Data Aggregation in Querying Range for Grid-based Sensor Networks

Tzung-Shi Chen¹ Yi-Shiang Chang¹ Hua-Wen Tsai² Chih-Ping Chu²

¹Department of Information and Learning Technology
National University of Tainan
Tainan 700, Taiwan
chents@mail.nutn.edu.tw

²Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan 701, Taiwan
chucp@csie.ncku.edu.tw

Abstract

This paper proposes an efficient data aggregation algorithm with range query capability for sensor networks. The proposed aggregation and query mechanisms are based on a virtual grid. In each grid, a head node is selected to be a manager. When a head node detects a generated event, it announces that to all other head nodes. A user, i.e. mobile sink, queries the interesting event via a head node, called agent, within the same grid. According to the received event type, the user issues an enquiry message to query the sensor network with a specific range. The user queries and aggregates the data in the regular-shape and irregular-shape range. The irregular-shape range is to aggregate the data of continuous event occurred. When the information is collected from the sensors, the information will be sent back to the user. Furthermore, we propose efficient approaches to gather the information from sensor networks while the void exists. Finally, experimental results show that our proposed approaches are more energy-efficiency than the existing approach.

Keywords: data aggregation, continuous event, query, tracking, routing, wireless sensor networks.

1. Introduction

Due to recent technological advance, the manufacturing of small and low-cost sensors has become technically and economically feasible [4]. General speaking, the applications for wireless sensor networks (for short WSNs) are quite diversity. For examples, WSNs have profound effects on military and civil applications such as target field imaging, intrusion detection, weather monitoring, security and tactical surveillance, distributed computing, and so on. The deployment of a sensor network in these applications can be in a random fashion (e.g., dropped from an airplane in a disaster management application) or manual (e.g., fire alarm sensors in a facility or sensors planted underground for precision agriculture). Creating a network of these sensors can assist the rescue operations by locating survivors, identifying risky areas, and making the rescue team more aware of the overall situation in a disaster area.

However, the most promising challenge in designing protocols is to make them more energy efficient by maximizing their lifetime due to the limited energy resource. A sensor network is mostly static and sensors are densely populated. The high density of the network may lead to multiple sensors generating and transmitting redundant sensed data, which results in unnecessary power consumption and hence significantly decreasing the network's lifetime. The actions of sensors, e.g., data transmission/reception and target sensing, result in a certain amount of energy consumption. Amongst them, the energy consumption for data communication is critical. Therefore, we have to use the most energy-saving communication link or minimizing the communication by eliminating or aggregating the redundant sensed data beforehand which saves a lot of energy and prolongs the lifetime of network ultimately.

In the previous researches, data dissemination protocols with the mobile sinks can be classified into two categories, tree-based and grid-based protocols. The two categories consider about how to aggregate data using mobile sinks in sensor fields efficiently. The first categories, tree-based protocol SEAD [3] and SAFE [4], have been proposed. SEAD is an approach for routing sensor's data to mobile sinks. SEAD is suitable for applications with less strict delay requirements. Another dissemination protocol, SAFE uses flooding that is geographically limited to forward the query to nodes along the direction of the source. SAFE find the gate to connect itself to the tree by geographical flooding. SAFE is effective in small-to-medium size sensor networks. However, in a very large network, the initial sensor flooding may consume too much energy. The d-tree would be frequently reconstructed due to mobile sinks.

The second categories are grid-based protocols, e.g. TTDD [9] and CODE [8]. TTDD and CODE are the event-driven data dissemination methods using mobile
sinks. TTDD relays the query and data by dissemination nodes. But it is not optimize the path from the source to the sinks. When a sensor communicates with a sink, the restriction of grid structure may multiply the length of a straight-line path by $\sqrt{2}$. This approach therefore incurs more energy and longer delays. The mobile sink updates its entire path to the dissemination point whenever it moves out of range in the local cell. Frequent renewal of path may increase energy consumption and the connection loss ratio. TTDD can work well with event-driven systems where sources are queried on demand, but it will not be suitable for applications where sources generate burst data. In such circumstances, the entire path would have to change when the sink switches to another dissemination node. It often happens in small cells. Large cells on the other hand increase the energy consumption due to the local flooding.

