covers network optimization theory and methods, network algorithm analysis and design, and complex systems modeling and computer simulation.