Another protocol, named CODE [8] has on demand query by the source. When a sink needs data, it sends a query to a coordinator in the same grid where it is locating. Based on the location of target and grid IDs, an efficient data dissemination path is established while the query traverses directly to the source. If the sink moves out original grid to another grid, it sends cache-removal message to clear out the previous data dissemination path and then re-sends a query to set up a new one. But CODE does not solve the routing problem with the obstacle or the void. Although CODE and TTDD use the mobile sinks to gather data in wireless sensor networks, the sinks have to renew a query to request data or using local flooding to request data when they move out the original grid. By this way, that will increase the overhead on operation. It means to increase energy consumption and the collision in the network. CODE has improved the drawback of TTDD to gather the reported data efficiently.

In this paper, we propose the approaches with the regular-shape and irregular-shape range to improve the CODE. Moreover, we combine the data aggregation and data dissemination to gather data with the different applications and energy efficiently. For example, a group of mountain rangers (mobile sinks) collects the occurred event from a sensor network deployed in the forest. The sensors detect the event from their surrounding and aggregate the sensing data to report the mobile sinks (mountain rangers) who query the regional range of detected a stimulus. By this way, the mountain rangers can judge this area what happened and how to handle properly.

We have three different characteristics compared to CODE. First, we construct the data link of tree structure to track the continuous event. Second, we think about the sink left the original grid and reconstruct data link. Third, we use the proposed protocols studied in [2][5] to solve the sensor field of the void. We can aggregate the data of regional range successfully based on a constructed virtual tree structure for continuous events while the void exists. As our best knowledge, this paper is not only the first protocol to combine with the data dissemination and data aggregation, but also it solves the problem of tracking the continuous event and data aggregation in the void.

The rest of this paper is organized as follows. In Section 2, we present the data aggregation in querying range. In Section 3, we present how to deal with the data aggregation in querying range while the void exists. We simulate the experiment in Section 4. In Section 5, we conclude our method.

2. Data Aggregation in Querying Range

First, we present the basic design of our protocol with the following assumptions. A monitoring field is deployed by a large number of homogeneous sensor nodes which communicate with each other through short-range radios. Sensor nodes remain stationary at their initial locations. Each sensor node is aware of its own location and has the limited battery energy. Sensors also know their assignment for monitor. When the sensor nodes are deployed the monitoring regions, they have global time synchronization. The sensor nodes have varied missions to detect some interesting events. Mobile sink(s) (users use mobile devices such as PDAs) query the regional range to collect and to aggregate the sensing data. There can be multiple sinks moving around in sensor fields and the number of mobile sinks may vary over time. Mobile sinks also know their locations by themselves.

2.1 Virtual Grid Structure

Because the sensor is aware of their physical location, we design the side length of grids to guarantee that the grid heads among each pair of neighboring grids can communicate directly with each other [1]. The monitored area (sensor field) is divided into virtual grids. Each node associates itself with a virtual grid depending on its physical location. Every sensor has its grid ID. We note $\text{Grid}_{(x, y)}$ as the grid coordinate, where $x$ is grid x-coordinate and $y$ is grid y-coordinate. When the sensors have been deployed, each grid selects the manager of grid (called head) in order for sensing, receiving and relaying data.

2.2 Query Forwarding

When a sensor (source) detects the interesting event, it propagates a register packet $<\text{Type, Src_id, Loc}_{(x, y)}, \text{Grid}_{(x, y)}, hc, \text{event_type, timeout}>$ to all heads using simple flooding (called data announcement), where $\text{Src_id, Loc}_{(x, y)}$ and $\text{Grid}_{(x, y)}$ are the source's information. Receiving the packet, heads store the information for the data dissemination path discovery in the register table, including the information of the event and the source location. In this approach, the sink easily requests and aggregates the data by source location. In order to avoid keeping register packet at each heads indefinitely, source includes a timeout parameter in register packet. If this timeout expires and a head does not receive any further register packet, it clears the information of the event and the source's location in the register table to release the cache.

When a mobile sink (for shorted as MS) want to know an interesting event, it sends a request packet to ask the head in the same grid. The head receives the request packet and checks its register table to see whether the event has occurred, and it sends reply packet
neighboring head that has the shortest distance to the greedy-forwarding) that refers to forwarding a packet to the simplest form of geographic routing [5], i.e., regular or irregular shape ranges. Our method uses the sink queries the source about a specific range that is the regular or irregular shape range. The method uses the simplest form of geographic routing [5] (i.e., greedy-forwarding) that refers to forwarding a packet to the neighboring head that has the shortest distance to the source. When the source receives the reply packet, it aggregates the data of regular-shape range (aggtype set to 1) or irregular-shape range (aggtype set to 0).

2.3 Querying Regular-Shape Range

We describe the method of data aggregation with the regular-shape range briefly. The mobile sink can collect and set the regional range from four directions, e.g., the range of the northwest $R_{x,y}$ is set (-2, +2) in Figure 1(a), the range $R_{x,y}$ of the southeast is set (+2, +2) in Figure 1(b), the range $R_{x,y}$ of the southwest is set (-2, -2) in Figure 1(c), and the range $R_{x,y}$ of the northeast is set (+2, -2) in Figure 1(d).

![Figure 1: Data aggregation with different four directions](image)

The source receiving the query packet constructs a data link of comb-like tree to aggregate data. The detailed procedure is described in the following steps:

Step 1: The source broadcasts the askdata packet <Type, B_Grid(\(x, y\)), S_Grid(\(x, y\)), event_type, R_{x, y}> to the neighboring heads, where B_Grid(\(x, y\)) is the broadcasting node’s grid, and S_Grid(\(x, y\)) is the source’s grid.

Step 2: When a node receives the askdata packet, it judges whether its grid’s location is in the requested regular-shape range. If yes, it runs Step 3. If no, it stops to forward packet.

Step 3: If its grid is not S_Grid(\(x, y\)) = R_x, it broadcasts the packet again and records the broadcasting node as its parent. And the process repeats step 2. If yes, it runs Step 4.

Step 4: The node starts to send the sensing data. And then the data is sent to source along the tree.

Step 5: When the node receives all the sensing data from its children, it aggregates their and its sensing data. Finally, it sends the aggregation data to its parent until the source node.

Here, we illustrate collecting the northwest regional range with an example. In Figure 1(a) when the source A receives the query packet (aggtype is 1) with the range size R(-2, +2), it broadcasts the askdata packet with the range size (-2, +2). The heads B and D receive the packet from the source A and they compare their relative positions of source A. And the heads B and D record the source A as their parent node. Because the heads B and D are not the leaf node, so they broadcast the askdata packet again. The head C receives the broadcasting packet from head B, and it records the header B as its parent node. The head C is the leaf node (S_Grid.x + R_x), S_Grid.x is 3 and R.x is -2 in the regional range, C has to relay its sensing data to its parent node B. Similarly, when head D receives the header A broadcasting packet, head D records the header A as its parent node and it broadcasts the packet again. When heads G and E receive the broadcasting packet form head D, they also record the header D as their parent node and broadcast the packet again. Similarly, when head H receives the broadcasting packet from G, it also records the header G as its parent and head F records the header E as its parent. But the header F is the leaf node in the regional range, so it stops to broadcast the packet and starts to relay its sensing data to its parent node E. Similarly, the header I receives the broadcasting packet from head H and relays it as its parent node. Because header I is leaf node in the querying regular-shape range, head I sends the sensing data to its parent H. When the source A receives all aggregated data from its child nodes, it aggregates their and its data and sends the final aggregated data to MS.

2.4 Querying Irregular-Shape Range

2.4.1 Continuous Event Detection and Construction

The second method of data aggregation with the irregular-shape range is presented in this subsection. Due to the event has the continuous or extensible characteristic, we construct the tree structure to track the continuous event. The method is described as follows. When a head detects an occurred event, it first looks up its register table to check whether the same registered event has been detected in its adjacent grid or not. If yes, it sends the askdata packet <Type, N_id, Loc(\(x, y\)), Grid(\(x, y\)), P_id> to the heads of its adjacent grids, where N_id is its id and P_id is parent id. If the register table has more than two registered locations in its adjacent grids, it selects the node with the hop count as its parent. If no, it broadcasts the register packet to all network (described in section 2.2). When the parent node receives the askdata packet, it records the forwarded node as its child, and sends the ack packet <Type, N_id, Loc(\(x, y\)), Grid(\(x, y\)), C_id, he> to child, where N_id is its id, C_id is child id and he is hop count. Also, the child head receiving the ack packet records that as its parent. Then it sends the notify packet to the heads of its...
adjacent grids (the east, west, south, and north of its grids). The neighboring heads receiving the notify packet <Type, N_id, Loc(x, y), Grid(x, y), he, event_type, timeout> have to store the information in the register table. If the source receives the notify packet, it discards the packet. By this way, when the event is spreading continuously, the continuous event will be tracked with constructing the tree structure efficiently. With this method, the mobile sink can aggregate the information of the continuous event easily.

In Figure 2, when the continuous event spreads to the Grid(3, 3) and Grid(3, 4), the heads B, C and D will detect the occurred event. They look up their register table and check whether the same event has been registered. Due to source A has registered the event in their table, they send datalink packet to the source A. When source A receives the packets from B, C or D, it replies the ack packet (hc is 1) to them respectively and records them as its children. When the children nodes B, C and D receive the ack packet, they broadcast the notify packet to their adjacent grids instantly. B broadcasts the packet to the heads of its adjacent grids (J, I, H and A). But node A is the source, it discards the packet. The other heads (J, I and H) have to store the packet in their register table. Similarly, C and D broadcast the notify packet to their adjacent grids. If the event spreads to the Grid(2, 2) and Grid(2, 3), the heads F and H will detect the spread event instantly. And then, they select a node that is the less hop count to source A and send datalink packet to that node. If the hop count is equal, it chooses the head of the latest register and sends datalink packet to that node.

2.4.2 Data Aggregation in Irregular Shape

In Figure 3 (a), the continuous event has been tracked with constructing tree structure in the grids. Source A receiving the query packet (aggtype is 0) and it sends a askexdata packet <Type, Src_id, S_id, event_type, C_id> to its children B and C, where Src_id is source id, S_id is sink id and C_id is child id. And then, nodes B and C send askexdata packet to their children D, E, F and G again. Because the nodes D, F and G are the leaf nodes, they send the sensing data to their parent nodes. Node E sends askexdata packet to its children I and H. Nodes I and H are the leaf nodes, so they send the sensing data to their parent node E. In Figure 3 (b), when node E receives the data from children I and H, it has to aggregate the sensing data of them and itself. Then node E sends the aggregated data to its parent node C. When node C receives the all sensing data from its children, it also sends the aggregating data of them and itself to its parent node A. Similarly, node A receives the sensing data from its children B, F and G. Then Source A sends the final aggregating data to mobile sink along the data path. In data aggregation of our method, we also relay the location information of each child in the tree.

2.5 Data Dissemination

The nodes receiving the data packet in the path has to cache data for a moment in their data table. The data packet also includes a TTL parameter. If the TTL expires, a head does not receive any further data packet from the source. It clears this data in the table and releases the cache memory. When the cached node receives a query packet from other MS that queries the same source with the same condition in different time, it replies the cached data packet to the MS instantly. All nodes on the data path between the source and MS have to cache the data in their data table.

2.6 Handling Sink Mobility

The mobile sink may want to gather the data in monitoring regions continuously. We also consider about
this problem with the mobile sink. As following Figure 5 (a), the mobile sink (MS) checks its location in every second. If the mobile sink detects that it moves out the original grid, it selects the head as new agent in the new grid. And then it sends move message to the new agent. The new agent receives the packet and checks itself whether it is the node of the data path or not. If no, it sends the move message to the old agent in order to construct the new data link. Finally, the old agent relays the data to the new agent node while it receives the data, and then the new agent node sends the data to the mobile sink again.

In Figure 5 (b), because the new agent node is in the data path, it does not send move message to old agent. On the contrary, it sends the remove message to delete the data path between the new agent and the old agent in order to avoid the problem of the loop.

![Figure 5: Handling sink mobility](image)

### 3. Data Aggregation in Querying Range with Void

#### 3.1 Void Discovery

In this section, we present a method to solve the void problem. At First, each sensor knows the neighboring information in their neighbor table. Due to this information, the node that is nearest the central of the grid is selected as head. Second, in order to achieve planarity on the graph, we employ the Gabriel Graph to construct the graph for sensor network as shown in Figure 6 (a).

![Figure 6: Void discovery](image)

Third, each head uses right-hand rule (clockwise rule) [5] to send the face packet in order to collect the face's information as shown in Figure 6 (a). The packet contains the following information: `<Type, N_id, Loc(x, y), Grid(x, y), Rly_id, face_list, hc>`, where N_id is issued node, Rly_id is the relay node, face_list is the list of the node's information in the face. If a head collects four nodes in the face_list of the face, it denotes the face does not the obstacle or void. The face packet is discarded and does not need to store in its face table. Otherwise, it denotes the face have the obstacle or void. Then the head has to store the face packet in its face table. As following in Figure 6 (b), we find the void (F) in the monitoring area. F is composed of nodes A, B, C, D, E, F, G, H, I, J, K, L, M and N. By this way, each head in the F knows the face's neighboring information.

#### 3.2 Query Forwarding with Void

If the forwarding node cannot find a node which the distance to the source is shorter than it has, it finds a node that is the shortest distance to the source in the face from its face table. And then it delivers the packet to the next node with the right-hand rule until the node of shortest distance to the source in the F received that (called face routing). Finally, the last node in the face also use greedy-forwarding to relay the packet until the source received that.

In Figure 7, the mobile sink sends the query packet using the greedy-forwarding to the source node. Node K cannot find a node which the distance to the source is shorter than it has. Then node K checks its face table and finds node C which the distance to the source is the shortest in the face. K forwards the packet along the path K→L→M→N→A→B→C using the face routing. When node C receives the query packet, it sends the packet to the source using the greedy-forwarding.

![Figure 7: Query forwarding with void](image)

#### 3.3 Querying Regular-Shape Range with Void

In this subsection, we consider the data aggregation of regular-shape range while the void or the obstacles exist. We construct the comb tree link to request the data aggregation in the regions of void. The detailed procedure is described as the following step: Notation: N_Grid.x, N_Grid.y is denoted the neighboring coordinates in neighbor table; F_Grid.x, F_Grid.y is denoted the face's neighboring coordinates in the face table.

**Step 1:** The source or node computes the child node in the regular-shape range. And then it forwards it sends askdata packet to its children.

**Step 2:** If the node cannot find its children from its neighbor table, it finds out the parallel F_Grid.x or F_Grid.y in the requested range from the face table. Then, it sends askdata packet to them using right-hand rule.

**Step 3:** When the node receives the askdata packet, it records the sending node as its parent node.
Step 4: If the node is not a leaf node, it sends the askdata packet to its child. It repeats to run Step 1. Otherwise, it starts to send the sensing data to its parent node.

Step 5: When the node receives all the sensing data from its children, it aggregates them with its sensing data. Finally, it sends the aggregation data to its parent until the source node received that.

As shown in Figure 8(a), the source $L$ receives the query packet including the range $R_{(2,5)}$. It first computes the next nodes $K$ in Grid$_{(3,4)}$ and $M$ in Grid$_{(3,3)}$ from its neighbor table, and sends askdata packet to them. They receive the packet from source $L$ and record it as their parent node. Node $M$ computes the next forwarding node $N$ and sends askdata packet to that node. When the leaf node $N$ receives the packet form its parent node, it starts to relay its sensing data to its parent. Similarly, node $K$ computes the next node in x-coordinate grid (i.e. Grid$_{(3,4)}$) from its neighbor table, but there is not any node in the Grid$_{(3,4)}$. So, it finds the parallel x-coordinate grid from its face table. $K$ finds a node in Grid$_{(3,3)}$ and then it sends the packet to node $A$ using face routing along the path $K$→$L$→$M$→$N$→$A$. When node $A$ receives the askdata packet, it records the node $K$ as its parent node. Because the node $A$ is a leaf node, it starts to relay its own sensing data to its parent node $K$ along the path $A$→$N$→$M$→$L$→$K$ as shown in Figure 8(b). When node $K$ receives the sensing data from its child $A$, node $K$ aggregates $A$’s and its data. And then $K$ sends the aggregated data to its parent $L$. Similarly, node $M$ aggregates and sends the data to its parent node $L$, when it receives the sensing data from its child $N$. When the source $L$ receives all aggregated data from its children, it aggregates their and its data and replies the final data to MS.

![Figure 8: Querying Regular-shape range in northwest direction with void](image)

### 3.4 Continuous Event Detection and Construction with Void

The continuous event may spread through the obstacle (or void) to the grids that are in the opposite side. We guarantee that our method can also construct tree structure for continuous events while the void exists.

In Figure 9 (a), the source $L$ detects the occurred event and broadcasts the register to all heads (called data-announcement). When the event spreads to the contiguous Grid$_{(3,2)}$, Grid$_{(4,3)}$ and Grid$_{(3,3)}$, the heads $P$, $K$ and $O$ send data-link packet to their parent node $L$. Then source $L$ sends ack packet to them. When $P$, $K$ and $O$ receive ack packet, they send notify packet to the heads of their adjacent grids (described 2.4.1 in detailed). When the event spreads to Grid$_{(2,4)}$, head $A$ sends data-link packet using right-hand rule to check if the same event had occurred in the face. Head $K$ receives the data-link packet from $A$ and it records node $A$ as its child. $K$ also sends ack packet to construct the virtual link along the original path. Head $A$ receives the ack packet and records $K$ as its parent. And then $A$ also sends notify packet to the heads of its adjacent grids.

Similarly, the event spreads to Grid$_{(4,6)}$ shown in Figure 9 (b). As above describe, head $E$ also sends data-link packet using right-hand rule to check if the same event had occurred in the face. Because head $K$ has the child $A$ in the face already and the child $A$ is not in the adjacent grid, $K$ sends change message to the original child node $A$ in order to change $A$’s parent node as $E$. $K$ also replies the ack packet to $E$ along the query path. Node $E$ receives the packet and records $K$ as its parent and $A$ as its child. Also, head $E$ broadcasts the notify packet to the heads of its adjacent grids. Finally, the virtual tree structure has been constructed. In Figure 9 (b), source $L$ has children $P$, $K$ and $O$; head $K$ has the child $E$; head $E$ has child $A$. We use this method to avoid the circle of the virtual data link. Moreover, if the event spreads to grid$_{(2,4)}$ and grid$_{(4,6)}$ at the same time, they also use the same method to construct the data link and we guarantee that it never cause the circle of link. By this way, the mobile sink can aggregate the information of continuous event efficiently while the void exists. Moreover, the mobile sink can track the location information of continuous event.

![Figure 9: Constructing the link of continuous event with void](image)

### 4. Experimental Results

#### 4.1 Simulation Model

Data dissemination protocols with grid-based CODE [8] and TTDD [9] protocols have been proposed. CODE has improved the drawback of TTDD and gathers the reported data efficiently. We propose the approaches of irregular-shape range to improve the CODE. First, we compare CODE with our approach using ns-2 [6]. Then, we discuss that the different number of sinks and percentage of void grid impact on the querying regular-shape range. We use the energy model used in ns-2 that requires about 0.7W, 0.35W and 0.035W for transmitting, receiving and idling, respectively. The initial
battery status is equal to $10J$. The simulation uses MAC IEEE 802.11 DCF that SENSE implements. We consider about the actual transmission range of mica2dot; we set the transmission range of each node is $112m$ in ns-2. In order to ensure that the adjacent grids can communicate each other, we set the grid size is $50m$. By this way, we assume that the ratio of sensing range and the communication range is $1:2$ [7]. The grid is fully covered with the sensing range of nodes. The speed of mobile sink is $2m/s$ and the mobility model is according to random way-point model. The sensor network consists of 200 sensor nodes, which are deployed uniformly in a $500mx500m$ field (i.e. two sensor nodes per $50mx50m$ grid). Two-ray ground is used as the radio propagation model and an omni-directional antenna having unity gain in the simulation.

with the number of spread event due to constructing the track link. In addition, our approach also decreases the energy consumed as shown in Figure 11 (b).

4.2.2 Impact of regular querying range

We assume that the speed of the sink is $2m/s$ and sink queries and aggregates the data with varied range in this simulation. In Figure 12, we observe the delay time and the querying energy consumed increases with the aggregating range. If the mobile sink is to query a bigger regional range, the delay time and energy consumption will follow to increase.

4.2.3 Impact of the number of sinks

The sensor network consists of 200 sensor nodes, which are deployed uniformly in a $500mx500m$ field (i.e. two sensor nodes per $50mx50m$ grid). The number of sinks is varied from 1 to 8. In addition, 8 events are occurred in wireless sensor networks. The speed of sink is $2m/s$ and the sink collects the events with the range $2*2$ grids. We use the method of cached data to save the energy consumption. We assume that the sink queries and collect data before the source registers are expired. Source has to register the occurred event in every 10 seconds. Eight kinds of different events existed in the network, so there are eight sources in different areas to announce data by turns. The simulation time of a run is 100 seconds. In Figure 13(a), the average delay time of cached method is reduced while the sink increased. This is because that the number of nodes that cached data is increased. When other sinks query the same data from the same source in different time, they can obtain the data from the relay node (the node cached data) of the data path. By this method, we reduce not only the delivery time but also the resource consumption. The result of energy consumption is shown in Figure 13 (b).

4.2.4 Impact of the ratio with void

We simulate the impact of the ratio with void in $500mx500m$ field and deployed 100 sensors in this field. And we simulate that the different percentage of void grids impacts on performance. We assume that the sink queries and collect data before the source registers are expired. The parameters of register, simulation time and source deploy are the same as that in subsection 4.2.3.
We vary the density of void grid from 10% to 20% in the network. The query range is also set as 2*2 grids. Delay time is increased while the density of void grids is increased as shown in Figure 14 (a). This is because that existed void grids impact and increase the hop count of routing. In high void grid density, the energy consumption is to reduce slightly as shown in Figure 14 (b). This is because that the success rate in 20% density of void grid is low than that in 10% as shown in Figure 14 (c). Some packets are not delivered successfully, so the energy consumption is to reduce slightly. In this scenario, the mobile sinks move with speed 2m/s according random waypoint model. Therefore, the mobile sink cannot obtain the information of data announcement when it is in the void grid. When the density of void grid increases, the success ratio is reduced in Figure 14 (c).

![Figure 14: Success ratio with different numbers of sinks](image)

5. Conclusions

In this paper, we introduced the data aggregation in querying range. First, we construct data link to collect the regular shape range data efficiently. By the data link, we can avoid aggregating the redundant data in desired range. Secondly, we construct the tree structure to track the continuous event. By the tree structure, we can query and collect the irregular shape range that covers the spread of the continuous event. We can track the spread condition of continuous event that include the location and the data of events. We also consider the problem of multiple sinks that query the same data from the same source in different time. We use the cached data method to reduce the delay time and the energy consumption for querying. Moreover, we have solved the problem of existed void grid in the sensor field. We use the information of face routing to overcome the problems that include packet routing, data aggregation and tracking in the continuous events. The experimental results showed that our approach not only reduces the energy consumption but also increases the performance of tracking continuous event and multiple mobile sinks effectively. In addition, our approach handles the routing and aggregating problem with the void grids.

References